Working with the Rules Engine

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

  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.

- Rules Engine

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

- Rules Engine Categories.

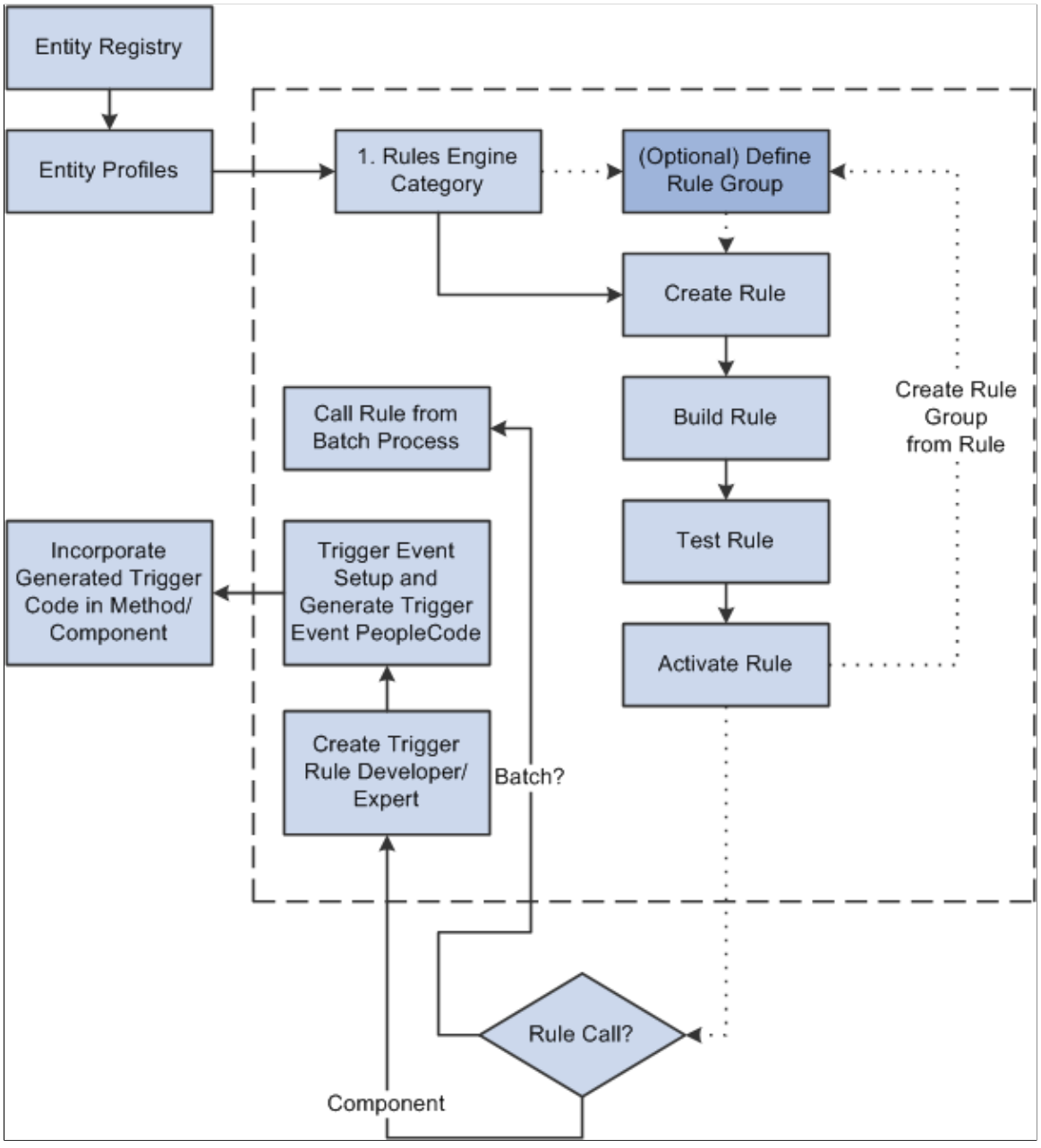
Rules Engine Categories are used to restrict access to pre-defined and secured areas of application functionality including access to Rules in other Rules Engine Categories and what types of Rules can be created: Triggers, Functions or Rules.

High Level Description of the Rules Engine

This diagram shows Rules Function functionality:

Image: Rules Engine Business Process (Generic)

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.

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 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 document as examples of how Entity-based data can be used in the Rules Engine.

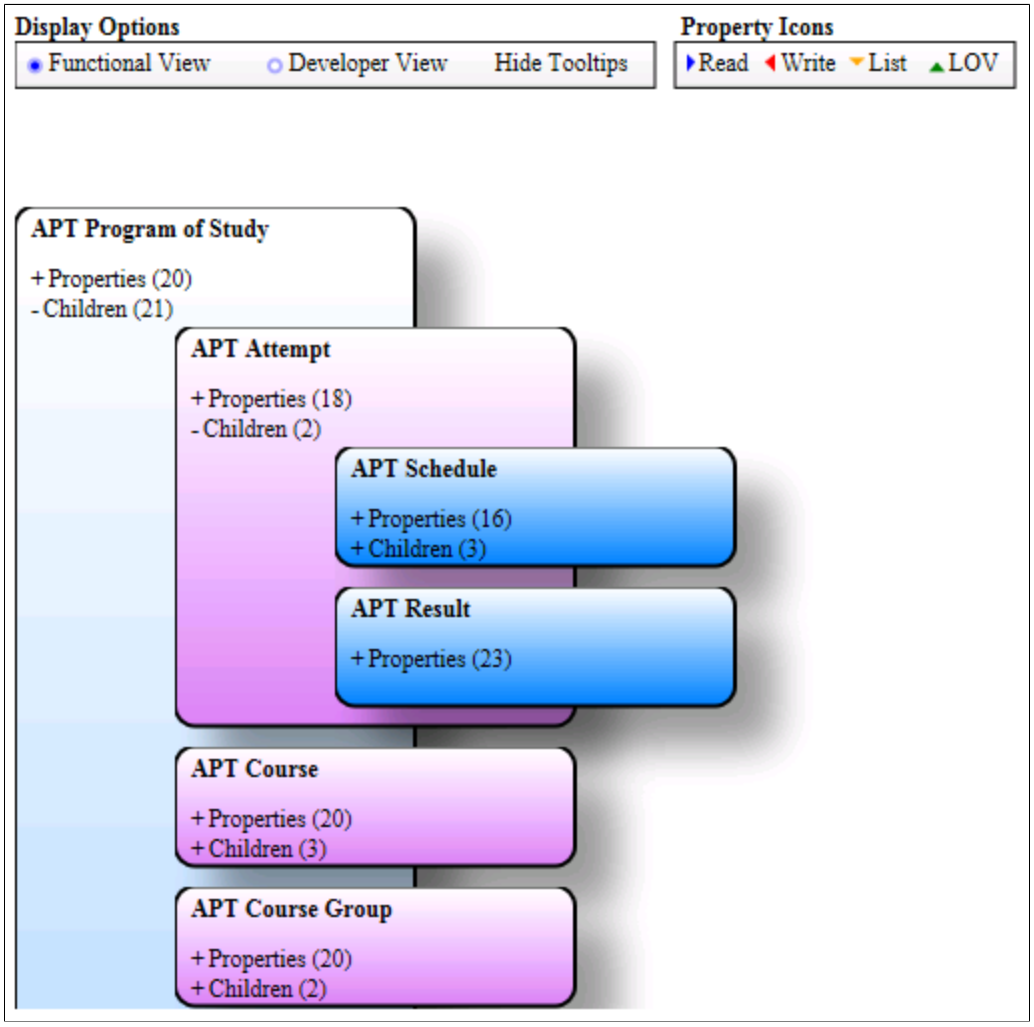
For more information about AIR and APT, see:

- "Understanding Program Enrollment" (PeopleSoft Campus Solutions 9.2: Student Records)
- "Understanding the Academic Progress Tracker" (PeopleSoft Campus Solutions 9.2: Student Records)

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

Image: Academic Progress Tracker Program of Study Entity Registry Hierarchy example

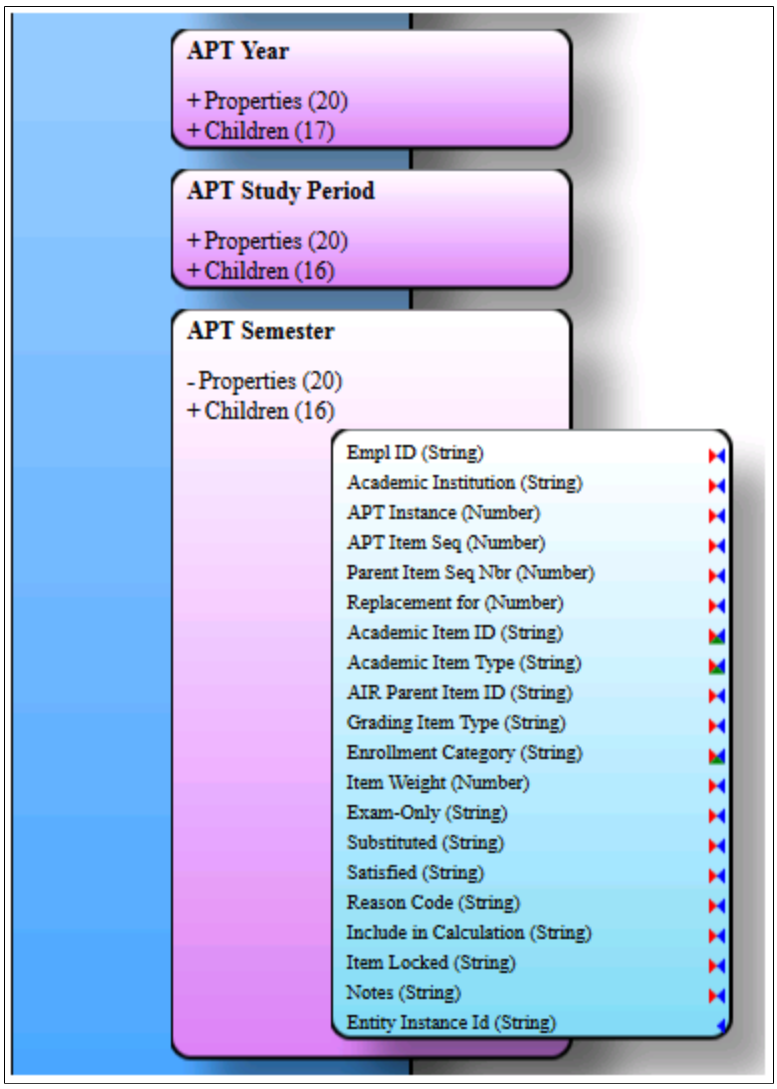
This example illustrates the fields and controls on the Academic Progress Tracker Program of Study Entity Registry Hierarchy example. You can find definitions for the fields and controls later on this page.



Here is an example that shows the entities APT Year, APT Study Period, and APT Semester. The properties of APT Semester are expanded and shown:

Image: Academic Progress Tracker Program, other children of the APT Program of Study example

This example illustrates the fields and controls on the Academic Progress Tracker Program, other children of the APT Program of Study example. You can find definitions for the fields and controls later on this page.

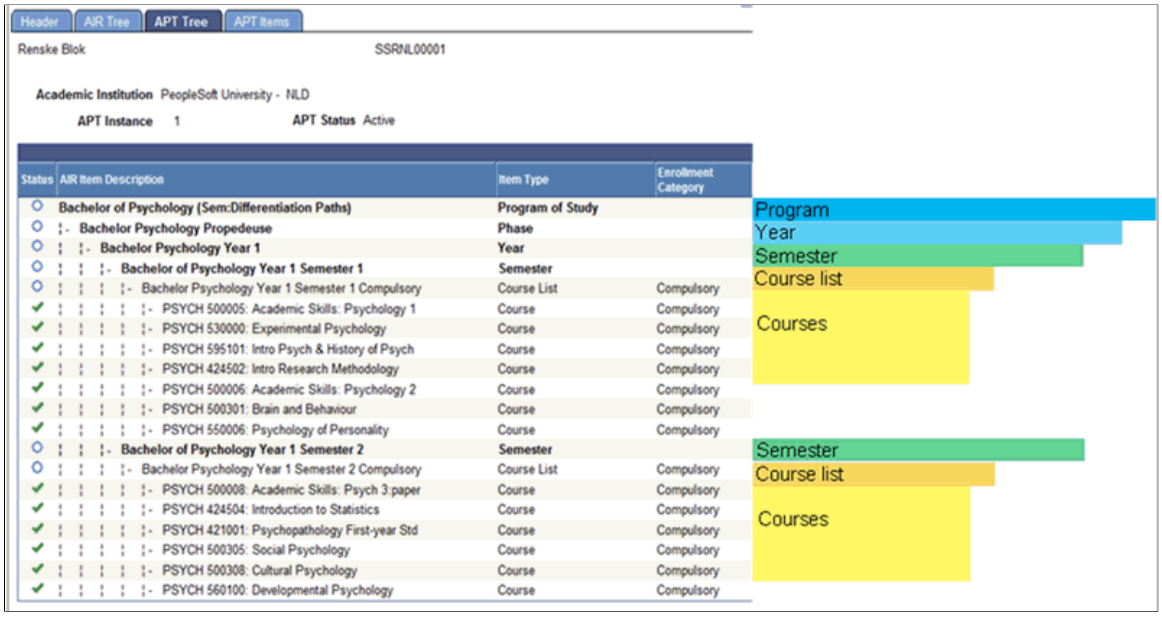


In the example above, the APT Year, APT Study Period, and APT Semester entities have been generated from non-system Academic Item Types Year, Study Period, and Semester. AIR and APT structures are unique in that they can be specified using institution specific objects. The Entity Registry reflects this.

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:

Image: Example of the Academic Progress Tracker Tree

This example illustrates the fields and controls on the Example of the Academic Progress Tracker Tree. You can find definitions for the fields and controls later on this page.



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

Image: Academic Progress Tracker Course Entity Registry Hierarchy example

This example illustrates the fields and controls on the Academic Progress Tracker Course Entity Registry Hierarchy example. You can find definitions for the fields and controls later on this page.

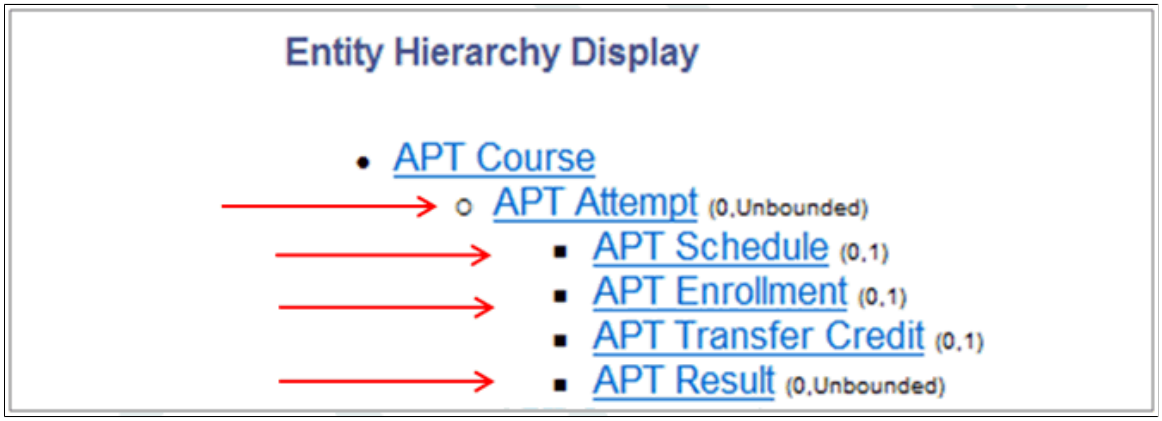
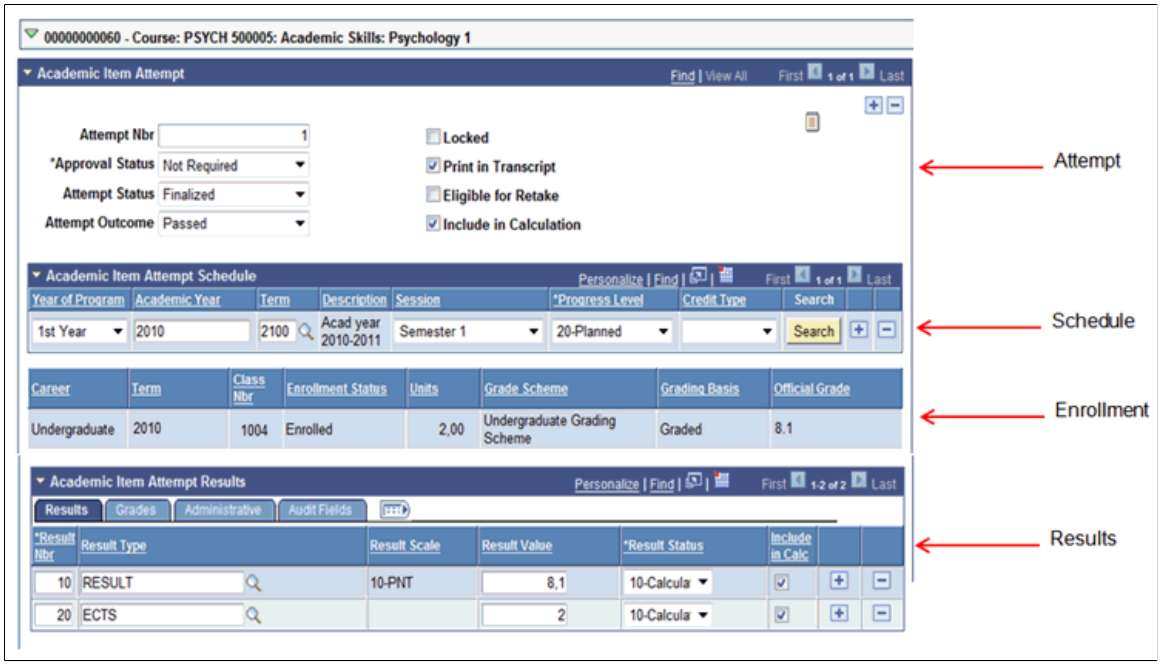


Image: Example of Academic Item Attempt

This example illustrates the fields and controls on the Example of Academic Item Attempt. You can find definitions for the fields and controls later on this page.

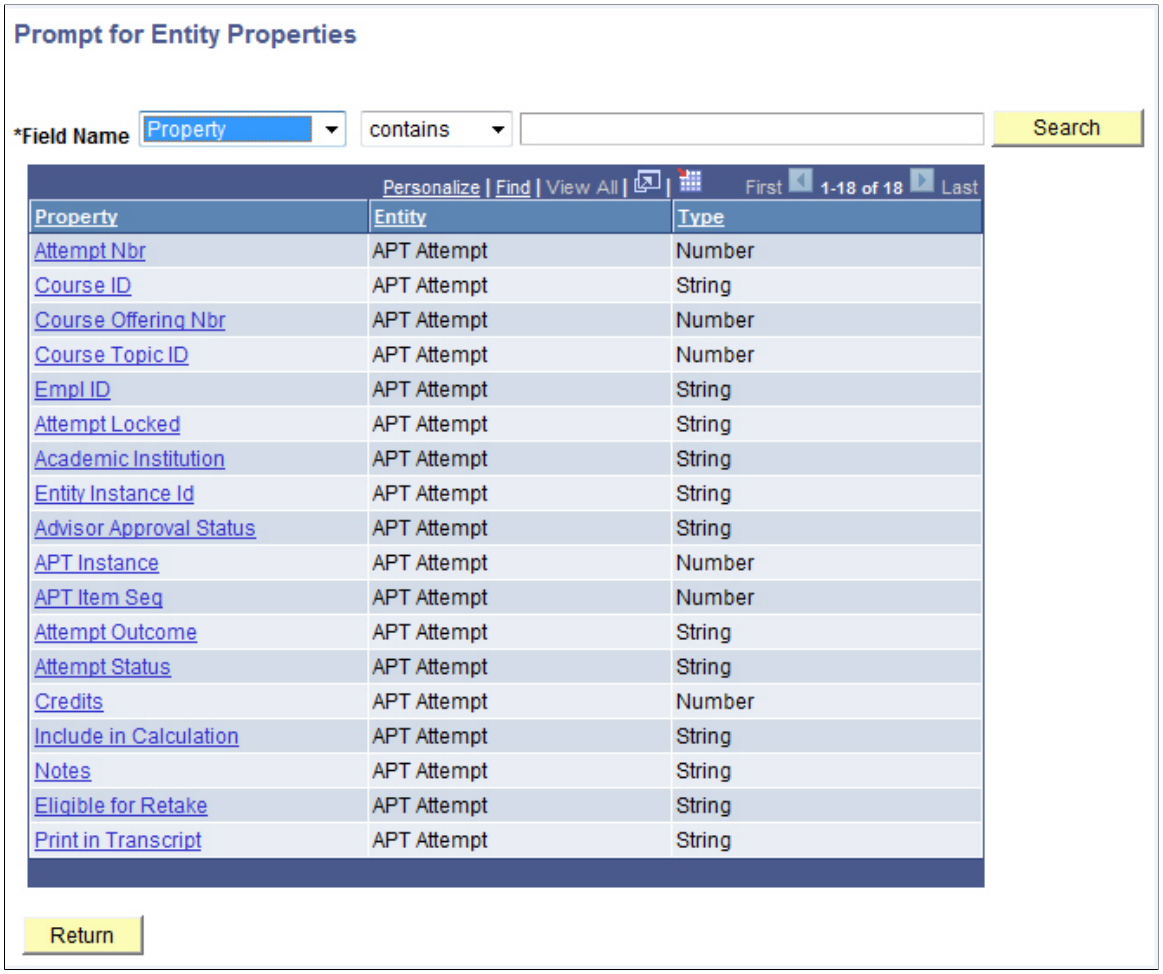


Entity Properties

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

Image: Example of Entity Properties for Academic Item Attempt

This example illustrates the fields and controls on the Example of Entity Properties for Academic Item Attempt. You can find definitions for the fields and controls later on this page.



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.

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.

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
- Rules Engine Batch Processing Component
- Application Component (for example, a user interface)

See Constructing Rules, “Defining Rule Triggers.”

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 for Rules Engine Variables.

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, Install Options, Rules Engine Install Options	Set up Rules Engine Install Options.

Page Name	*Definition Name*	*Navigation*	*Usage*
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.
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.
Rule Category Cross Reference	SCC_RULE_CAT_XREF	Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Cross Reference	Displays Rules, Rule Groups, System Variables, and Triggers associated with the 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, Definition	Define System Variables to use when creating Rules.
System Variables Cross Reference	SCC_RULE_SYSV_XREF	Set up SACR, System Administration, Rules Engine, Setup, Define System Variables, Cross Reference	Displays Rules and Rule Groups associated with the System Variable.
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	Click the Details link on the Define Data Sets page.	Define Data Set property details.

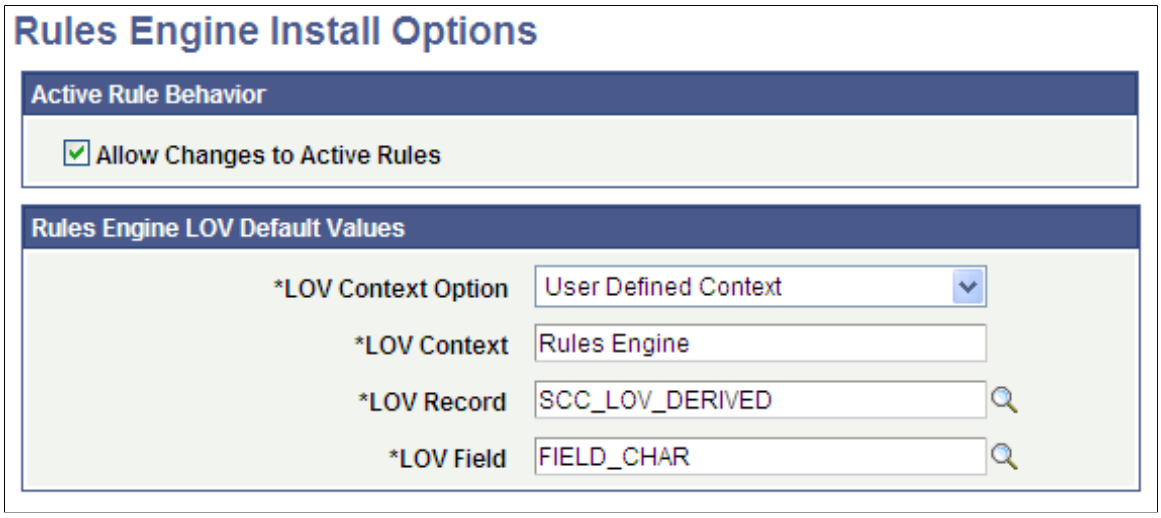
Page Name	Definition Name	Navigation	Usage
Data Set Cross Reference	SCC_RULE_EDS_XREF	Set up SACR, System Administration, Rules Engine, Setup, Define Data Sets, Cross Reference	Displays Rule and Rule Groups associated with the Data Set.
Define List of Values	SCC_RULE_LOV_DEFN	Set up SACR, System Administration, Rules Engine, Setup, Define List of Values	Define lists of values to use for property prompting when creating Rules.
List of Values Cross Reference	SCC_RULE_LOV_XREF	Set up SACR, System Administration, Rules Engine, Setup, Define List of Values, Cross Reference	Displays Rule and Rule Groups associated with the List of Values.

Setting Up Rules Engine Install Options

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

Image: Rules Engine Install Options page

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

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 Setting Up Entity Registry.

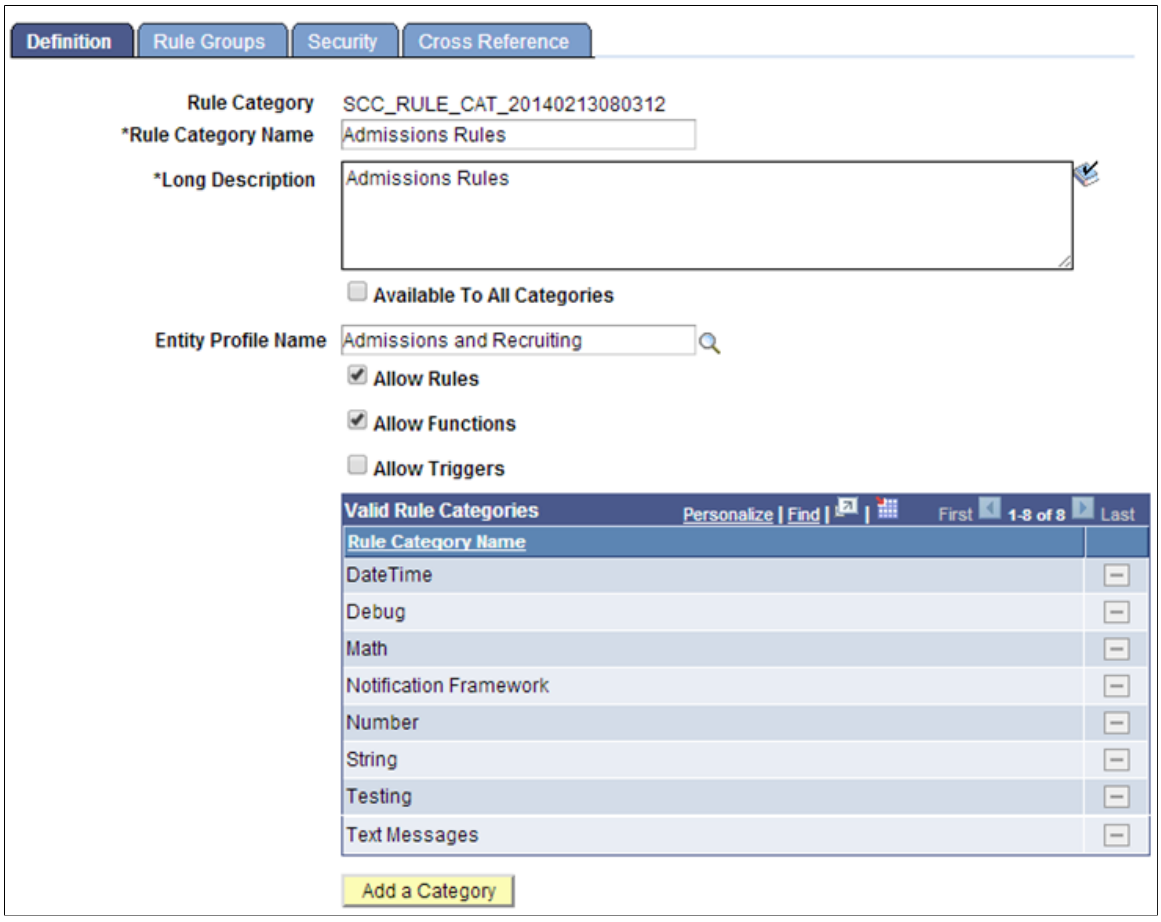
- Valid Rule Groups. Valid Rule Groups can be associated with a Rules Category. Restrictions may be applied so that you can only build Rules from a Rule Group. This can be beneficial in circumstances where you want to guide a group of users with Rule creation by making sure that they build their Rule according to the defaults provided by the Rule Group or Rule Groups associated with this Rule Category.

Note: For Rule Categories delivered with the system, only the Long Description on the Definition page and Rule Groups and Security pages can be modified. Rules Engine Categories delivered with the system are marked by the words “System Data” to the right of the Rule Category field.

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

Image: Rule Category Definition page

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

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.

Available to All Categories Select this check box to make this Category automatically available as a valid Category in other Categories. Categories that are defined as “Available to all Categories” cannot have any Valid Categories of its own. When this check box is deselected, the Category is no longer available in other Categories.

Note: Security settings on the Rule Category affect all Rules, Functions and Triggers which have been called from or created in this Category.

In the example below the Category Math has been made available to Category “Admissions Rules” as a Valid Rule Category. Every Function and Rule Created in Category Math can now be called from a Function or Rule created in Category “Admissions Rules”.

Note: Deselecting “Available to All Categories” on category Math would remove the Category from Admissions Rules and invalidate the security on the Call statement of rules which have already been created. Although this would not break compiled Rules it would cause problems when users want to change those rules.

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 have 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 Category Rule Groups page (Set up SACR, System Administration, Rules Engine, Setup, Define Categories, Rule Groups).

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.

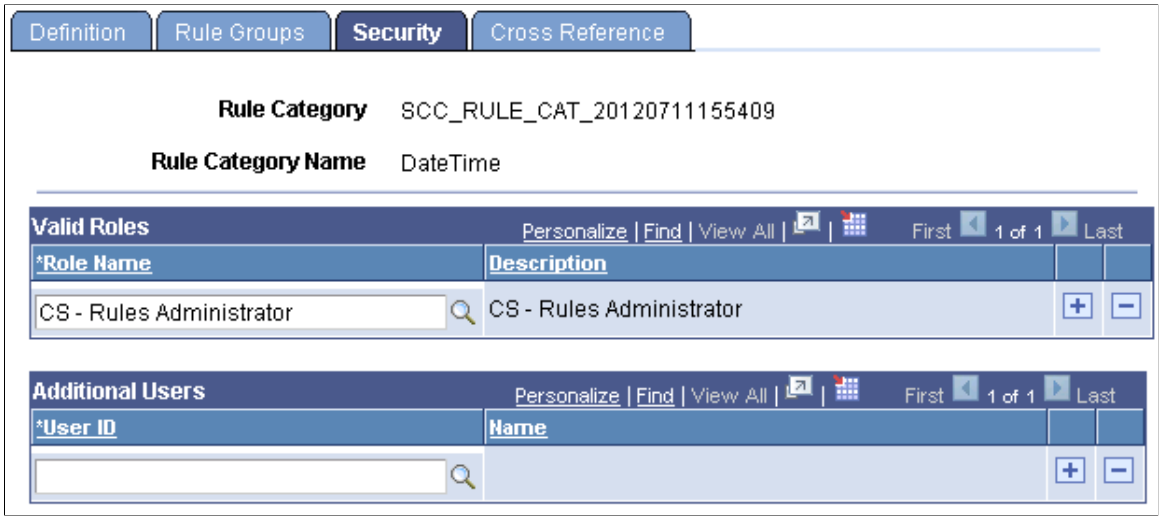
Valid Rule Groups Select Rule Groups.

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

Image: Rule Category Security page

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



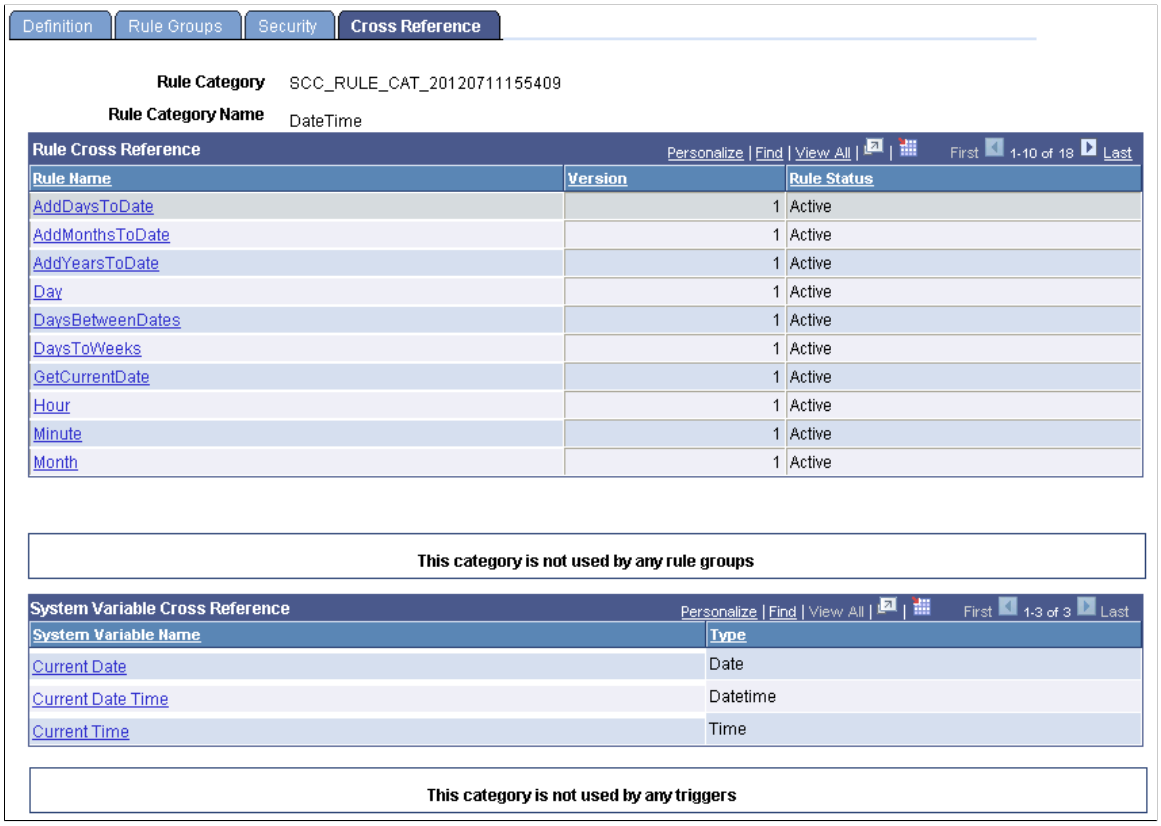
Valid Roles Enter Valid Roles for which the Rules Category definition is valid. These are Roles that are set up using the standard PeopleTools Security function.

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

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

Image: Rule Category Cross Reference page

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



This page shows Rules, Rule Groups, System Variables, and Triggers which have been created as part of this Category.

Rule Name, **Rule Group Name**, **System Variable Name**, and **Trigger Name** Displays to authorized users a link for any of these Rules Engine objects that have been created in this Rule Category. If a user is not authorized, the link is disabled.

Click a link to transfer out of the component and to the referenced object.

Warning! Make sure you have saved any data that you need to before confirming you want to transfer to the referenced object in a new component.

	If no objects are associated with the Rule Category, a notification is shown in place of the object details. for example, "This Category is not used by any triggers".
Version	Displays the Rule Version Number.
Rule Status and **Rule Group Status**	Displays the status of the Rule Group or Rule Name (*In Progress*, *Active*, *In-active*).
Type	Displays the Type of System Variable.

Defining Color Codes for Rules Engine Manager Elements

Note: The assigning of different colors to different Rules Engine Manager Elements for purposes of displaying the Elements in different colors in user interfaces is *optional* and does not affect how a Rule is used in System processing. If no color setup is done, all Element text is displayed in black.

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
Orange	ffa500
Maroon	800000

Note: Colors delivered with the system cannot be modified.

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

Image: Define Colors page

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

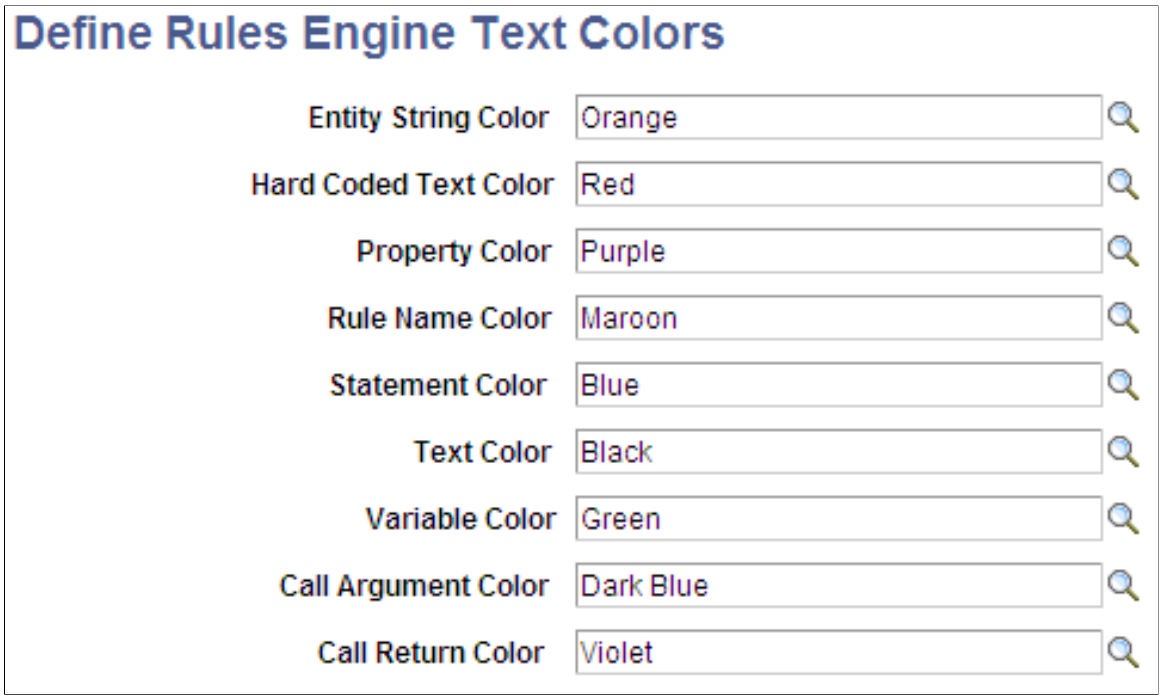
Enter an HTML standard color code. New color codes can be defined using HTML standards.

- 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).

Image: Define Rules Engine Text Colors page

This example illustrates the fields and controls on the Define Rules Engine Text Colors page. You can find definitions for the fields and controls later on this 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 Using Statements for Evaluation and Calculation in a Rule.

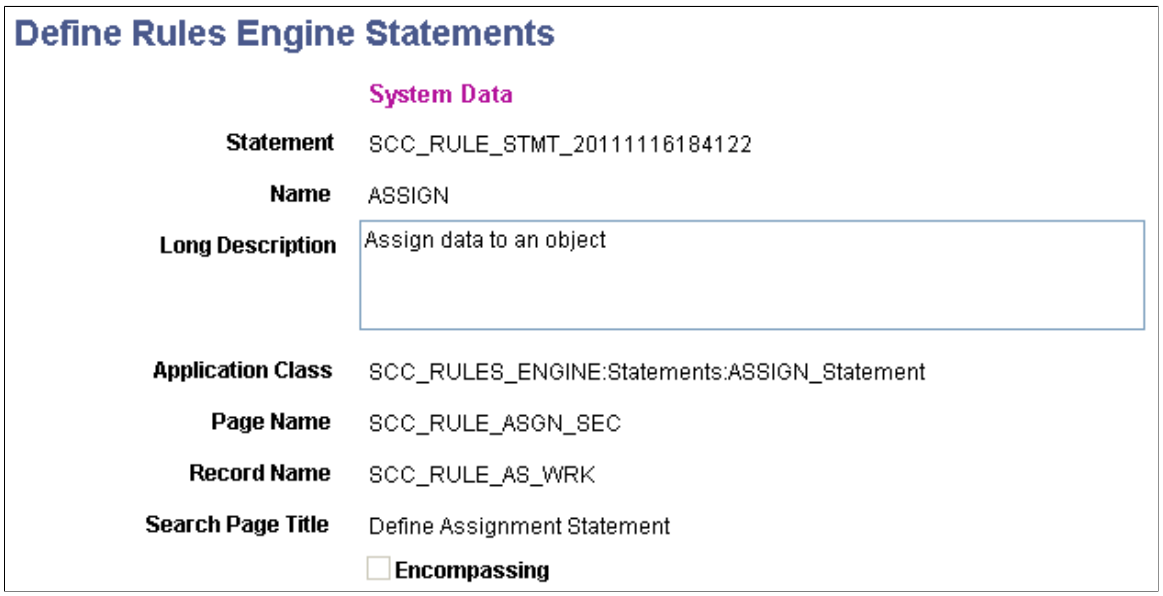
Note: For Statements delivered with the system, only the Long Description can be modified.

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).

Image: Define Rules Engine Statements page

This example illustrates the fields and controls on the Define Rules Engine Statements page. You can find definitions for the fields and controls later on this 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.

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 Using Statements for Evaluation and Calculation in a Rule.

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.

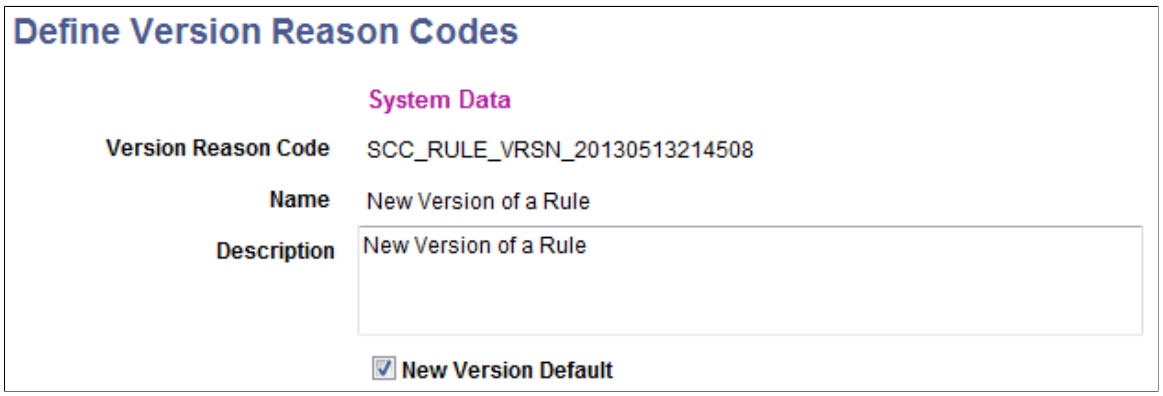
Note: For Version Reason Codes delivered with the system, only the Description can be modified.

Note: The system-delivered Version Reason Code of *New Version of the rule* is used automatically whenever a new version of a rule is created. This Version Reason Code is delivered with the Initial Version Default check box *selected*. If the Initial Version Default check box is *not selected*, the Version Reason Code of *New Version of the rule* can be selected to be both the Initial Version Reason Code as well as the New Version Reason Code.

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

Image: Define Version Reason Codes page

This example illustrates the fields and controls on the Define Version Reason Codes page. You can find definitions for the fields and controls later on this 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..

Initial Version Default — Select this check box to use this Version Reason code automatically when the rule is created. Comments can be added to the Version Reason code before the Version page is saved. The system-delivered Version Reason Code *Initial Version of the Rule* is delivered with this check box selected.

New Version Default — Select this check box to use this Version Reason Code automatically whenever a new version of a rule is created. The system-delivered Version Reason Code *New Version of the rule* is delivered with this check box selected.

If the Initial Version Default check box for the system-delivered Version Reason Code *Initial Version of the Rule* is *not selected and* the system-delivered Version Reason Code *New Version of*

the rule is *selected*, the New Version Default reason is used for both.

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

Student ID	Student Name	Course List ID	Total Units
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 *Adding Variables to a Rule*.

Defining Rules Engine System Variables

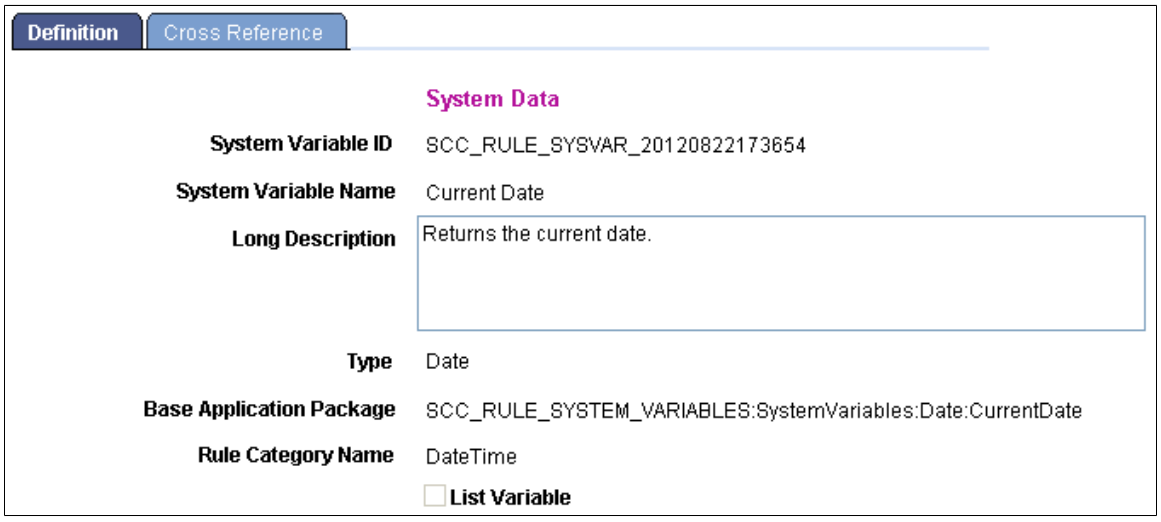
A System Variable is predefined in the system typically by a developer or programmer. Once created, the System Variable can be used in Rules built by functional experts. System Variables can be created to provide re-usable defaults for commonly used values such as system date, Operator ID, etc.

Note: Rules Engine System Variables delivered with the system cannot be modified.

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

Image: Define System Variables page

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

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.

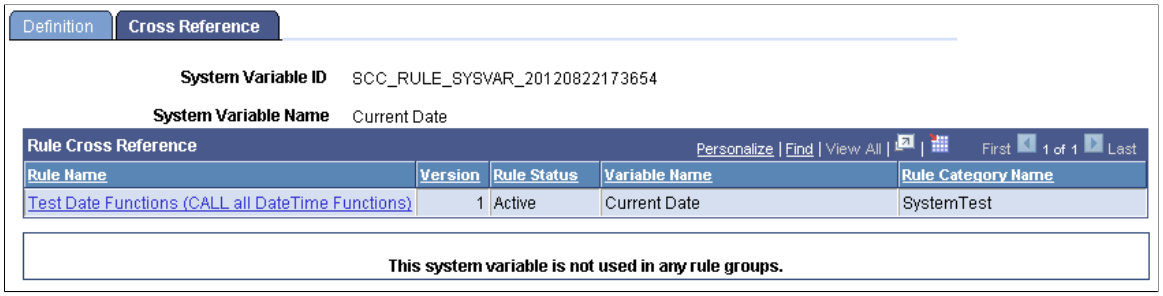
List Variable Select this check box if the System Variable needs to accommodate the return of multiple values.

For a complete listing of System Variables delivered with the system for use in Rules and Functions by an Expert user, see Library of System-Delivered Rules Engine Objects

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

Image: Define System Variables Cross Reference page

This example illustrates the fields and controls on the Define System Variables Cross Reference page. You can find definitions for the fields and controls later on this page.



This page shows Rules and Rule Groups associated with this System Variable.

Rule Name and **Rule Group Name**

Displays to authorized users a link for any of these Rules Engine objects which use this System Variable. If a user is not authorized, the link is disabled.

Click a link to transfer out of the component and to the referenced object.

Warning! Make sure you have saved any data that you need to before confirming you want to transfer to the referenced object in a new component.

If no objects use the System Variable, a notification is shown in place of the object details. for example, “This System Variable is not used by any Rule Groups”.

Version

Displays the Rule Version Number.

Rule Status and **Rule Group Status**

Displays the status of the Rule Group or Rule Name (*In Progress*, *Active*, *In-active*).

Variable Name

Displays the name of the Variable associated with the System Variable.

Rule Category Name

Displays the Rule Category Name in which the Rule or Rule Group has been created.

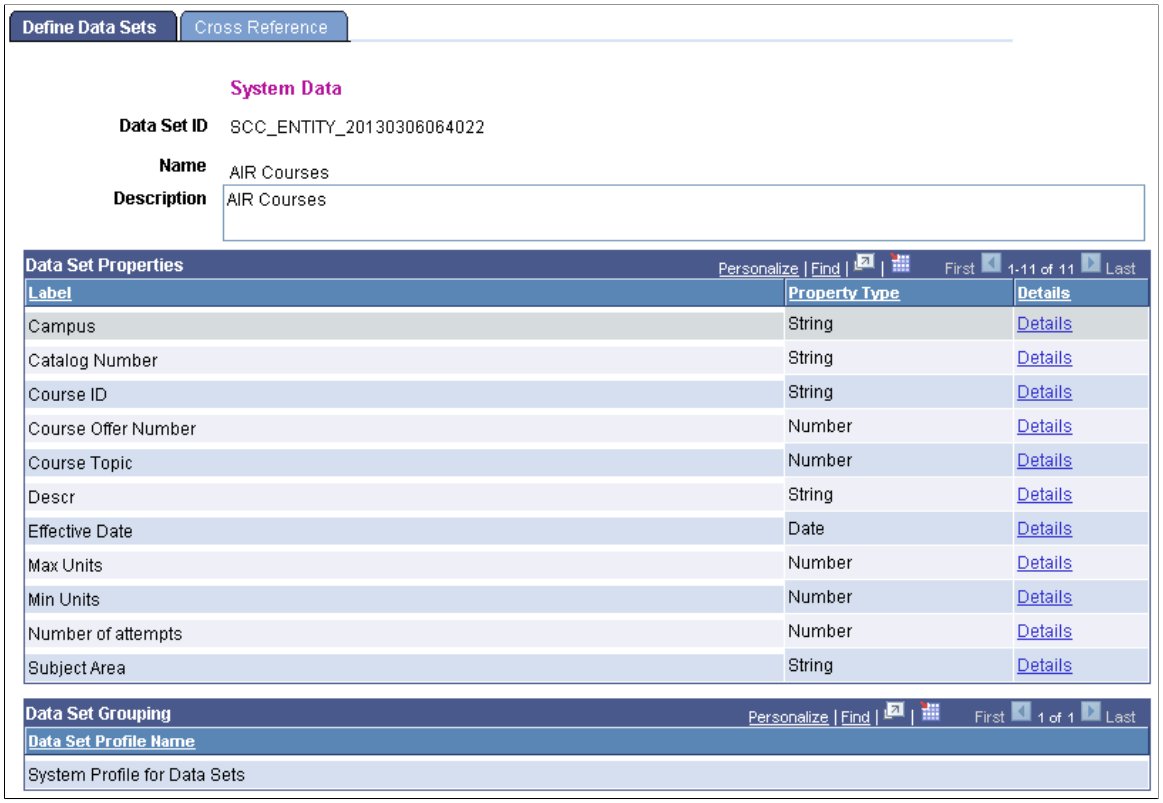
Defining Rules Engine Data Sets

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 Sets).

Image: Define Data Sets page

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

Actions

- *Clone Data Set* – Select this option to copy from an existing Data Set. A prompt is presented from which a Data Set can be selected.
- *Copy Properties from* – Select this option to copy from an Entity. A prompt is presented from which an Entity can be selected. A Data Set Property is created for each Property which exists on the selected Entity.

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:

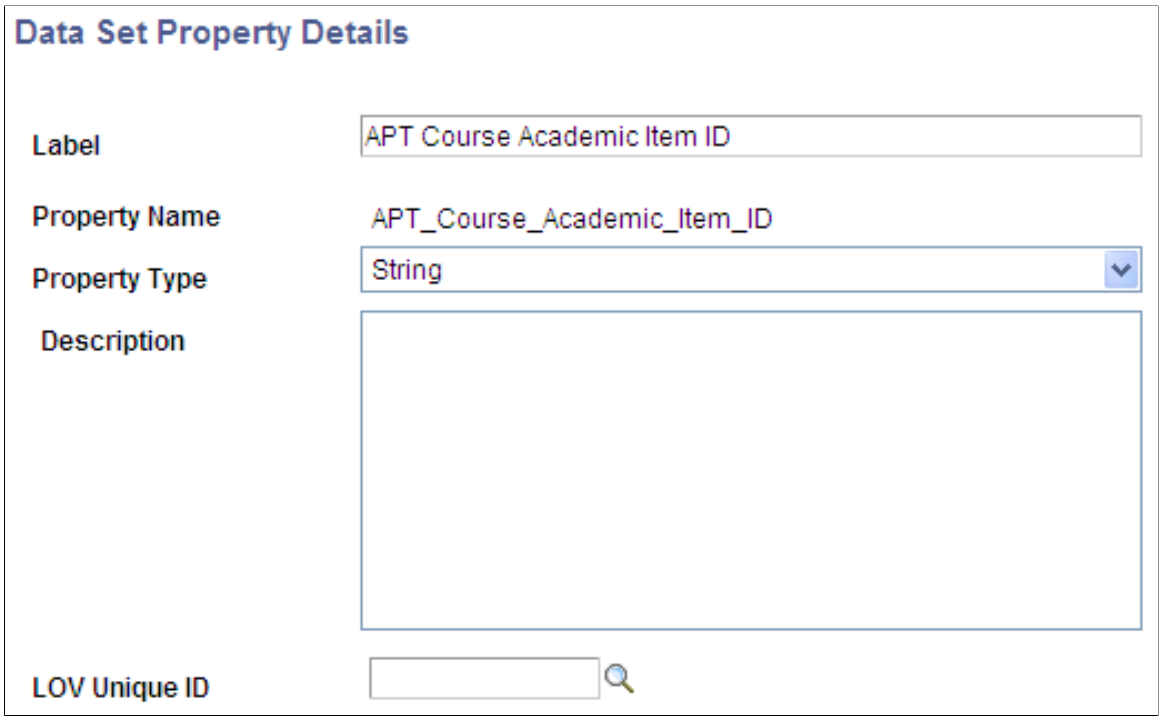
- Date
- DateTime
- Number
- String
- Time

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

Access the Data Sets Details page (click the Details link on the Define Data Sets page).

Image: Data Set Property Details page

This example illustrates the fields and controls on the Data Set Property Details page. You can find definitions for the fields and controls later on this 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 *Defining Lists of Values for Rules Engine Variables*.

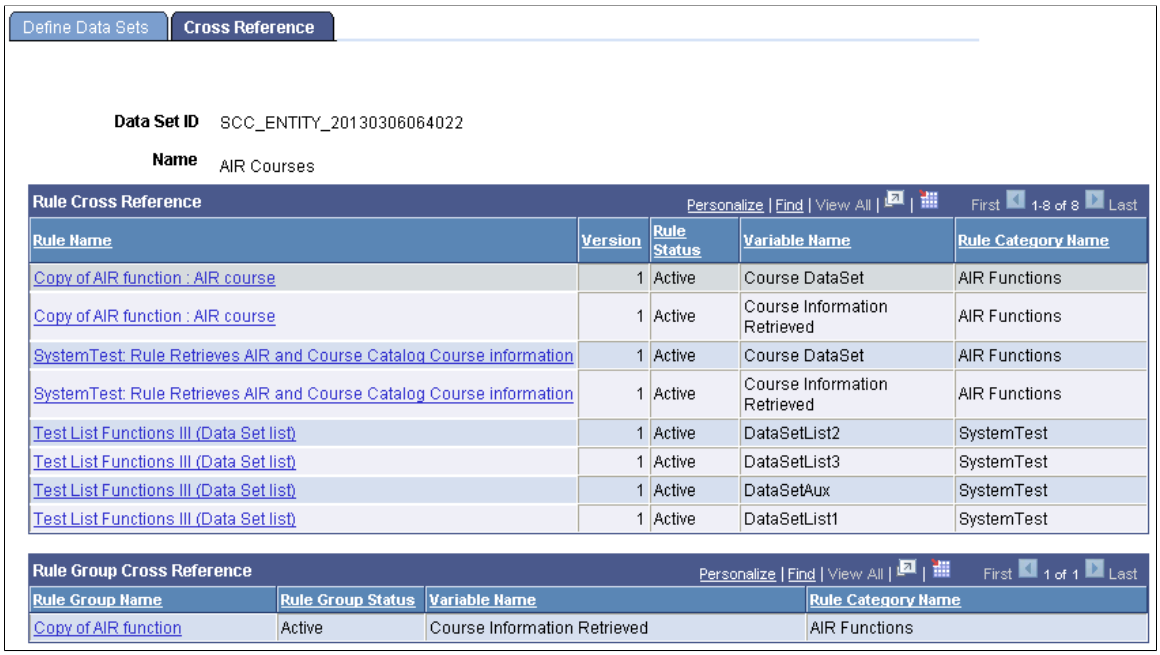
Data Set Profile Name Select Entity Profiles to associate multiple Data Sets to a group of Data Sets. The Category is used to search for Data Sets when associating Rules Engine Variables with created Data Sets.

There is no security tied into this functionality. A created Data Set can be used by any Rule in the system.

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

Image: Define Data Sets Cross Reference page

This example illustrates the fields and controls on the Define Data Sets Cross Reference page. You can find definitions for the fields and controls later on this page.



Define Data Sets | Cross Reference

Data Set ID SCC_ENTITY_20130306064022

Name AIR Courses

Rule Cross Reference Personalize | Find | View All | First 1-8 of 8 Last

Rule Name	Version	Rule Status	Variable Name	Rule Category Name
Copy of AIR function : AIR course	1	Active	Course DataSet	AIR Functions
Copy of AIR function : AIR course	1	Active	Course Information Retrieved	AIR Functions
SystemTest: Rule Retrieves AIR and Course Catalog Course information	1	Active	Course DataSet	AIR Functions
SystemTest: Rule Retrieves AIR and Course Catalog Course information	1	Active	Course Information Retrieved	AIR Functions
Test List Functions III (Data Set list)	1	Active	DataSetList2	SystemTest
Test List Functions III (Data Set list)	1	Active	DataSetList3	SystemTest
Test List Functions III (Data Set list)	1	Active	DataSetAux	SystemTest
Test List Functions III (Data Set list)	1	Active	DataSetList1	SystemTest

Rule Group Cross Reference Personalize | Find | View All | First 1 of 1 Last

Rule Group Name	Rule Group Status	Variable Name	Rule Category Name
Copy of AIR function	Active	Course Information Retrieved	AIR Functions

This page shows Rules and Rule Groups associated with this System Variable.

Rule Name and **Rule Group Name** Displays to authorized users a link for either of these Rules Engine objects which use this Data Set. If a user is not authorized, the link is disabled.

Click a link to transfer out of the component and to the referenced object.

Warning! Make sure you have saved any data that you need to before confirming you want to transfer to the referenced object in a new component.

If no objects use the Data Set, a notification is shown in place of the object details. for example, “This Data Set is not used by any Rule Groups”.

Version Displays the Rule Version Number.

Rule Status and **Rule Group Status** Displays the status of the Rule Group or Rule Name (*In Progress*, *Active*, *In-active*).

Variable Name	Displays the name of the Variable associated with the System Variable.
Rule Category Name	Displays the Rule Category Name in which the Rule or Rule Group has been created.

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

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, Entity Property, or DataSet Property which has the same Type (for example, String) as the defined LOV.

LOV Prompts may already exist on an Entity Registry Property. The Entity Registry automatically adds a LOV prompt to a property for which the underlying record field contains a Prompt Table. Adding an LOV to a property which already has an automatic prompt table defined overwrites that functionality.

For the Lists of Values are delivered with the system for the Rules Engine, see Library of System-Delivered Rules Engine Objects.

Note: Lists of Values delivered with the system cannot be modified, but the Test button can be used to test return of valid LOV values.

To understand more about how to add an LOV to an Entity Property, see Setting Up Entity Registry, “Setting Up Entity Property Details.”

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

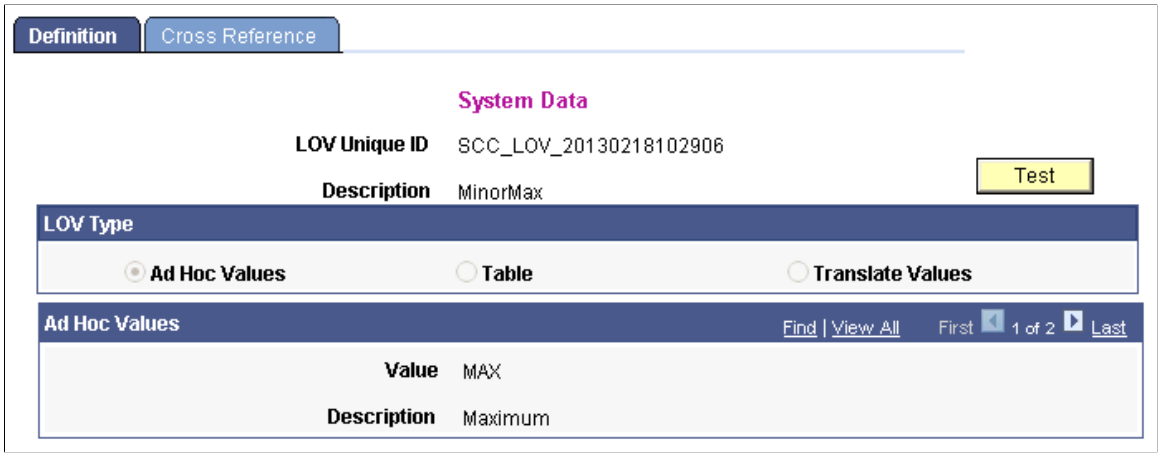
To understand more about how LOV functionality can be used throughout the system, see Setting Up List of Values.

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).

Image: Define List of Values – Ad Hoc Values page

This example illustrates the fields and controls on the Define List of Values – Ad Hoc Values page. You can find definitions for the fields and controls later on this 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.

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.

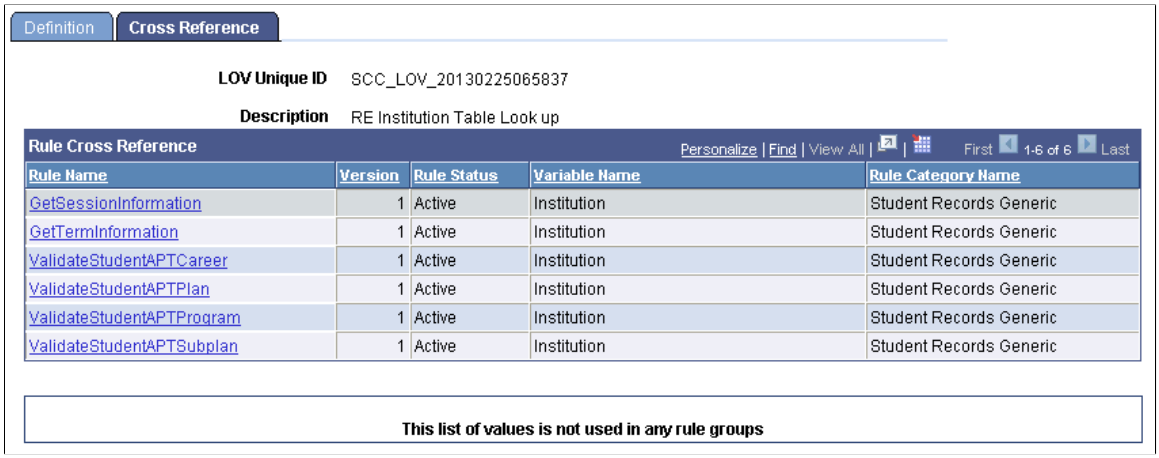
Test — Click the Test button to display a preview of the values that will be returned by the LOV when it is deployed as a prompt edit.

Defining Lists of Values – Table

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

Image: Define List of Values – Table

This example illustrates the fields and controls on the Define List of Values – Table. You can find definitions for the fields and controls later on this page.



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. 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 are 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.

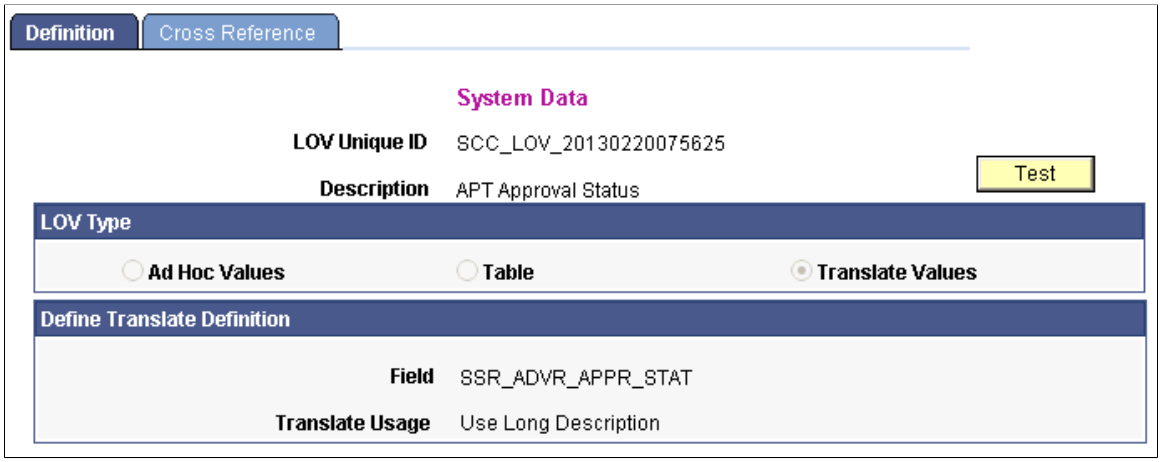
Defining Lists of Values – Translate Values

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

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

Image: Define List of Values – Translate Values

This example illustrates the fields and controls on the Define List of Values – Translate Values. You can find definitions for the fields and controls later on this page.



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.

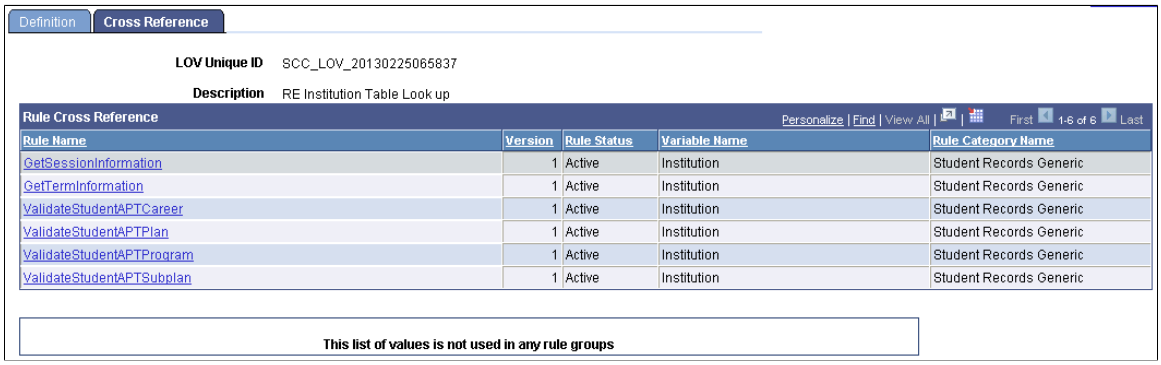
- Remove All Values – Click button to remove all LOV values from the excluded values list.

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.

Access the Define List of Values Cross Reference page (Set up SACR, System Administration, Rules Engine, Setup, Define List of Values, Cross Reference).

Image: Define List of Values Cross Reference

This example illustrates the fields and controls on the Define List of Values Cross Reference. You can find definitions for the fields and controls later on this page.



This page shows Rules and Rule Groups associated with this System Variable.

Rule Name and **Rule Group Name** Displays to authorized users a link for either of these Rules Engine objects which uses a variable with this List of Values. If a user is not authorized, the link is disabled.

Click a link to transfer out of the component and to the referenced object.

Warning! Make sure you have saved any data that you need to before confirming you want to transfer to the referenced object in a new component.

If no objects use the List of Values, a notification is shown in place of the object details. for example, “This List of Values is not used by any Rule Groups”.

Version Displays the Rule Version Number.

Rule Status and **Rule Group Status** Displays the status of the Rule Group or Rule Name (*In Progress*, *Active*, *In-active*).

Variable Name Displays the name of the Variable that uses this List of Values.

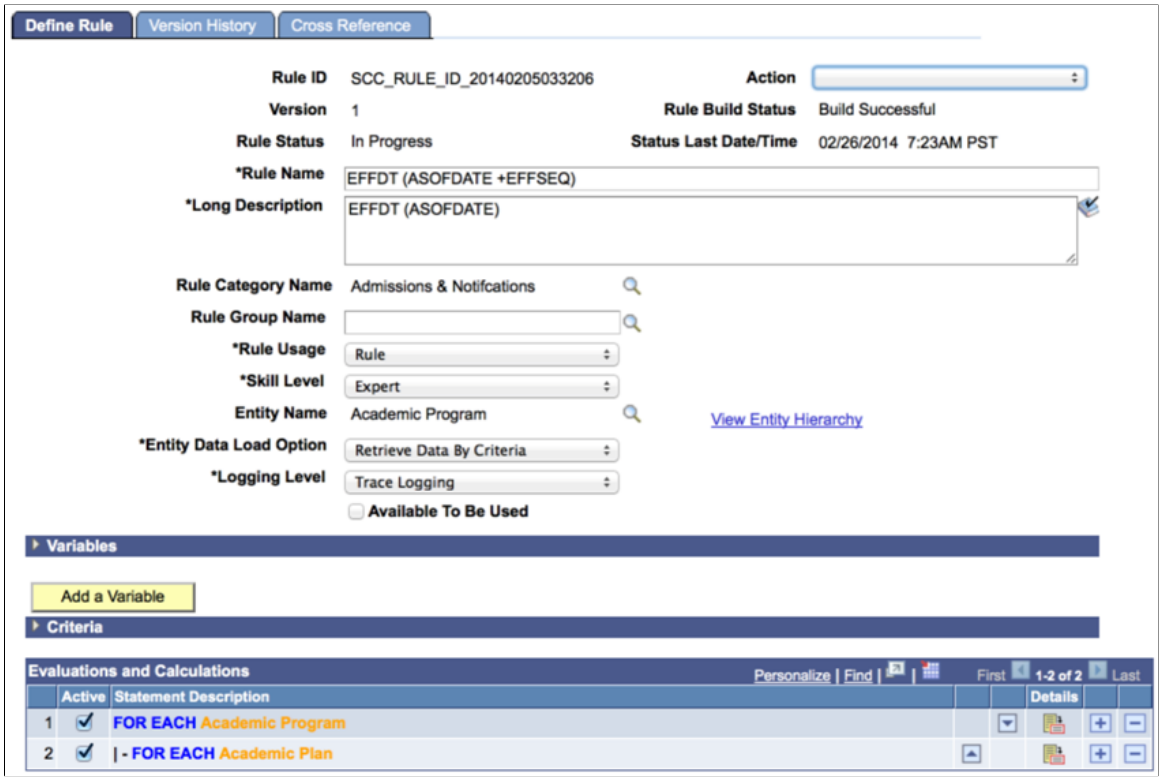
Rule Category Name Displays the Rule Category Name in which the Rule or Rule Group has been created.

Example of an LOV Added to a Property from Record Field Prompts

The following is an example of an Entity Registry LOV which has been added to the Property due to the fact that the underlying Record field was defined with a prompt table value. In this Rule, the Entity Academic Program is used to retrieve applicants.

Image: Example of Define Rule Page Using a LOV from Field Record Prompts

This example illustrates the Define Rule page using a LOV from Field Record Prompts.

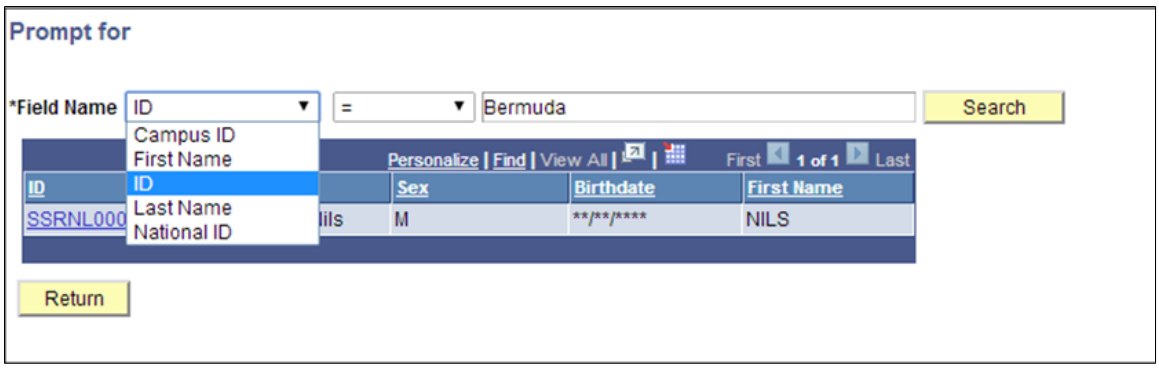


In this example in the Criteria grid, the Type of the Property with theLabel of *EmplID* is set to *Text*. Click the Search prompt button to the right of the Object (value) for Property EMPLID to open a Lookup table. This is the PEOPLE_SRCH Prompt for table. This functionality also works whenever a Property with a

Lookup table is used in combination with the Assign or IF Evaluative statements. The same Search values are available and, if relevant, permission level security is enforced:

Image: Prompt Window for Example Of Define Rule Page Using A LOV From Field Record Prompts

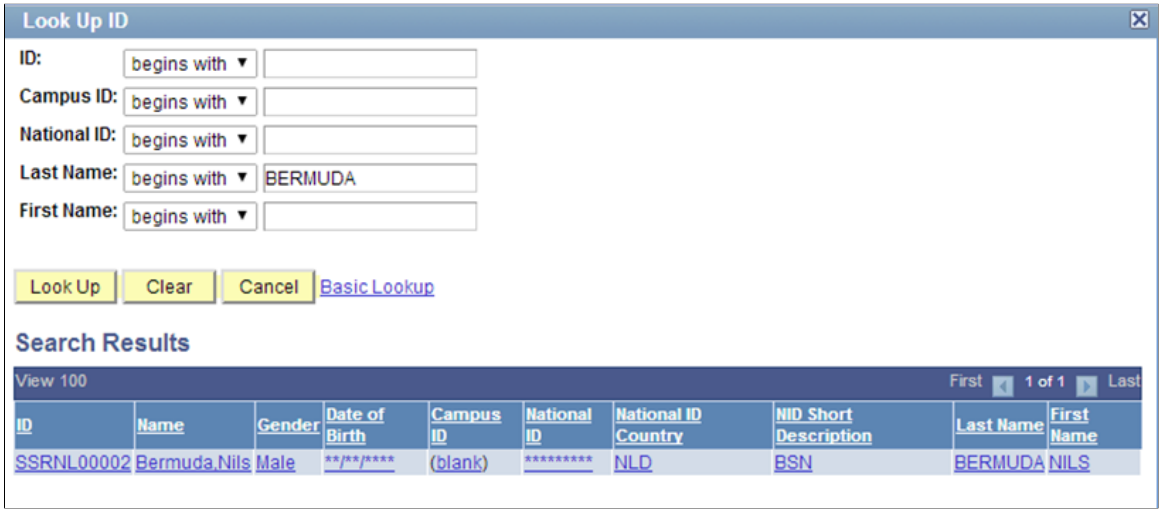
This example illustrates the Prompt Window for Example Of Define Rule Page Using A LOV From Field Record Prompts.



The Search function behaves the same as if prompted from a normal search record or component:

Image: Example of Prompt Options from Component Search Record (Student Admissions, Application Maintenance, Maintain Applications)

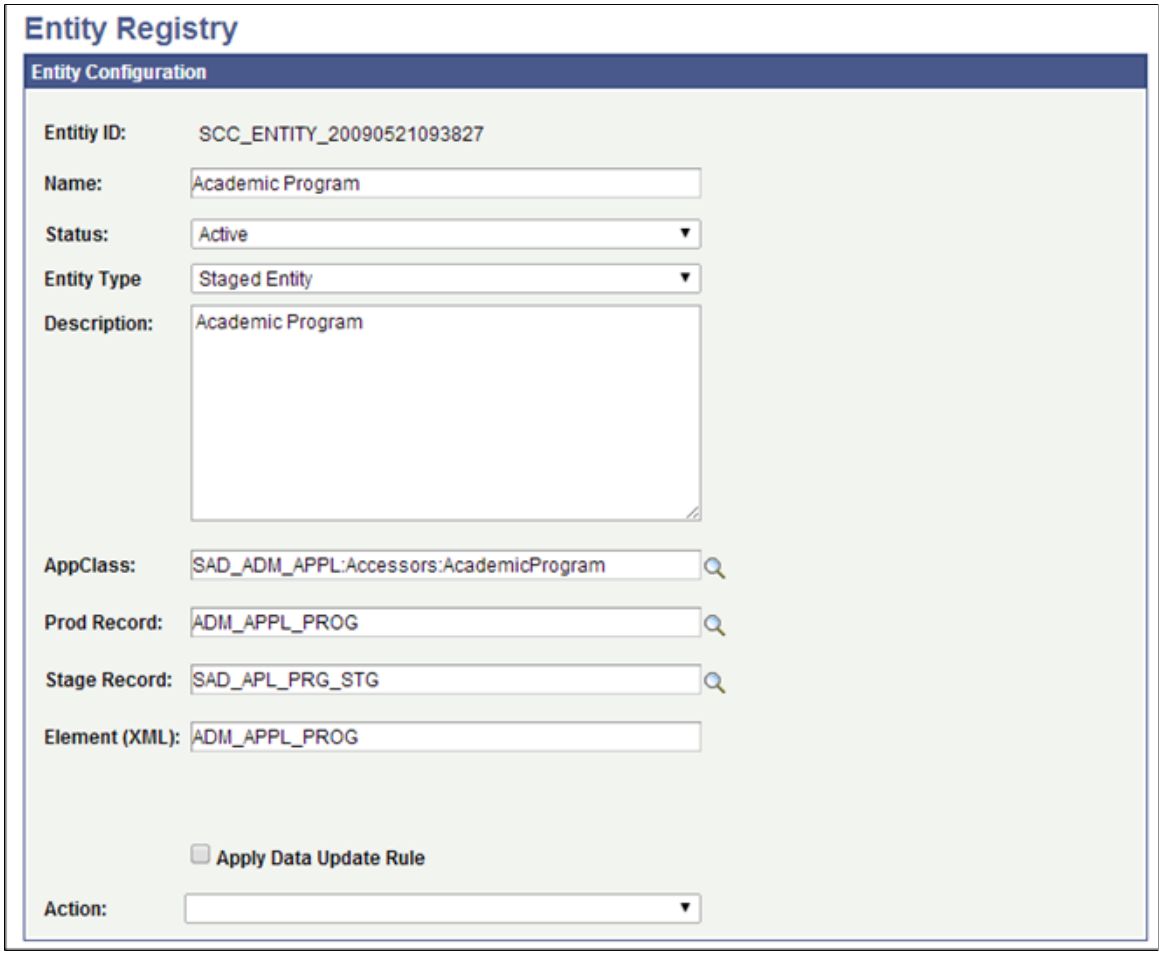
This example illustrates Prompt Options from Component Search Record (Student Admissions, Application Maintenance, Maintain Applications).



The Lookup functionality for EmplID Property is enabled by the Entity Registry. To view the Entity Prompts, access the Entity Registry page (Set Up SACR, System Administration, Entity, Entity Registry).

Image: Entity Registry Window for Example of Define Rule Page Using a LOV from Field Record Prompts

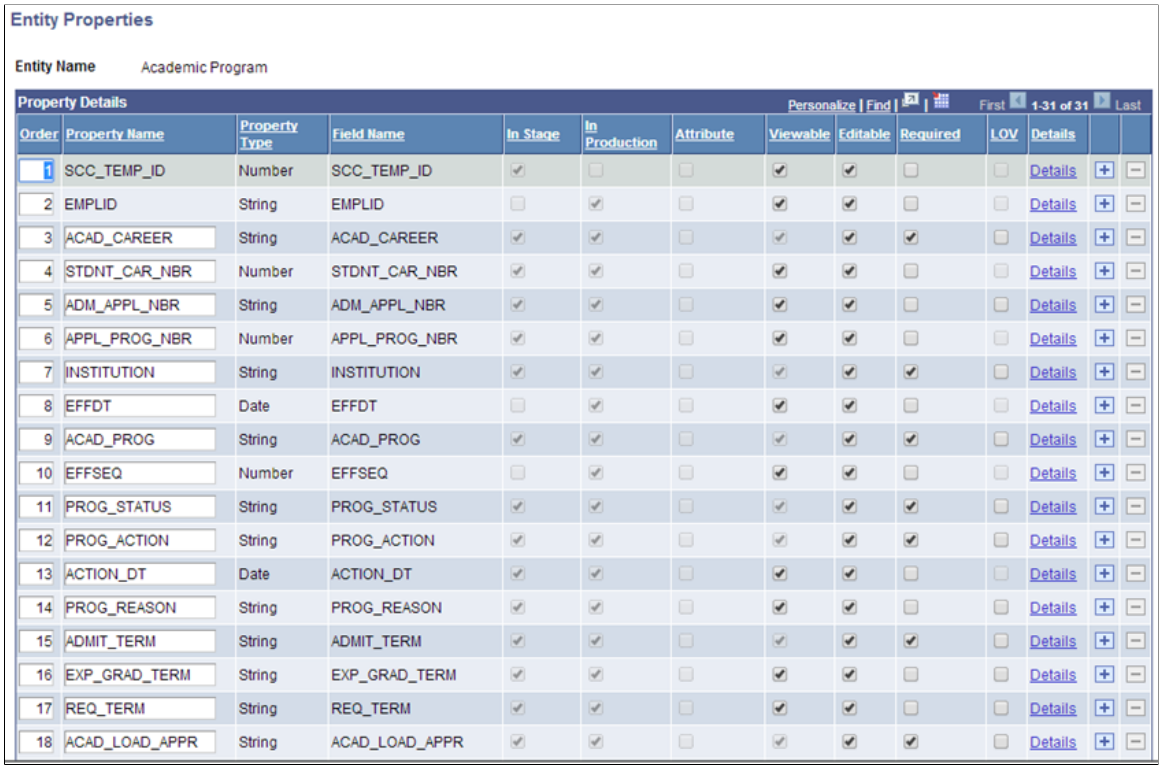
This example illustrates the Entity Registry Window for Example of Define Rule Page Using a LOV from Field Record Prompts



Select *Edit Properties* from the Action drop-down to view the Entity properties:

Image: Entity Properties Window for Example Of Define Rule Page Using a LOV from Field Record Prompts

This example illustrates Entity Properties Window for Example Of Define Rule Page Using a LOV from Field Record Prompts.



Entity Properties

Entity Name: Academic Program

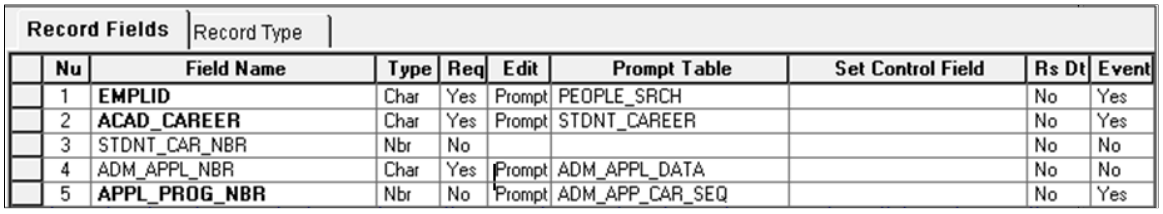
Property Details — Personalize | Find | First 1-31 of 31 Last

Order	Property Name	Property Type	Field Name	In Stage	In Production	Attribute	Viewable	Editable	Required	LOV	Details
1	SCC_TEMP_ID	Number	SCC_TEMP_ID	☑	☐	☐	☑	☑	☐	☐	Details
2	EMPLID	String	EMPLID	☐	☑	☐	☑	☑	☐	☐	Details
3	ACAD_CAREER	String	ACAD_CAREER	☑	☑	☐	☑	☑	☑	☐	Details
4	STDNT_CAR_NBR	Number	STDNT_CAR_NBR	☑	☑	☐	☑	☑	☐	☐	Details
5	ADM_APPL_NBR	String	ADM_APPL_NBR	☑	☑	☐	☑	☑	☐	☐	Details
6	APPL_PROG_NBR	Number	APPL_PROG_NBR	☑	☑	☐	☑	☑	☐	☐	Details
7	INSTITUTION	String	INSTITUTION	☑	☑	☐	☑	☑	☑	☐	Details
8	EFFDT	Date	EFFDT	☐	☑	☐	☑	☑	☐	☐	Details
9	ACAD_PROG	String	ACAD_PROG	☑	☑	☐	☑	☑	☑	☐	Details
10	EFFSEQ	Number	EFFSEQ	☐	☑	☐	☑	☑	☐	☐	Details
11	PROG_STATUS	String	PROG_STATUS	☑	☑	☐	☑	☑	☑	☐	Details
12	PROG_ACTION	String	PROG_ACTION	☑	☑	☐	☑	☑	☑	☐	Details
13	ACTION_DT	Date	ACTION_DT	☑	☑	☐	☑	☑	☐	☐	Details
14	PROG_REASON	String	PROG_REASON	☑	☑	☐	☑	☑	☐	☐	Details
15	ADMIT_TERM	String	ADMIT_TERM	☑	☑	☐	☑	☑	☑	☐	Details
16	EXP_GRAD_TERM	String	EXP_GRAD_TERM	☑	☑	☐	☑	☑	☐	☐	Details
17	REQ_TERM	String	REQ_TERM	☑	☑	☐	☑	☑	☐	☐	Details
18	ACAD_LOAD_APPR	String	ACAD_LOAD_APPR	☑	☑	☐	☑	☑	☑	☐	Details

Prompt values present in a production record or staging record of an Entity are added to the Entity Registry automatically. Any Property which has inherited a Prompt from a production or staging record is displayed as view-only in the LOV column of the Entity Properties page. In this example the field EMPLID has inherited a prompt to the PEOPLE_SRCH table from the ADM_APPL_PROG production record:

Image: Property and Prompt Table for Example of Define Rule Page Using a LOV from Field Record Prompts

This illustrates the Property and Prompt Table for Example of Define Rule Page Using a LOV from Field Record Prompts



Record Fields | Record Type

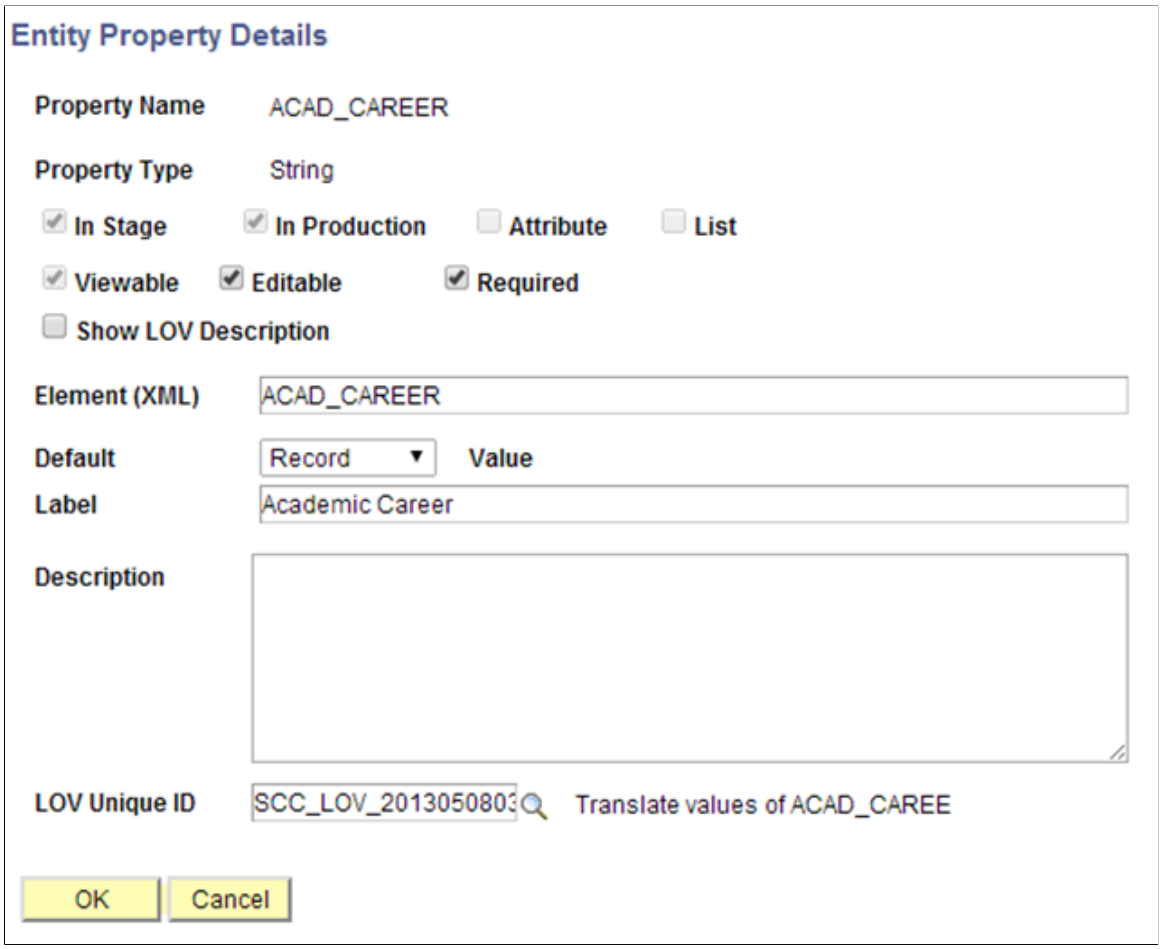
Nu	Field Name	Type	Req	Edit	Prompt Table	Set Control Field	Rs Dt	Event
1	**EMPLID**	Char	Yes	Prompt	PEOPLE_SRCH		No	Yes
2	**ACAD_CAREER**	Char	Yes	Prompt	STDNT_CAREER		No	Yes
3	STDNT_CAR_NBR	Nbr	No				No	No
4	ADM_APPL_NBR	Char	Yes	Prompt	ADM_APPL_DATA		No	No
5	**APPL_PROG_NBR**	Nbr	No	Prompt	ADM_APP_CAR_SEQ		No	Yes

When the Academic Program Entity is used to build Rules in the Rules Engine Manager, the prompt to PEOPLE_SRCH becomes active. These prompts cannot be removed; however, they can be overridden by adding a new LOV Lookup Table value to the Property.

For an example of a LOV Look Up Table that is NOT present on the production or staging record and has been added to an Entity Property, select the Details link for property ACAD_CAREER in the Entity Properties page example. This opens the Entity Properties Detail page where an LOV Lookup Table can be added using the LOV Unique ID field for a relevant LOV:

Image: Entity Property Details Window for Example of Define Rule Page Using a LOV from Field Record Prompts

This illustrates the Entity Property Details Window for Example of Define Rule Page Using a LOV from Field Record Prompts.



Note: The Rules Engine automatically enforces the security used by security views by the OPRID or OPRCLASS of the user who built the rule to the security view.

Note: %EDITABLE prompts do not enable look-up functionality in the Rules Engine.

Note: Performance should be considered when using prompts in the Rules Engine. Do not use look-up tables in the Rules Engine user interface without providing at least one relevant field value.

Constructing Rules

This section discusses how to:

- Use Rules Engine Search.
- Use Rule Engine Groups Search.
- Create Functional Rules.
- Add Variables to a Rule.
- Add Criteria to a Rule.
- Define Rule Groups.
- Define Rule Triggers.

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*.

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 documentation. 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.
Rule Group Search	SCC_RULEGRSRCH_SEC	Set up SACR, System Administration, Rules Engine, Define Rule Groups, select the Search for a Rule 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 the Search button, select a Rule from the results and click Next. 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. 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 the Search button, select a Rule Group from the results, and click Next.	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.

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

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. 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 the Search button, and select a Rule from the results. 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 the Search button, and select a Rule Group from the results.	Define Rule Groups.

Page Name	*Definition Name*	*Navigation*	*Usage*
Rule Groups Cross Reference	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 the Search button, select a Rule Group from results, click the Cross Reference tab. Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select the Add a New Rule Group radio button, click Next, and click the Cross Reference 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, and click the Cross Reference 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 the Cross Reference tab.	View Rules associated with the Rule Group.
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).

Image: Rules Engine Manager Search page

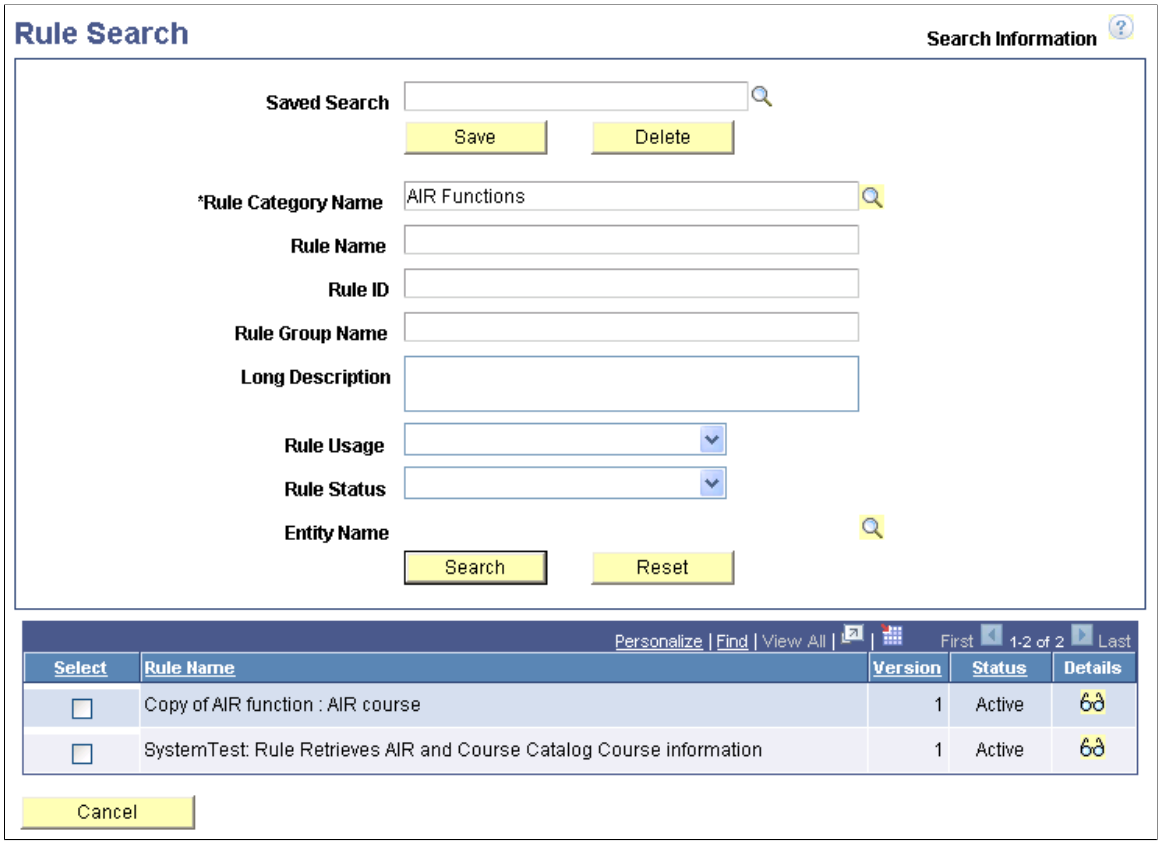
This example illustrates the fields and controls on the Rules Engine Manager Search page. You can find definitions for the fields and controls later on this page.

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.

Image: Rule Search Options page

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

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

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 Constructing Rules.

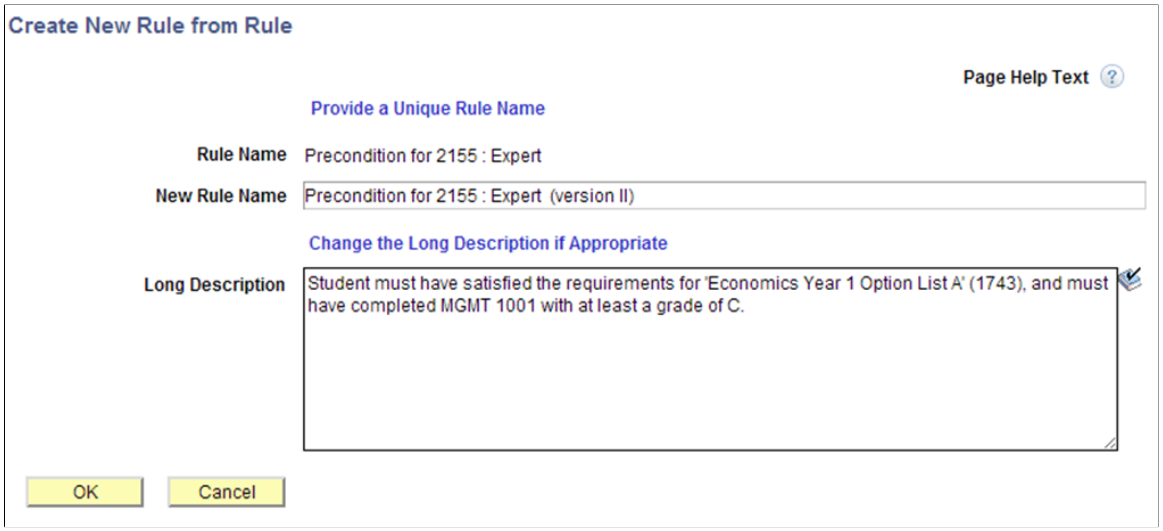
Creating a New Rule from an Existing Rule

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 Create New Rule from Rule page and provide a New Rule Name and/or Long Description.

Image: Create New Rule From Rule page

This illustrates the Create New Rule From Rule page.



A new Rule is created with Rule Status *In Progress* , and it inherits all the Variables, Criteria, and evaluative statements from the Rule you selected to clone

New Rule Name	Enter a name for the new Rule. The New Rule Name must be different from the original Rule Name.
Long Description	The Long Description of the original Rule populates this field and can be edited for the new Rule.

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.

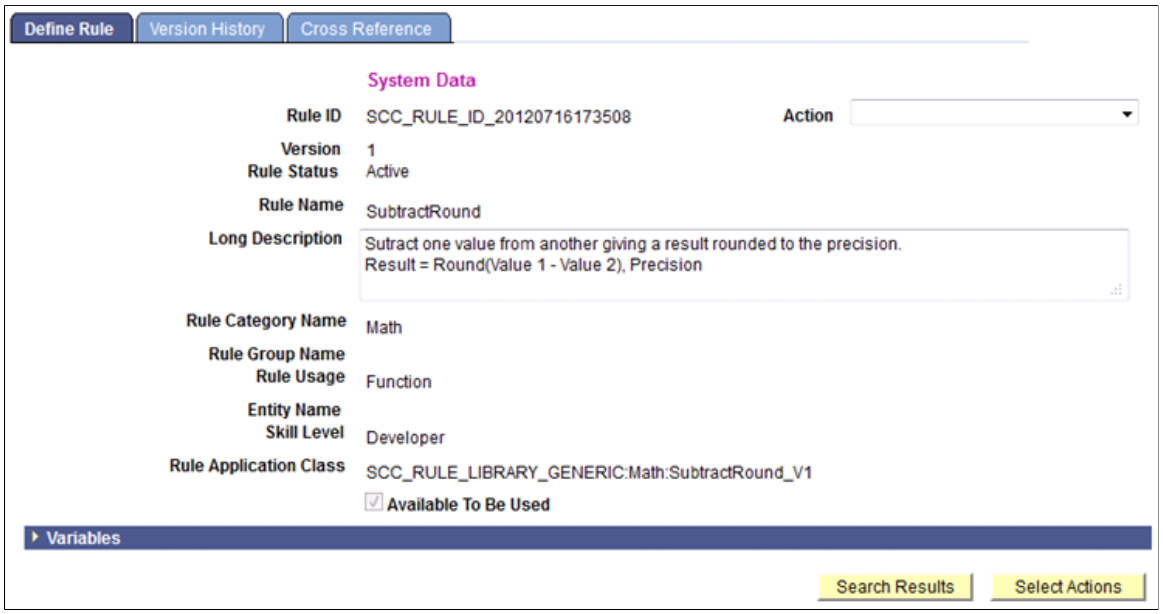
When you have found the Rule Group to which you want to add a Rule, click Next to open the Create a New Rule from Rule Group page and provide a New Rule Name and/or Long Description. A new Rule is created with Rule Status *In Progress*, and it has inherited the Input and Output Variables, Entity Profile, 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.

Additional Rule Search Options

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

Image: Rule Search Results and Select Action Options Example

This example illustrates the fields and controls on the Rule Search Results and Select Action Options Example. You can find definitions for the fields and controls later on this page.



Search Result — Click this button to return to the Rule Search Options Page. Search results are displayed from last executed search, and the last selected Rule is highlighted. The Search Results button is only available when the search actions *Search for a Rule* or *Create Rule from Rule* are used.

Select Actions — Click this button to return to the Rules Engine Manager to select a new search action: *Search for a Rule*, *Add a new Rule*, *Create Rule from Rule*, or *Create a Rule from Rule Group*.

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.

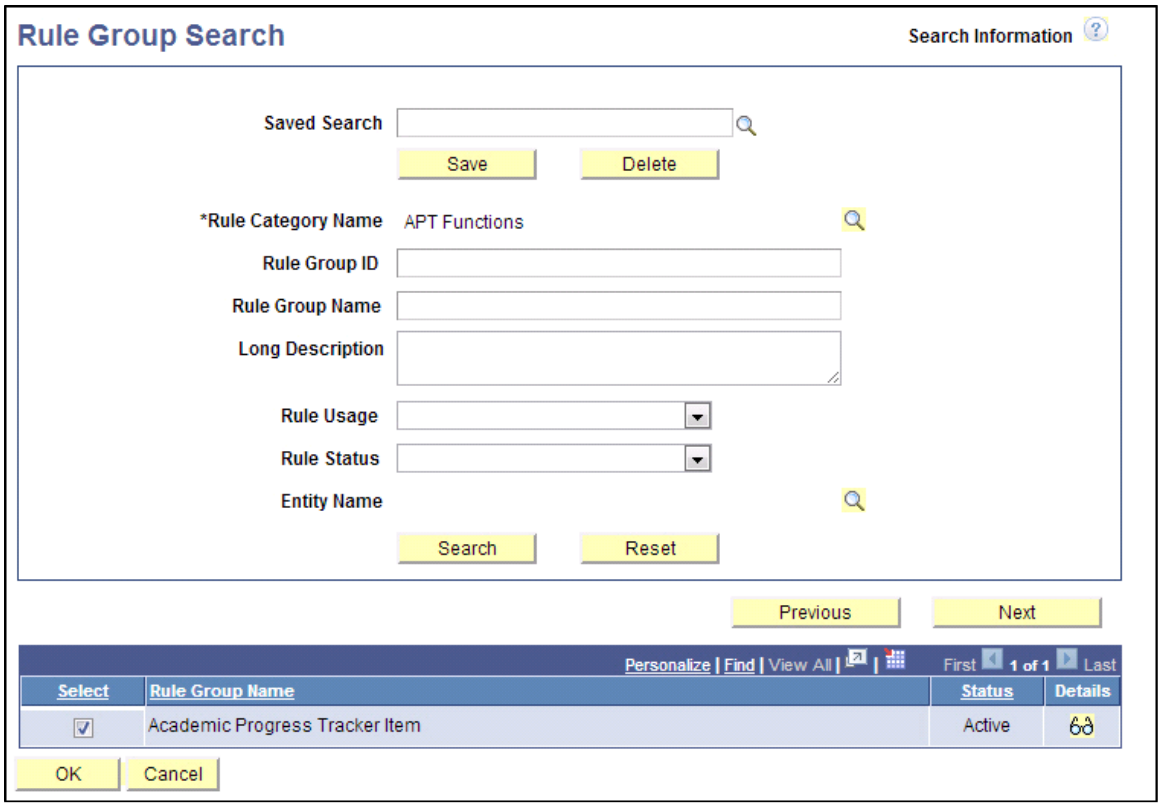
Access the Define Rule Groups Search page (Set Up SACR, System Administration, Rules Engine, Define Rule Groups).

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.

Image: Rule Group Search Options page

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

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.

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

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 *Defining Rule Groups*.

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.

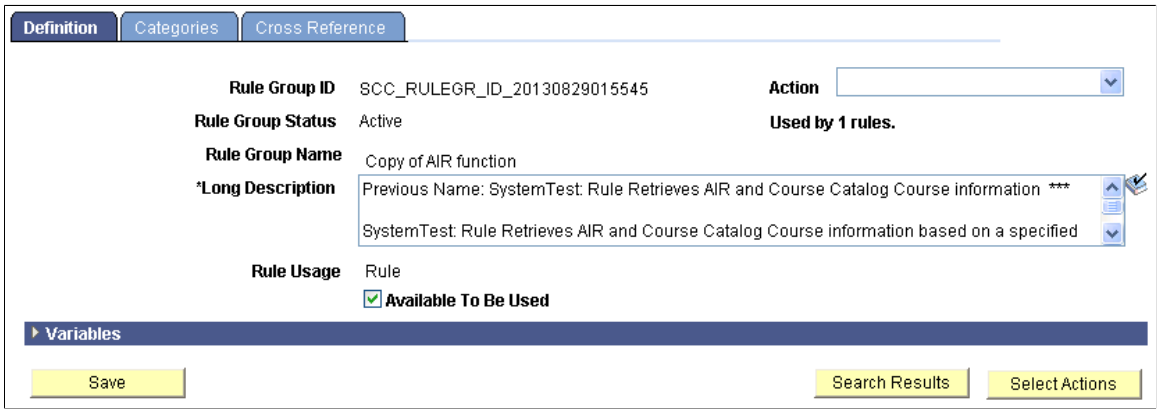
For more information on how to create and edit Rules, see *Defining Rule Groups*.

Additional Rule Group Search Options

Access the Rule Definition page (Set Up SACR, System Administration, Rules Engine, Define Rule Groups, select Search for a Rule Group, search for and select a Rule Group, Definition).

Image: Rule Group Search Results and Select Action Options Example

This example illustrates the fields and controls on the Rule Group Search Results and Select Action Options Example. You can find definitions for the fields and controls later on this page.



Search Result Click this button to return to the Rule Groups Search Options page. Search results are displayed from last executed search, and the last selected Rule Group is highlighted. The Search Results button is only available when the search actions *Search for a Rule Group* or *Create Rule Group from Group* are used.

Select Actions Click this button to return to the Define Rule Groups search page to select a new search action: *Search for a Rule Group*, *Add a New Rule Group*, *Create a Rule Group from Rule*, or *Create a Rule Group from Group*.

Creating Functional Rules

The section describes how to create functional Rules.

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).

Image: Define Rule page

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

Note: For Rules delivered with the system and that have *System Data* displayed in the Rules Engine user interfaces, only the Long Description can be modified.

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.

Action

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

Note: For System-delivered Rules, only the *Build Rule*, *Create New Group from Rule*, *Create New Rule from Rule*, and *Update Status Information* (if Rule Status is *Not Built*) Actions are available.

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.
- *Create New Group from Rule* – Selecting this Action transfers you to the Rule Group component. This Action is only available when a Rule is *Active* and not used by another Rule. The parameters for the new Rule Group are predefined based on the Rule from which it is being created.
- *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.

- *Create New Version of Rule* – Use this Action

  to introduce changes to Active Rules. By design, Rules are versioned rather than effective dated. Only one Version of a Rule can be active at any given time. This Action creates a new Version of the Rule with status *In Progress*. Activating the newest Version inactivates the previous Version automatically.
- *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.
- *Remove Base Entity* – Use this Action to remove the Base Entity from this Rule. This Action is available when Rules have a status of *In Progress* or *Active*. For Active rules, the setup option to allow changes to active rules must be selected. If the Rule has references to any Entity or Property which is logically part of the Base Entity Hierarchy structure, for example a child Entity or Property, then a warning is displayed.

Note: The Base Entity cannot be removed if the Rule is attached to a Rule Group.

- *Remove Rule Group from Rule* – If you use this Action, create new version of Rule and deselect the “Available in Other Rules” check box in the old Version/inactivated version. This Action is only available when a Rule is part of a Rule Group and has a Rule Status of *In Progress*.

Note: Concerning the effects of adding a New Rule Group to an existing Rule:
When a Rule Group needs to change and new Input or Output Parameters need to be added or removed, the process requires that you create a new copy (clone) of the Rule Group. New Parameters and Fields are then added to the New Rule Group. As the Rule Group provides the template for Rule Input and Output, all Rules associated with the old Rule Group need to move to the new Rule Group. This can be done using the following steps:
1. Deselect the Available in Other Rules option of the Rule attached to old Rule Group.
2. Create a new Version of the Rule attached to the old Rule Group. This action creates a new Version of the Rule with a Rule Status of *In Progress*.
3. Remove the Rule Group attached to the new Version of the Rule.
4. Add a new Rule Group to the new Version of the Rule.
When a Rule Group is removed using the *Remove Rule Group from Rule* option, a new Rule Group can be added using the Rule Group Name prompt search. When adding a Rule Group to a Rule, the Rule must have the same or fewer input and output parameters than the Rule Group.
New input and output Parameters can be added from the Rule Group to a Rule. In the case where there are input or output parameters defined on the Rule which do *not* exist in the Rule Group, they must be removed from the Rule first before adding it to the Rule Group.
Make any other needed changes to the new Version of the Rule, build the Rule, and test all changes. Then, activate the new Version of the Rule. The old version of the Rule is automatically deactivated.

- *Test Rule* – Opens the Test Rule window.

  For more information, see *Testing Rules*.

- *Update Status Information* – Use this Action to refresh page information and view the latest Rule Build Status.

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 drop down 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.

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.

Note: Oracle recommends adding a Rule Group Name directly after you have created your Rule or Function and before adding Entity Profiles, Rule Usages, or Variables as they may conflict with those that are defined in the Rule Group.
If you select a Rule Group that has Variables, a Rule Usage, or Base Entity that conflicts with the Rules you are creating, a warning message appears and the Rule Group is not added.

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 is 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 meant to be tied to a specific system event using the Trigger component. Triggers which are delivered with the System can be versioned; unlike other Rules or Functions. This means that when a Trigger has been attached to a Trigger component and marked as System, you can create a new Version of the Trigger which can subsequently be edited.
- 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 *Understanding Contextual Referencing*.

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.

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 *Defining Rule Category Security* and *Understanding Contextual Referencing*.

See Setting Up Entity Registry.

View Entity Hierarchy

Click this link to display the Entity Hierarchy view for the Base Entity attached to the Rule.

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.

Entity Data Load Option

Note: This field is available and required when the Rule Usage is *Rule* and an Entity Name has been entered.

Select the Data Load Option when creating a Rule.

- *Select Data by Criteria*: This is the default option. The Criteria grid is available. Data from the attached Entity is selected according to Criteria provided.
- *Select All Data*: The Criteria grid is not available. All data is retrieved from attached Entity and brought into the Rule.
- *Select No Data*: The Criteria grid is not available. No data is retrieved from the attached Entity. Use this option when the intent is to insert a new row of data into the Base Entity using the CREATE-ENTITY statement without first retrieving data.

Logging Level

Select the Logging Level to use when testing this Rule:

- *Error Messages* – The first level of logging. This option only show errors.

- *Informational Messages* – This option shows all informational messages.
- *No Message Logging* – Turns off logging at the build level.

Note: Oracle recommends that all Rules run on a production environment have this setting to optimize performance.

- *Trace Logging* – Show all errors, warning, errors, information messages, and the following:

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

  - Statements

    Descriptions from the Rules Engine for every Statement that is run. If the internal Function has Trace Logging, its Statements are also shown. An internal Function is a Function which is called by the main Rule. If the Function itself does not have Trace Logging settings, no trace options are shown for the called Function.

  - Variable Maps

    The complete Variable Map, every variable and its value, is shown at the beginning and the end of each run along with any Variable Map for any called Function or Rule if Trace Logging for that Function or Rule is turned on.

    A Variable map example is Assign V_StudentID = ‘SSRN0012’ as an example to where a Variable is assigned.

  - Call Statements

    All arguments sent in and all returns received in the log file.

  - XML Dump of the Entity Before Rule Execution

    The XML Dump is created for rules with Entity Processing. The XML Dump contains a dump of the Base Entity prior to Entity processing.

  - XML Dump of the Entity after Rule Execution

    The XML Dump is created for rules with Entity Processing. The XML Dump contains a dump of the Base Entity after Entity processing has completed. Any changes to the Entity (inserted or updated data) are reflected in the Entity Dump.

- *Warning Messages* – This options shows all errors and warnings.
- *Write to Log* – Select this option to create a log file when processing Rules in Batch. *Write to Log* can be used in combination with Statement “Write To Log”

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

Available To Be Used

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.

Note: You *cannot* version Rules with a Rule Usage of *Rule* or *Function* that are delivered with the System and that have *System Data* displayed in the Rules Engine user interfaces. These Rules must be copied (cloned) and a new Rule or Function created.
You *can* version Rules with a Rule Usage of *Trigger* that are delivered with the System and have *System Data* displayed in the Rules Engine user interfaces.

Adding Variables to a Rule

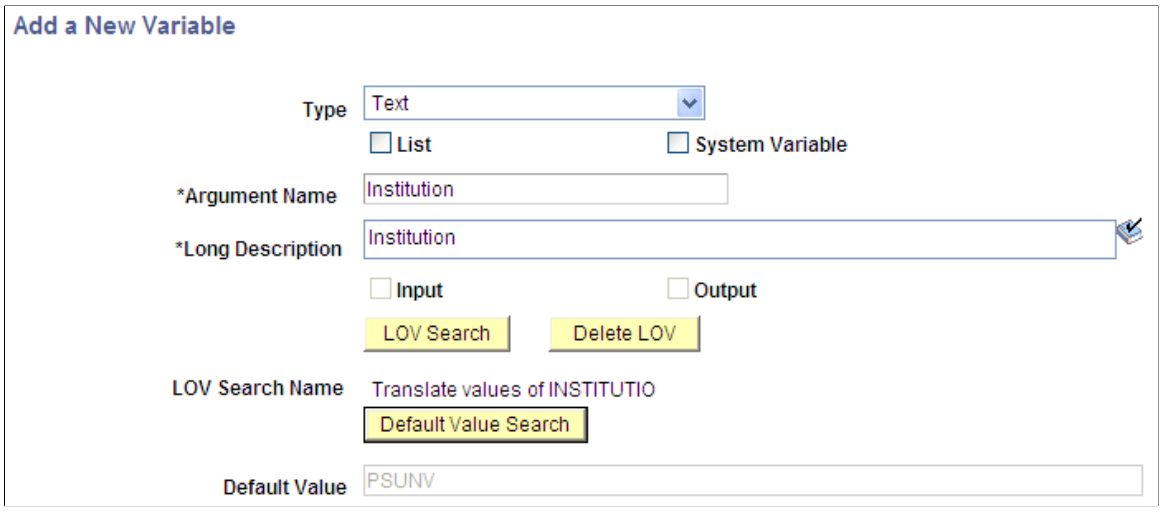
Once Rule options are defined, you can add the Variables you want to use in the Rule. The Variables grid on the Rules Engine Manager Definitions tabs is display-only. The grid shows whether variables are in use and whether they are used as input, output or required. Variables can be created before starting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

Image: Add a New Variable page

This example illustrates the fields and controls on the Add a New Variable page. You can find definitions for the fields and controls later on this 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.
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 Value 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 a Value 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.
LOV Search	Click this 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 Descr (description) link. Click the link to select an LOV. This returns you to the Add a New Variable page with the LOV Search Name displayed. **Note:** LOV Search results display a list of all available LOV values. The list is not limited to those LOV values which have been created specifically for the Rules Engine.

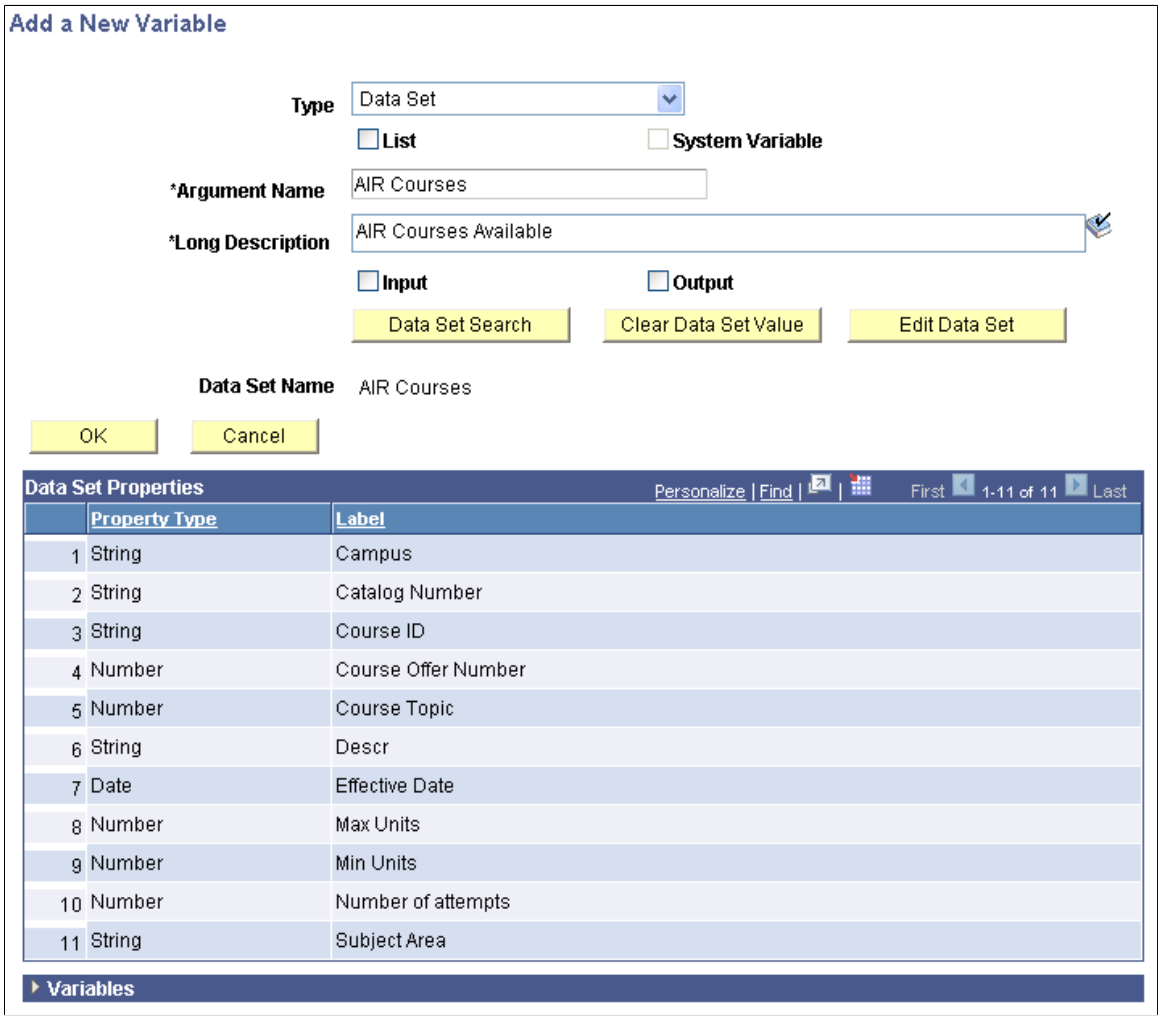
Delete LOV	Click this button to remove an LOV. This button becomes visible when a LOV has been attached to the variable
LOV Search Name	Displays selected LOV.
Default Value Search	Click this button to open the Prompt for (selected LOV) page. Search on a Field Name of *Description* or *Value*. Results display a Values link. 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.
Clear Default Value	Removes default value from Variable.

Adding a Data Set Variable to a Rule

When a *Data Set* variable Type is selected, there are different options to select.

Image: Add a New Variable – Data Set Option

This example illustrates the fields and controls on the Add a New Variable – Data Set Option. You can find definitions for the fields and controls later on this page.



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 Entity Profile link. Click the link 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.

Clear Data Set Value Click this button to remove the Data Set from the Variable.

Edit Data Set	Click this button to open a secondary page to add default values to the Data Set.

Adding Criteria to a Rule

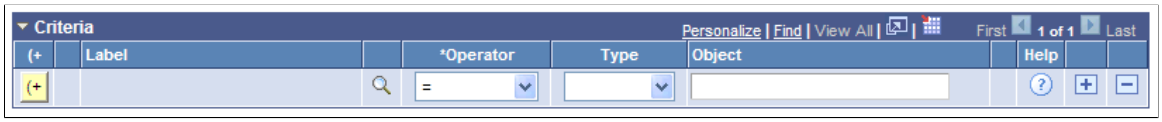
Once Rule options are defined and Variables are added for a Rule Usage of *Rule*, you can add Criteria to identify data from the chosen Base Entity. In the example being followed in this documentation, you want to add all criteria needed to identify and select the correct APT instance record for a particular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

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

Image: Criteria Grid – Initial Row

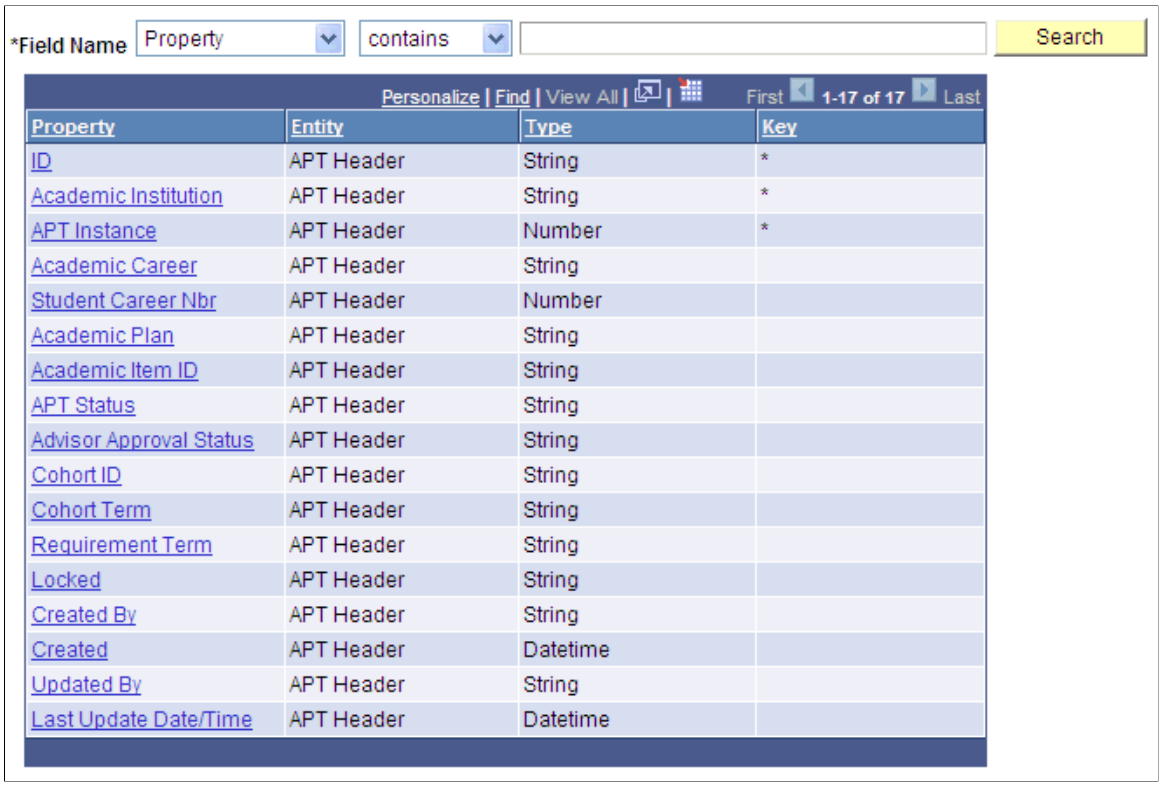
This example illustrates the fields and controls on the Criteria Grid – Initial Row. You can find definitions for the fields and controls later on this page.



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.

Image: Prompt for page

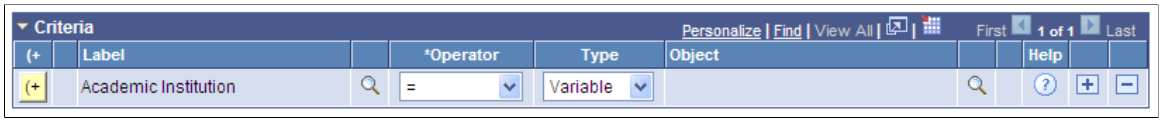
This example illustrates the fields and controls on the Prompt for page. You can find definitions for the fields and controls later on this page.



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.

Image: Example of Criteria Grid – Label Populated

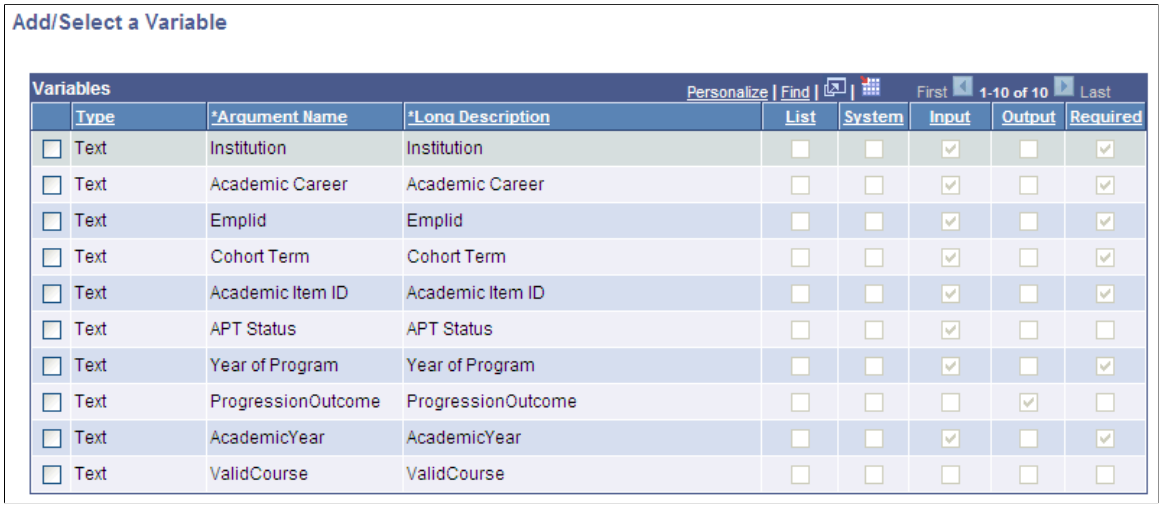
This example illustrates the fields and controls on the Example of Criteria Grid – Label Populated. You can find definitions for the fields and controls later on this page.



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

Image: Add/Select a Variable page

This example illustrates the fields and controls on the Add/Select a Variable page. You can find definitions for the fields and controls later on this page.

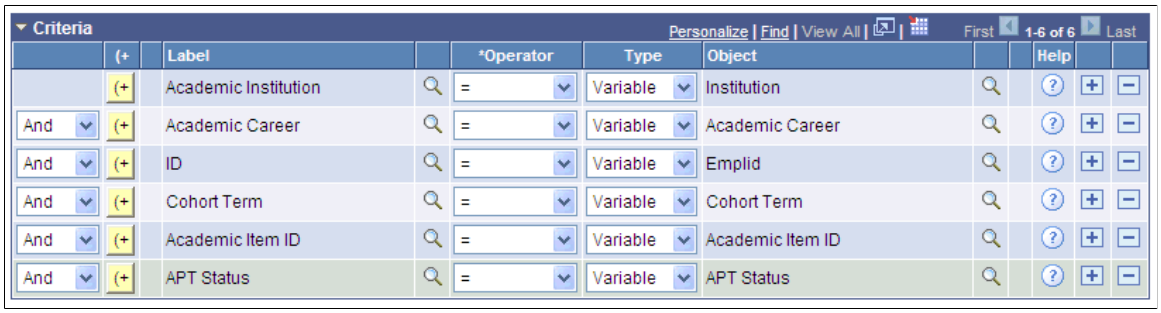


Select the check box next to the Variable you want to add, and click the Add button.

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

Image: Example of Criteria Grid

This example illustrates the fields and controls on the Example of Criteria Grid. You can find definitions for the fields and controls later on this page.



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.

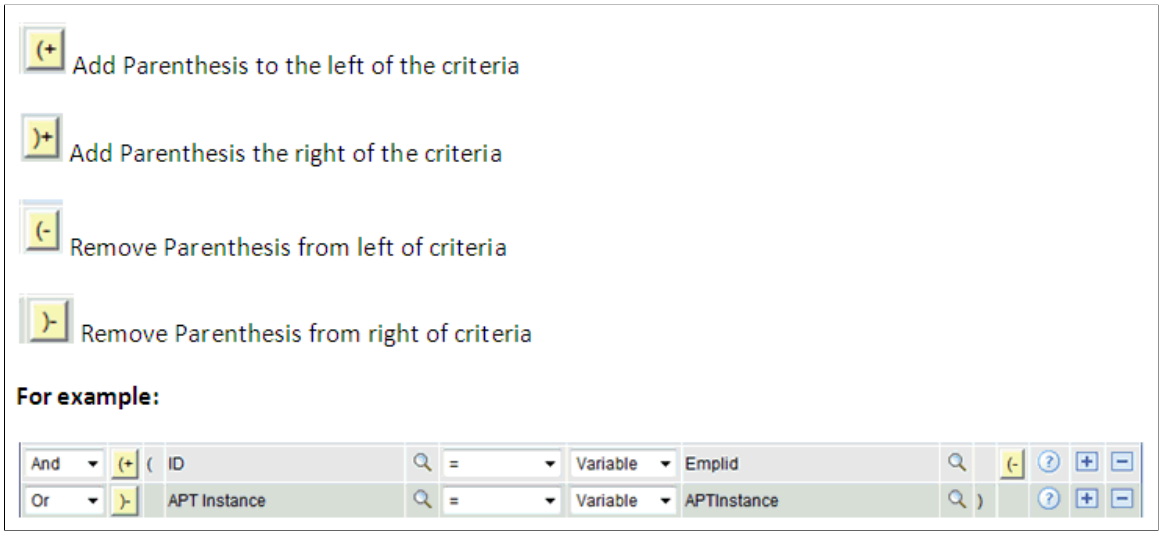
Here is more information about fields in the Criteria grid:

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.

Image: Example of Connectors and Parentheses

This example illustrates the fields and controls on the Example of Connectors and Parentheses. You can find definitions for the fields and controls later on this page.



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 always returns *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.
- *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.

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.

Here are possible options for Types and Objects in the Criteria grid:

Label	***Operator***	***Type***	***Object***
Numeric Property	= <> > >= < <=	Variable or Number	Open field or Prompts on *Number* Variables.
String Property	= <> > >= < <=	Variable or Text	Open field or Prompts on *Text* Variables.

Label	***Operator***	***Type***	***Object***
Date(time) Property	= <> > >= < <=	Variable or Date(time)	Open field or Prompts on *Date(time)* Variables.
Date Property and field name is Effective Date	Asofdate	Variable	Variable Special handling exists for selection of Effective Date. For Entities with an Effective Date, you can choose to have the system automatically select the maximum Effective Date.
Number Property and field name is Effective Sequence	FirstSeq LastSeq	No value available	Special handling exists for selection of Effective sequence. For entities with an effective sequence the user can choose to have the system automatically select the maximum or minimum effective sequence. Can be used in combination with the Asofdate option.
List Property	In Not In	Variable	*List* Variable which matches property Type
Text Property	Like Not like	Variable Text	Open field or *Text* Variable
Property of any type	Exists Not exists	No value available	<no value> If the Property is a Date or Date(time), any value not equal to Null exists. If the Property is a Number, any number unequal to 0 (zero) exists. If the Property is a String, any value unequal to space (blank) exists.

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.

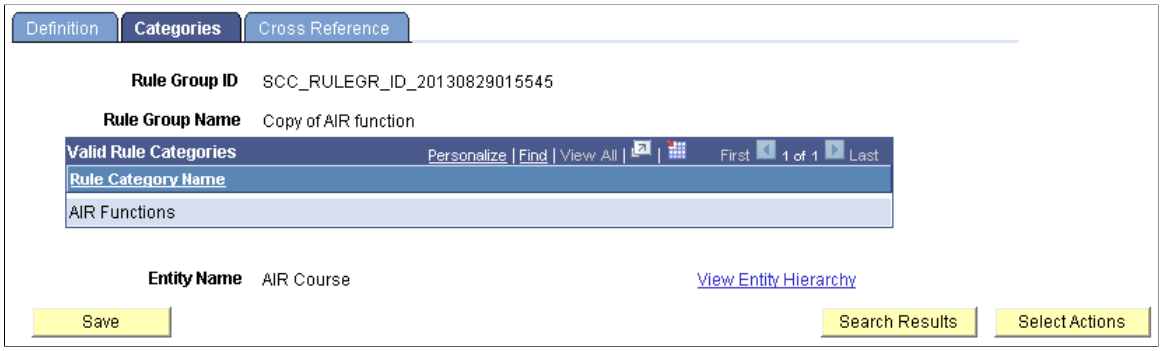
Note: Rule Groups should be created by Rules Engine Experts or Developers. The settings used in Rule Groups are enforced for any Rule added to the Rule Group. Once a Rule has been added to a Rule Group, the Rule's input and output parameters cannot be altered; however, it is still be possible to add 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).

Image: Define Rule Groups Categories page

This example illustrates the fields and controls on the Define Rule Groups Categories page. You can find definitions for the fields and controls later on this page.



Note: For Rule Categories delivered with the system, only the Long Description on the Definition page and Rule Groups and Security pages can be modified

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.

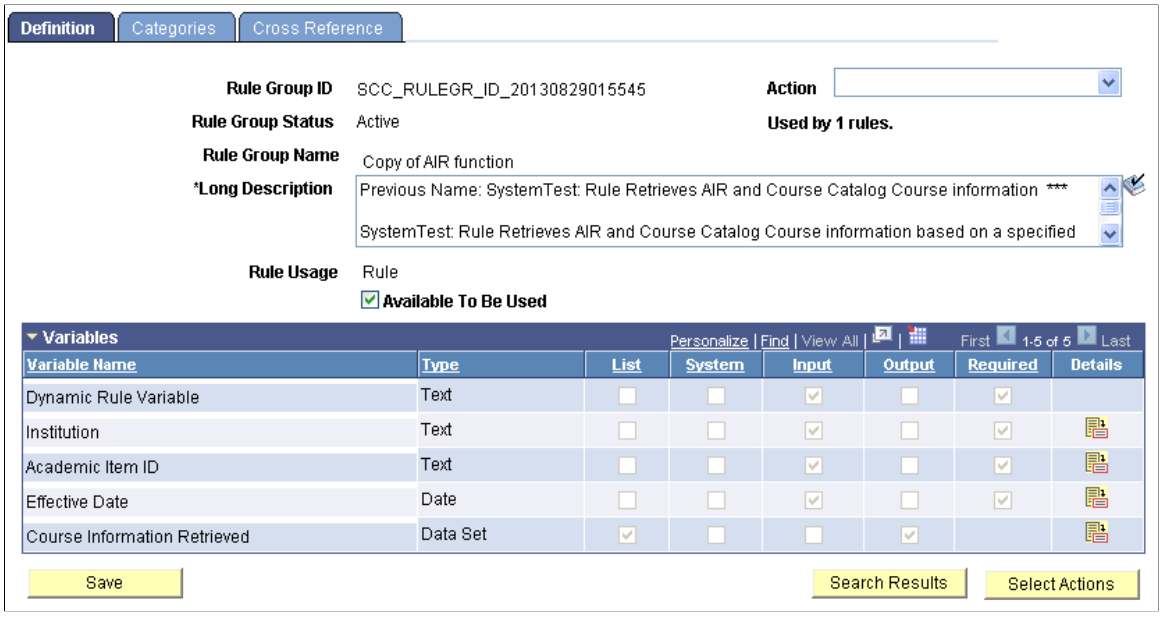
View Entity Hierarchy

Click this link to display the Entity Hierarchy view for the Base Entity attached to the Rule Group.

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).

Image: Define Rule Groups Definition page

This example illustrates the fields and controls on the Define Rule Groups Definition page. You can find definitions for the fields and controls later on this page.



Action

Note: For System-delivered Rule Groups, only the *Create Rule from Rule Group* and *Create Rule Group from Group* Actions are available.

Select an Action for this Rule Group:

- *Activate Rule Group* – In Progress Rule Group can be Activated using the Activate Rule Group option.

  Note: When creating a Rule Group from an Active or In Progress Rule, the Rule Group is immediately active and does not need not be activated. The Activate Rule Group action is not available.

- *Create Rule from Rule Group* – This action creates a Rule from the current Rule Group using the values (Variables, Rule Usage, Entity Profile from the Current Group). If multiple Categories are present for the Rule Group you are prompted to select one.

  Note: The *Create Rule from Group* action is only available for Active Rule Groups.

- *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* – Active Rule Groups can be inactivated using the Inactivate Rule Group option. This is only possible if *no* Rules are attached to the Rule Group
- *Remove Base Entity* – Use this Action to remove the Base Entity from this Rule Group. This Action is available when Rules have a status of *In Progress* or *Active*. For Active Rule Groups, the setup option to allow changes to active Rules must be selected. If an Active Rule Group has Rules attached, then the Base Entity may not be removed.

Note: The Base Entity cannot be removed if the Rule is attached to a Rule Group.

Note: The *Inactivate Rule Group* and *Delete Rule Group* actions are not available for Rule Groups with Active Rules attached.

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

Available To Be Used 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.

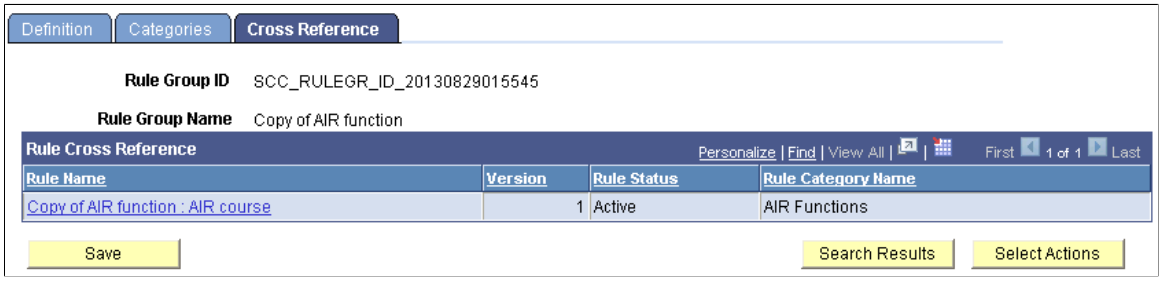
Add a Variable Select this button to open the Add a New Variable page.

For more information, see *Adding Variables to a Rule*.

Access the Rule Groups Manager Cross Reference page (Set up SACR, System Administration, Rules Engine, Define Rule Groups, search for a Rule Group and select the Cross Reference tab).

Image: Define Rule Groups Cross Reference page

This example illustrates the fields and controls on the Define Rule Groups Cross Reference page. You can find definitions for the fields and controls later on this page.



This page shows Rules and Rule Groups associated with this System Variable.

Rule Name Displays to authorized users links for Rules attached to this Rule Group. If a user is not authorized, the link is disabled.

Click a link to transfer out of the component and to the referenced object.

Warning! Make sure you have saved any data that you need to before confirming you want to transfer to the referenced object in a new component.

If no objects use the Rule Group, a notification is shown in place of the object details; for example, “This Rule Group is not used by any Rules”.

Version Displays the Rule Version Number.

Rule Status Displays the status of the Rule (*In Progress*, *Active*, *In-active*).

Rule Category Name Displays the Rule Category Name in which the Rule has been created.

Defining Rule Triggers

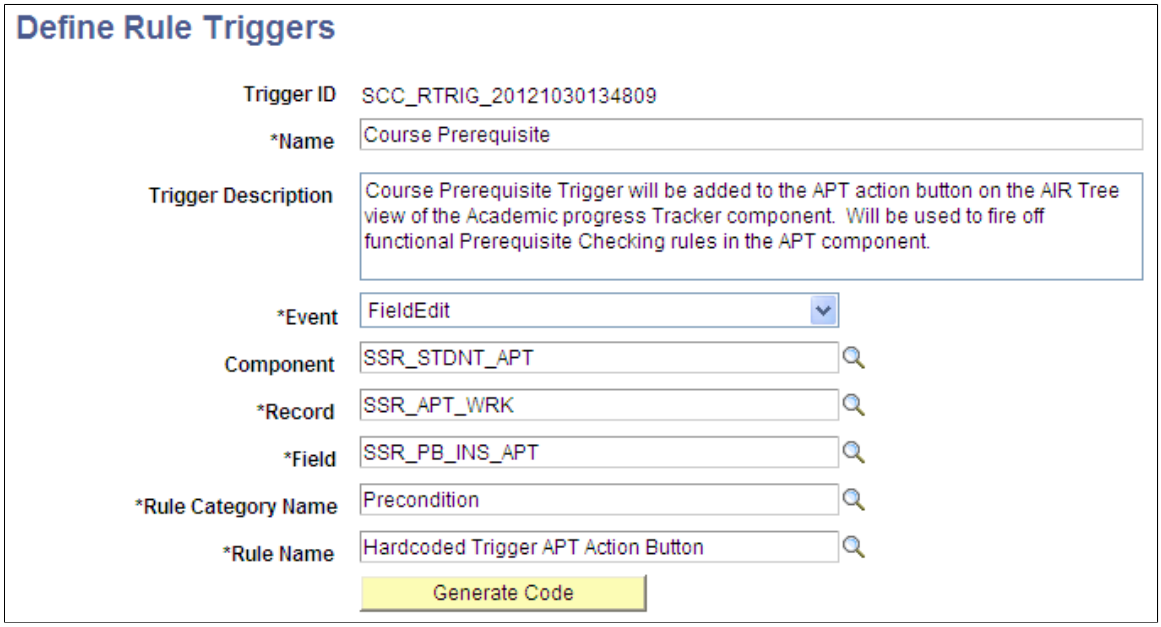
After a Trigger Rule is defined, a Trigger Definition can be created. Trigger Definitions allow you to associate the Trigger Rule to a Component Event or an Entity Method and generate PeopleCode that can be added to the associated component, record, or field event specified on the Define Rule Triggers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

Image: Define Rule Triggers page

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



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.

For information about integrating the Rules Engine with a user interface, see Integrating User Interfaces with the Rules Engine

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.

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.

Page Name	*Definition Name*	*Navigation*	*Usage*
Write to Log Statement	SCC_RULE_WLOG_SEC	On the Select a Statement page, select WRITE TO LOG	Select a WRITE TO LOG statement to log the contents of variables.

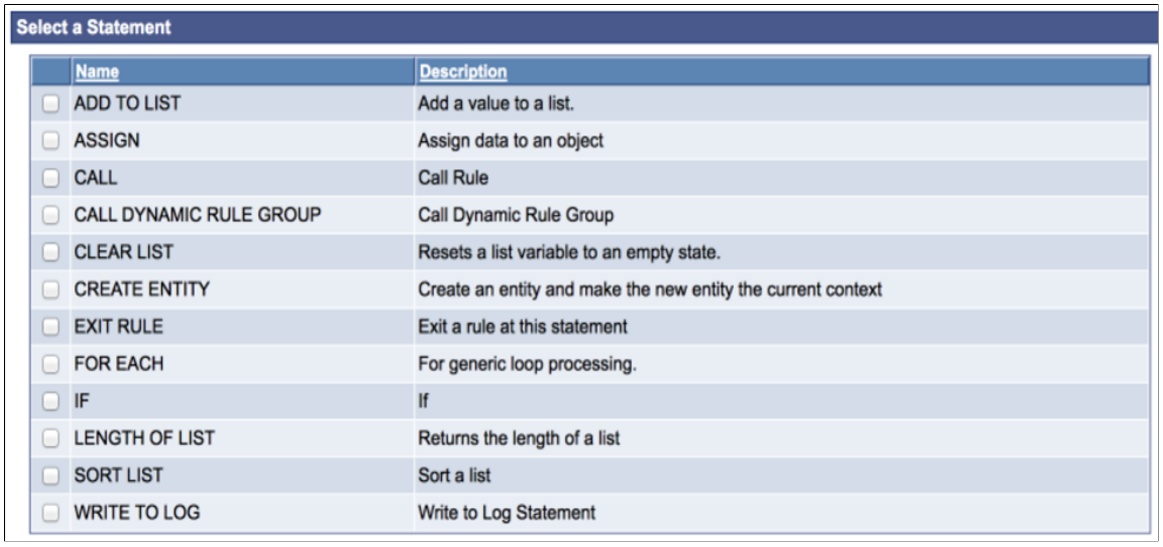
Understanding Statements for Evaluation and Calculation in a Rule

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).

Image: Delivered Statements

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



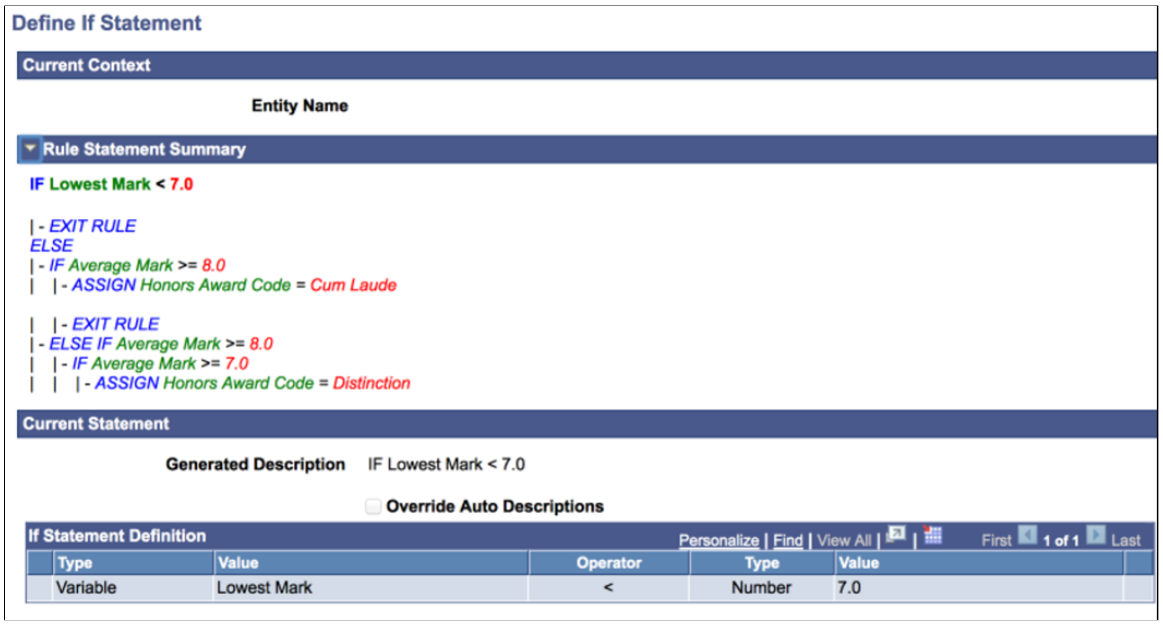
Understanding Common Statement Attributes

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).

Image: Example of Common Statement Attributes on a Statement Page

This example illustrates the fields and controls on the Example of Common Statement Attributes on a Statement page.. You can find definitions for the fields and controls later on this page.

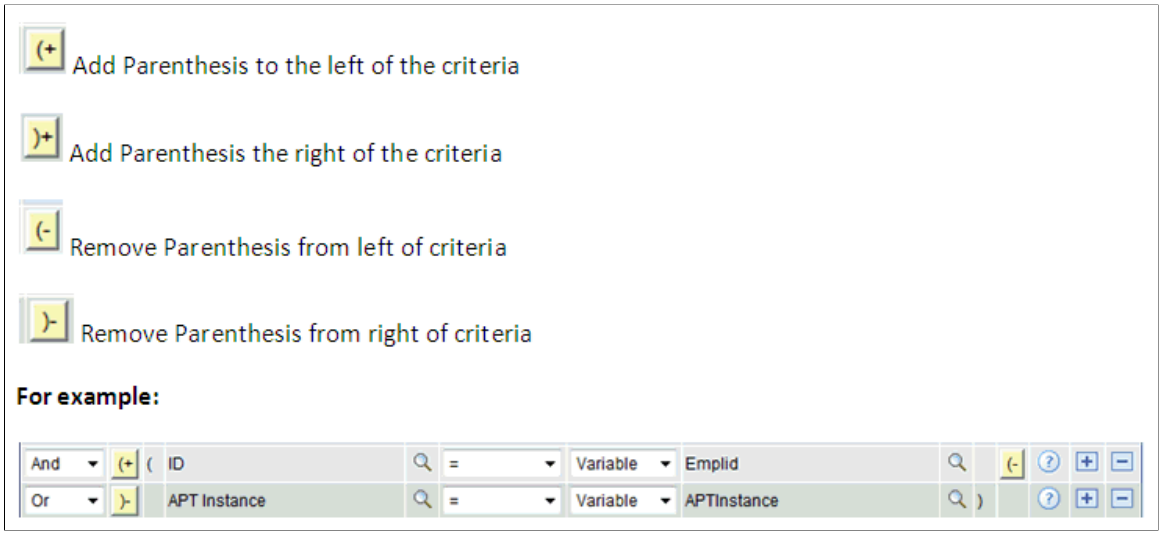


Connectors and Parentheses

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

Image: Example of Connectors and Parentheses

This example illustrates the fields and controls on the Example of Connectors and Parentheses. You can find definitions for the fields and controls later on this page.

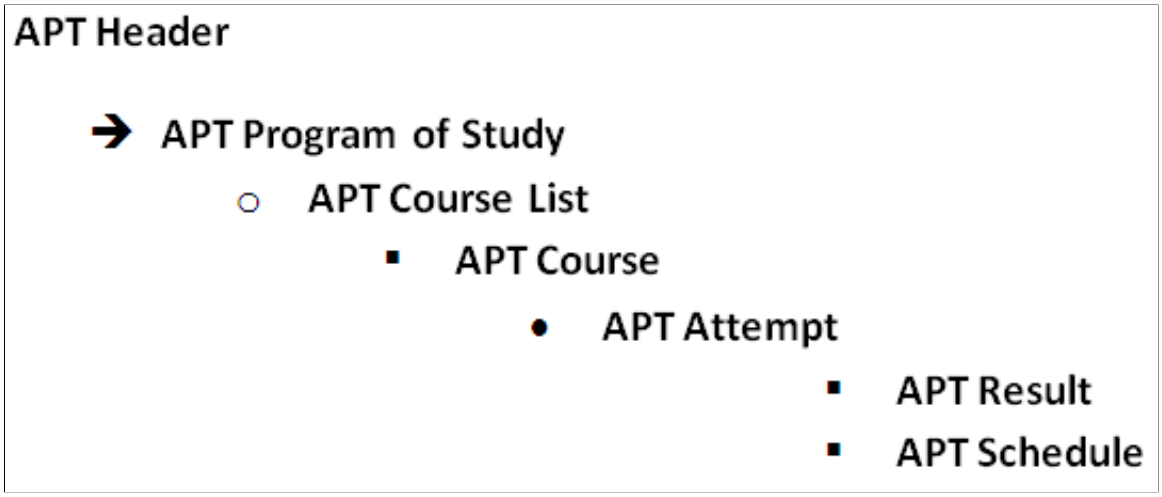


Current Context

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:

Image: Example of Academic Progress Tracker Entity Hierarchy

This example illustrates the fields and controls on the Example of Academic Progress Tracker Entity Hierarchy. You can find definitions for the fields and controls later on this page.



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 the Current 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.

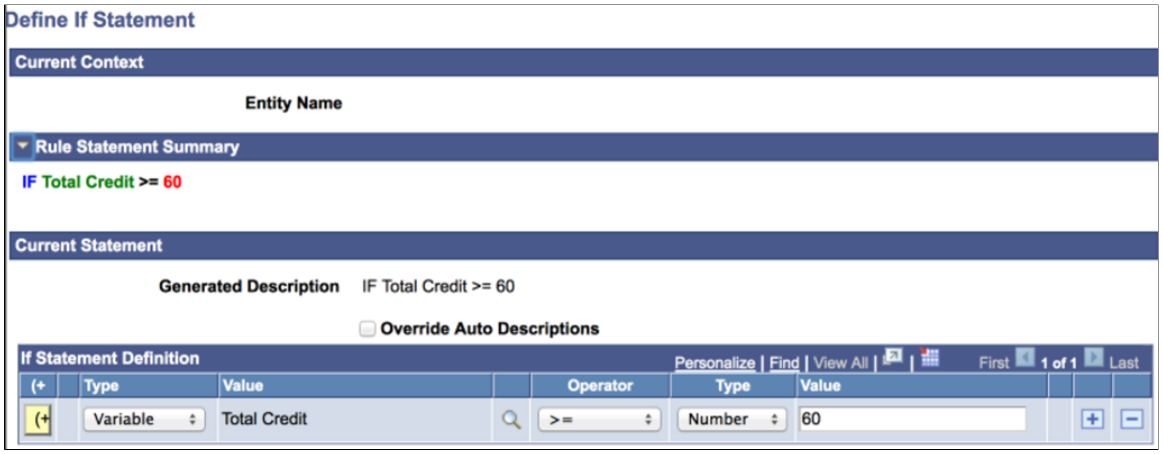
IF Statements

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.

Image: Example of IF Statement before Saving

This example illustrates the fields and controls on the Example of IF Statement before Saving. You can find definitions for the fields and controls later on this page.



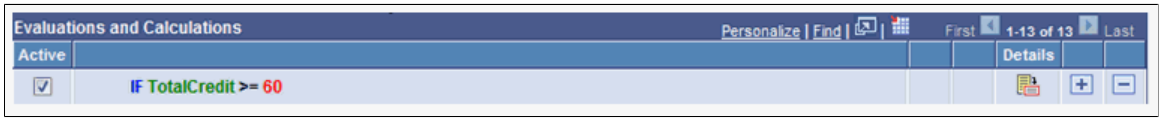
The possible combinations of Objects and Types in IF Statements are:

Type	***Object***	***Operator***	***Type***	***Value***
Property or Data Set Property (DS Prop) or Variable	Prompts on Properties or Prompts on Variables	> < >= <= = Like Not like	Variable or Text (*String* Property) or Number (*Number* Property or *Number* Variable) or Date(time) or Data Set Property	Prompt on Variable which matches a *Data Set Property* or *Variable* Type. or Text entry in an open field if Type is *Text*. or Numeric entry in open field if a Variable is numeric. or Date(time) in an open field if a *Data Set Property* or *Variable* is Date(time). or Prompt on *Data Set Property* which matches *Property* or *Variable* Type
Property or Data Set Property (DS Prop) or Variable	Prompts on Data Set Properties or Variables	In Not in	Variable	*List* Variable
Property or Data Set Property (DS Prop) or Variable	Prompts on Data Set Properties or Variables	Exists Not exists	no value	no value

The saved IF Statement looks like this:

Image: Example of IF Statement After Saving

This example illustrates the fields and controls on the Example of IF Statement After Saving. You can find definitions for the fields and controls later on this page.



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.

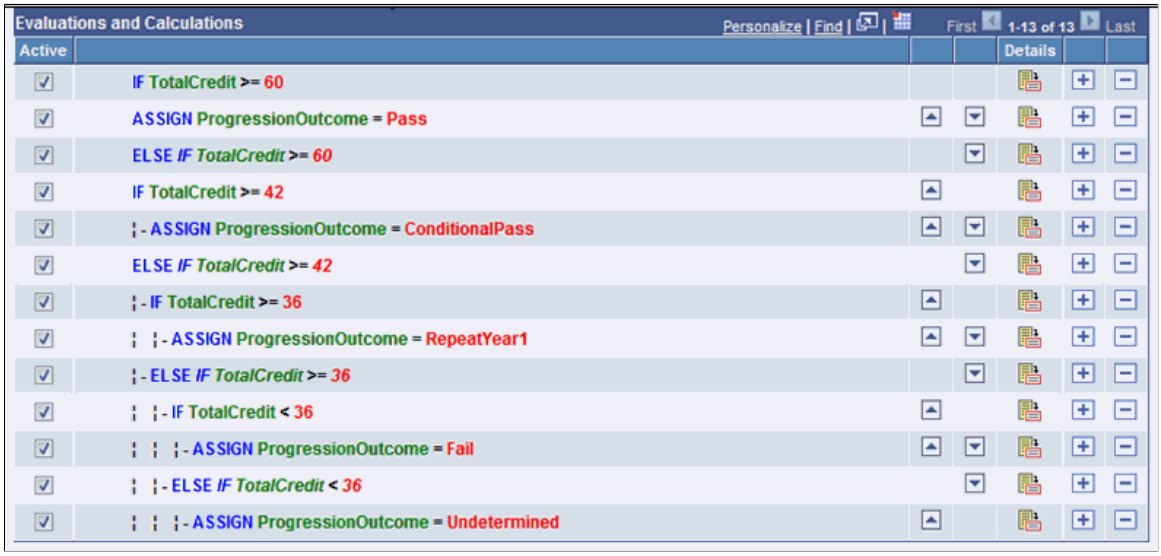
Image: Define ELSE Statement page

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

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

Image: Example of IF and ELSE Statements Used Together

This example illustrates the fields and controls on the Example of IF and ELSE Statements Used Together. You can find definitions for the fields and controls later on this page.



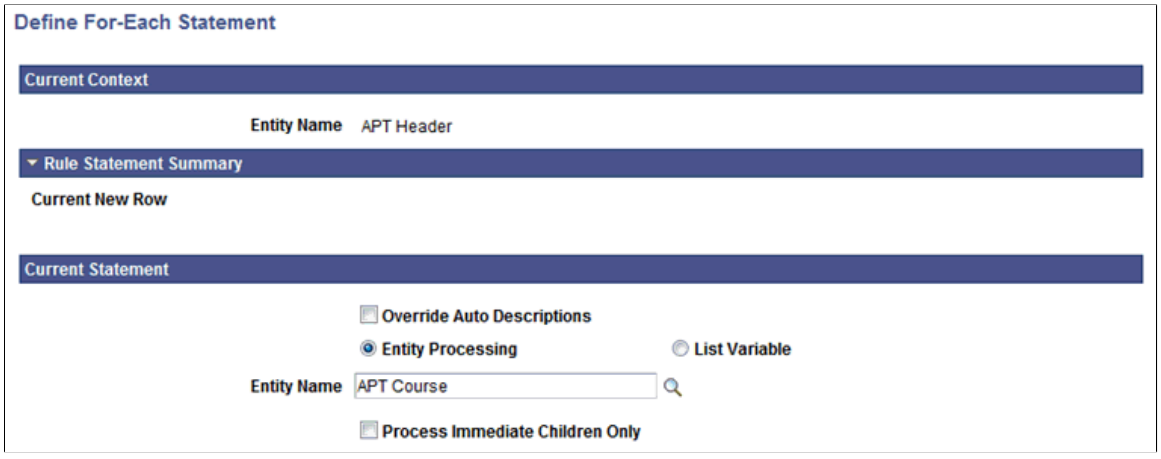
FOR EACH Statements

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:

Image: Define FOR-EACH Statement page

This example illustrates the fields and controls on the Define FOR-EACH Statement page. You can find definitions for the fields and controls later on this 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.
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.

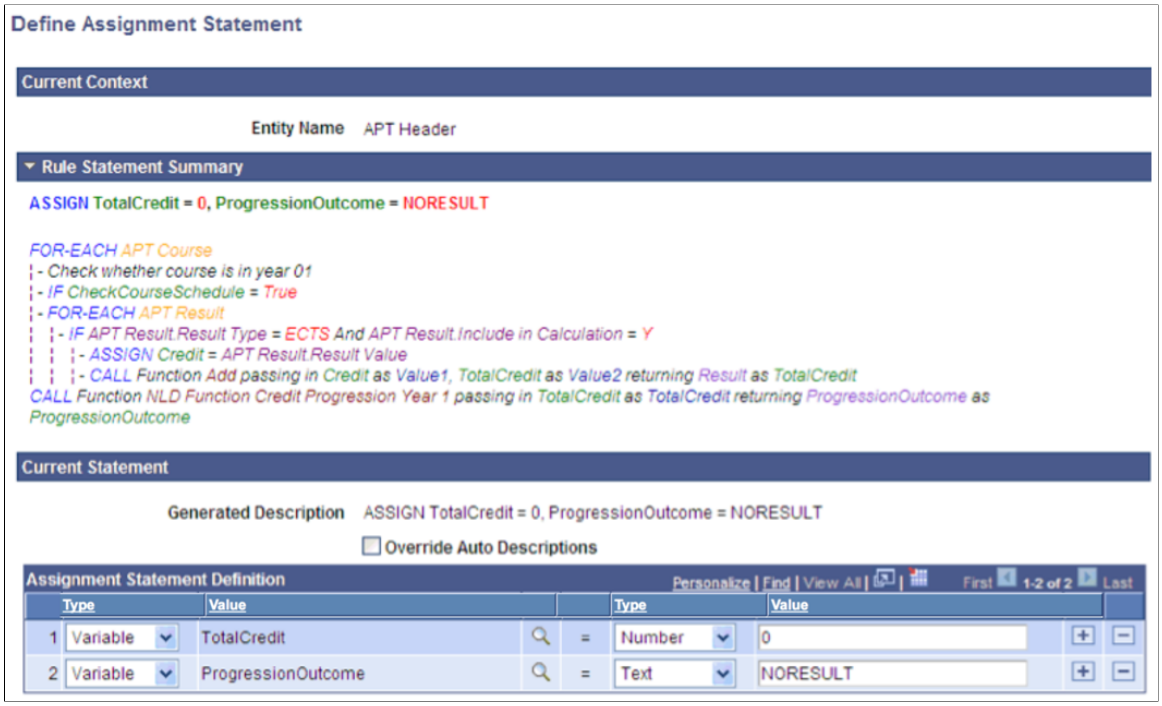
ASSIGN Statements

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:

Image: Define Assignment Statement page

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



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

Type	*Object*	*Operator*	*Type*	*Value*
Property or Data Set Property (DS Prop) or Variable	Prompts on Properties or Prompts on Variables	=	Variable or Text (*String* Property or Number (*Number* Property or *Number* Variable) or Date(time) or Data Set Property	Prompt on Variable which matches Data Set Property Type or Variable Type. or Text entry in open field if Type is Text. or Numeric entry in open field if Data Set Property or Variable is numeric. or Date(time) in open field if Data Set Property or variable type is Date(time). or Prompt on Data Set Property which matches Property or Variable Type.

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.

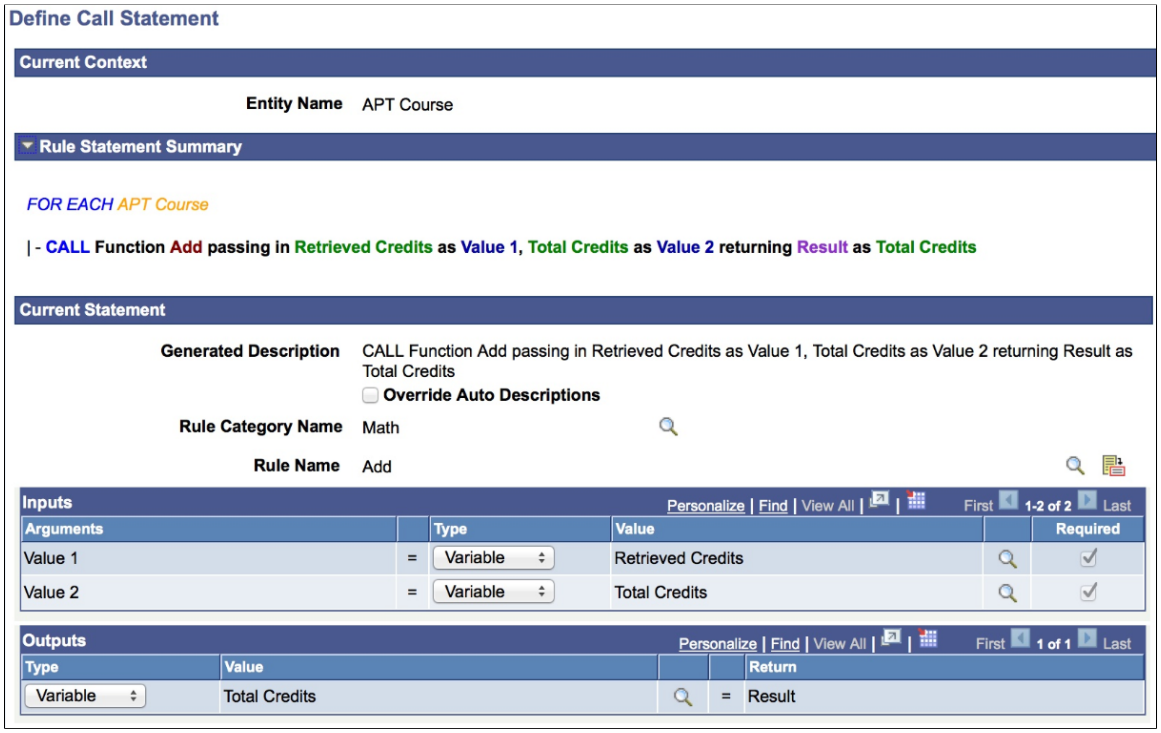
When you call a *Function* which uses Entity Data, you do *not* need to provide Criteria to find the correct data. That data is passed to the called Function automatically through Contextual Referencing.

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:

Image: Define Call Statement page

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

Click the Go To icon to the right of the Rule Name to open a new page that displays information on Rule Definition, Version History, and Cross Reference.

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.

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.

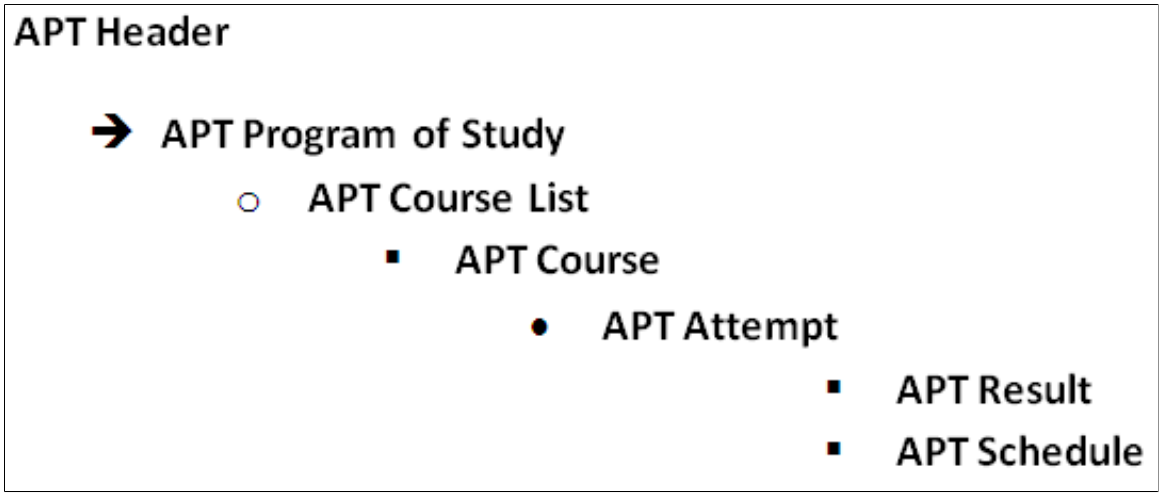
CREATE ENTITY Statement

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.

Image: Example of APT Header Hierarchy

This example illustrates the fields and controls on the Example of APT Header Hierarchy. You can find definitions for the fields and controls later on this page.



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

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 that calculation in the 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.

Note: The Rules Engine Manager has logic which validates nested Create Entity Statements that are present in the Evaluations and Calculations Grid upon saving the Rule.
A Create Entity Statement cannot be added to a Rule that is invalid, but it is possible to invalidate an existing Create Entity statement by removing a Statement from the Evaluations and Calculations grid or by deactivating a line on the Evaluations and Calculations grid. This would make the overall Rule invalid. For example, if a Create Entity statement is used to create the APT Attempt followed by a Create Entity statement to create the APT Result row, and the Create Entity APT attempt row is removed from the Evaluations and Calculations grid or moved to a new position after having been added, the overall Rule becomes invalid.
The APT Result can only exist in the Rule as a child to APT Course and not as a child to APT Attempt. Deletion, moving, and deactivating statements while the Rule is being built is allowed because the action may have been done in the course of correcting the Rule. However, when the Rule is saved, the whole Rule is validated and appropriate warning messages are displayed.

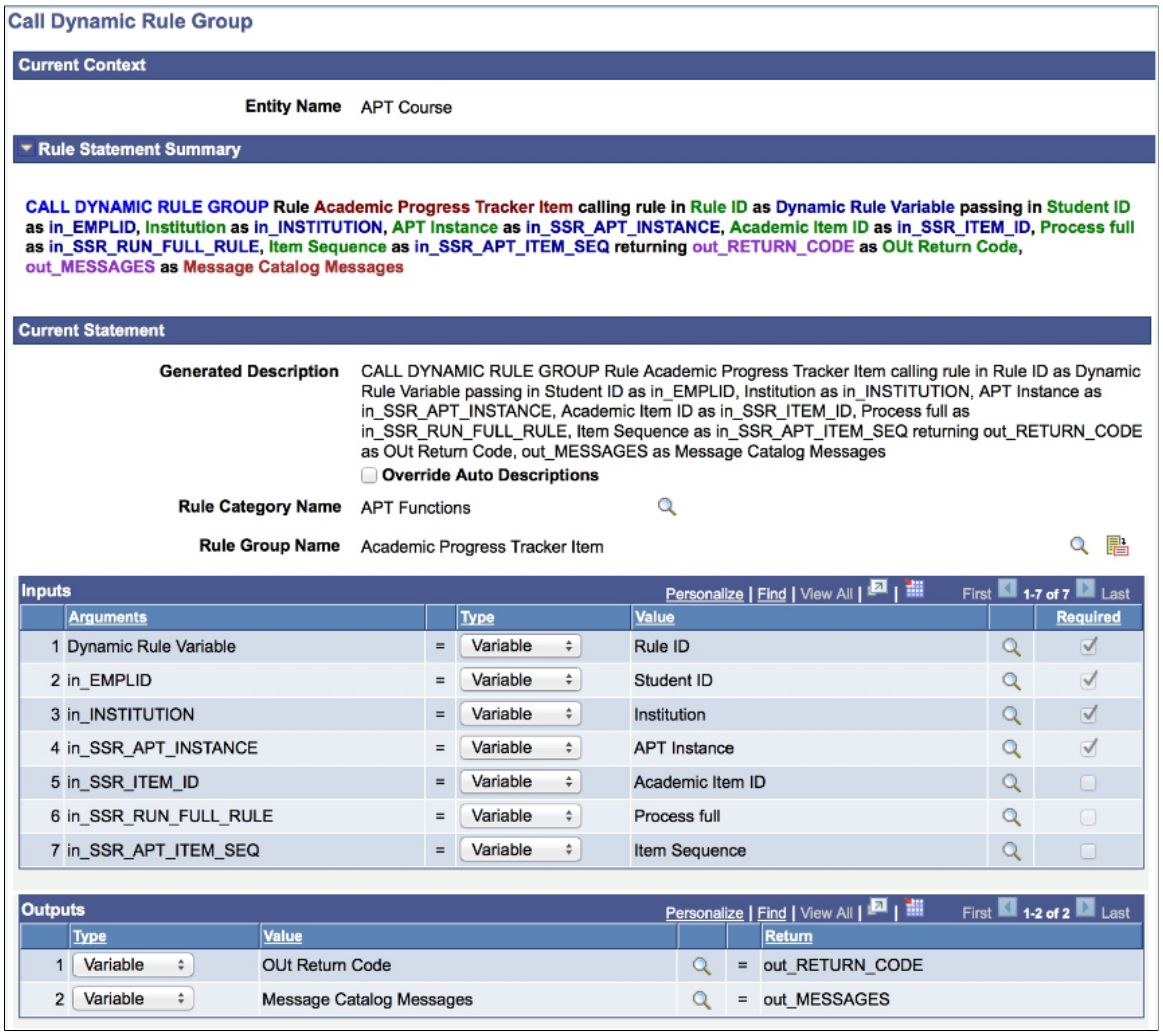
CALL DYNAMIC RULE GROUP Statements

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 *Understanding Rule Groups* and *Define Rule Groups*.

Image: CALL DYNAMIC RULE GROUP Statement page

This example illustrates the fields and controls on the CALL DYNAMIC RULE GROUP Statement page. You can find definitions for the fields and controls later on this 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.

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. Click the Go To icon to the right of the Rule Group Name to open a new page that displays information on Rule Group Definition, Categories, and Cross Reference.
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 is always 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.

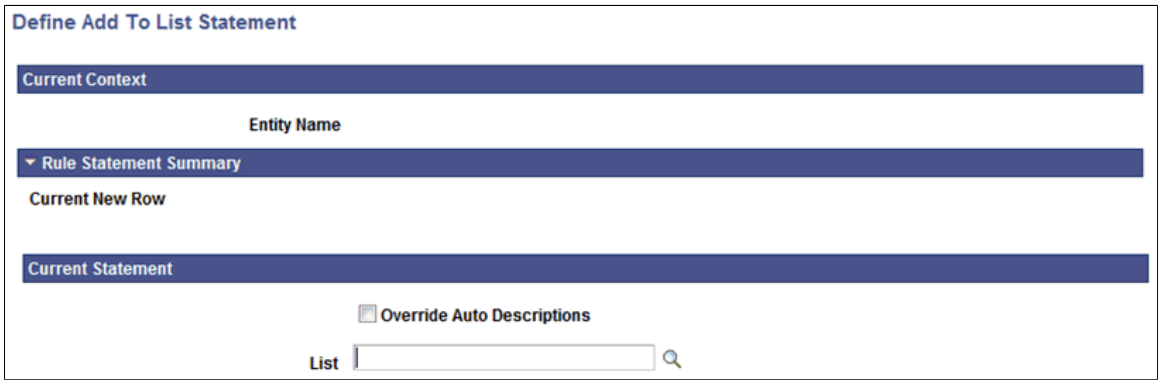
ADD TO LIST Statements

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.

Image: Define ADD TO LIST Statement page

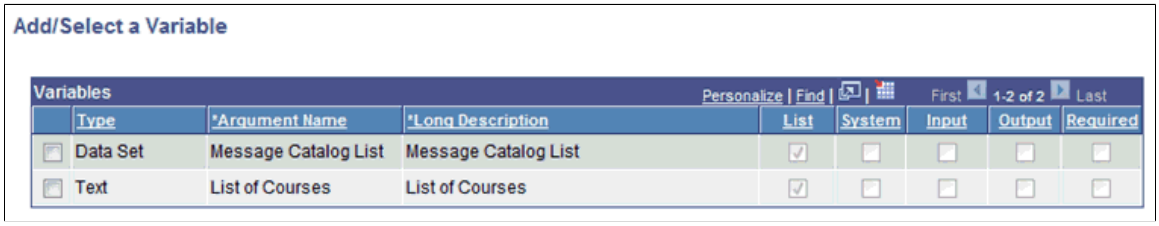
This example illustrates the fields and controls on the Define ADD TO LIST Statement page. You can find definitions for the fields and controls later on this page.



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

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

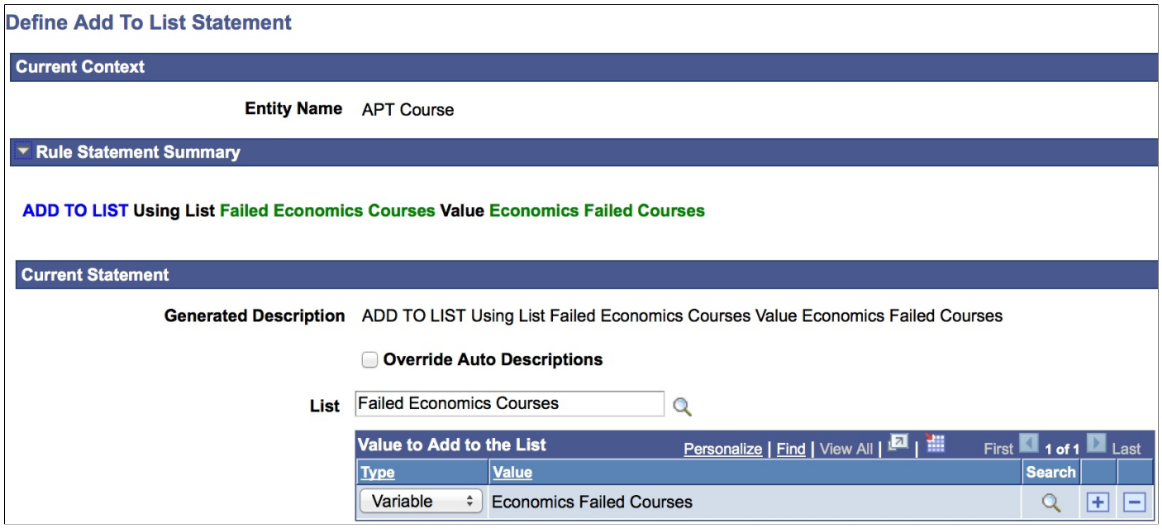
This example illustrates the fields and controls on the Example of Add/Select a Variable page with only List Variables. You can find definitions for the fields and controls later on this page.



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

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

This example illustrates the fields and controls on the Example of ADD TO LIST statement page with Value to Add to the List Grid. You can find definitions for the fields and controls later on this page.



- 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.

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.

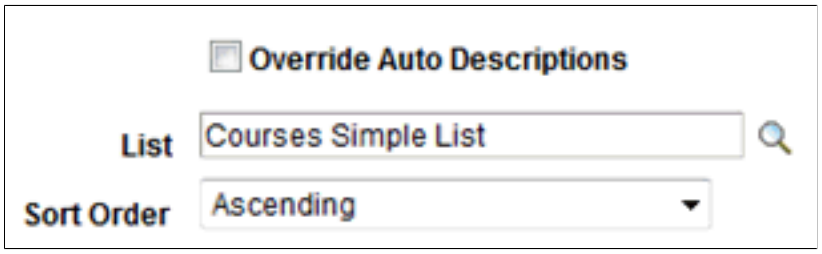
SORT LIST Statements

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*.

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

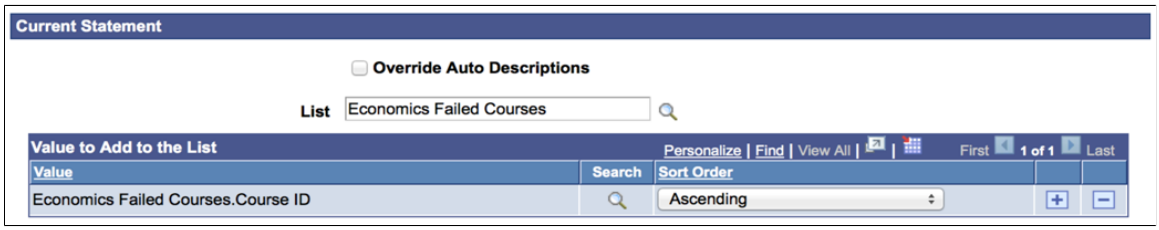
This example illustrates the fields and controls on the Example of a Sort Order Specification that is not a Data Set. You can find definitions for the fields and controls later on this page.



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.

Image: Example of a Sort Order Specification that is a Data Set

This example illustrates the fields and controls on the Example of a Sort Order Specification that is a Data Set. You can find definitions for the fields and controls later on this page.



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.

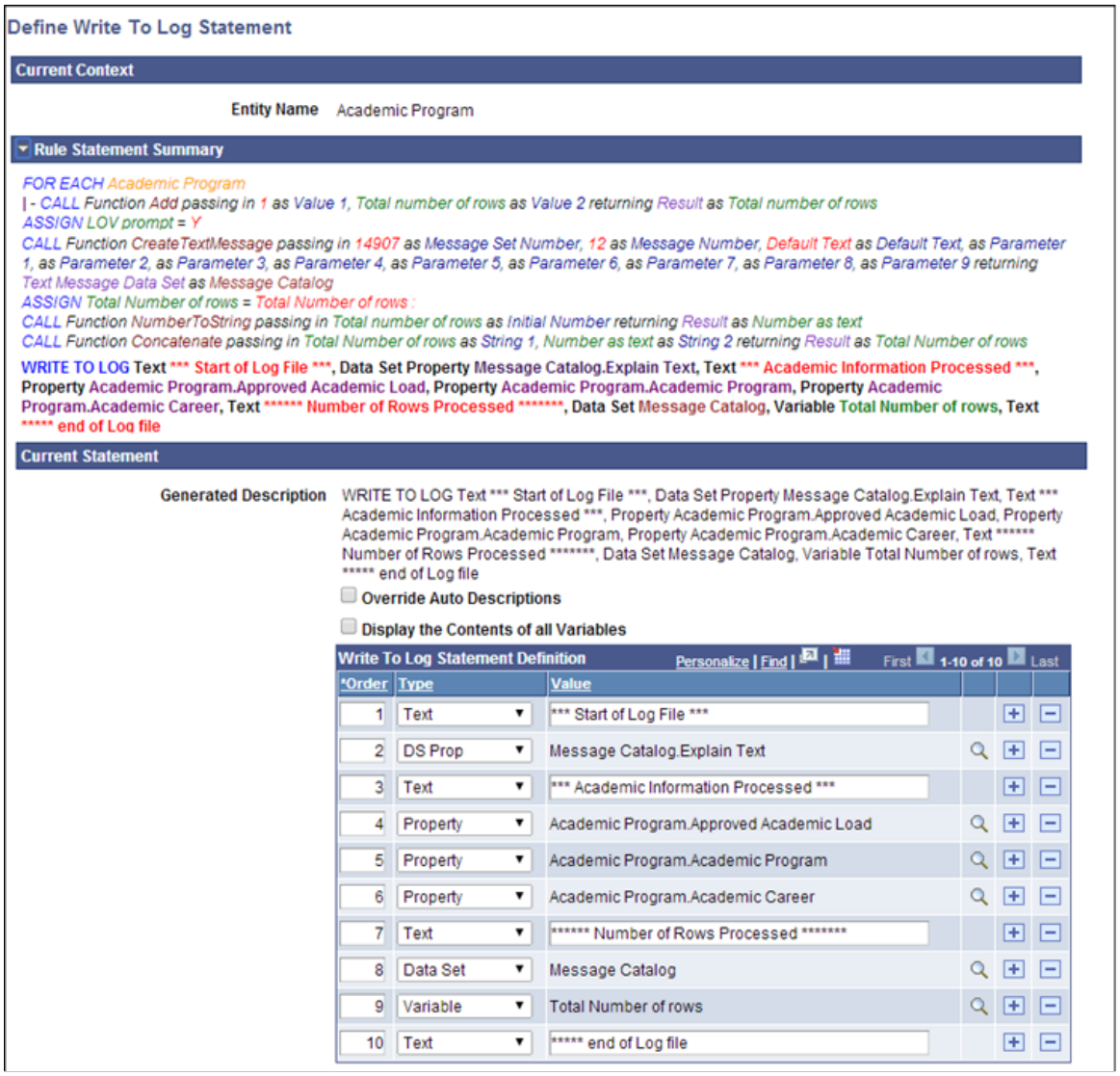
WRITE TO LOG Statements

Write to Log functionality can be used to provide Rule users with useful processing feedback when Rules are run in batch. For example, a log file can show information such as numbers of rows processed or retrieved.

The Logging Level *Write to Log* must be set for a log file to be created. Subsequently, the a Write to Log statement can be used to write text strings as well as the contents of Variables, Data Set Properties, and Entity Properties to a log file.

Image: Define Write to Log Statement page

This example illustrates the fields and controls on the Define Write To Log Statement page. You can find definitions for the fields and controls later on this page.



Display the Contents of All Variables Select this option to write all Rule Variable and their contents to the log file.

Use the Write to Log Statement Definition grid to add one or more values to be written to the log file. The Order number is automatically augmented when rows are added. Rows are written to the log file in the same order.

Type

Select a Value Type to include in the log:

- *Data Set* — The contents of the complete Data Set are written to the log file. This includes the Data Set Property Name followed by the Data Set Property Content. This option is available if the Rule references a Data Set
- *DS Prop* (data set property) — The value of the selected Data Set Property. This option is available if the Rule references a Data Set Variable.
- *Property* — The contents of the Entity Property are written to the log file.
- *Text* — Text entered by the user.
- *Variable* — Only Variables of type *Text* can be written to the log file. If you need the write the contents of a number Variable to a log, use the function *NumberToString* to convert your value first.

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

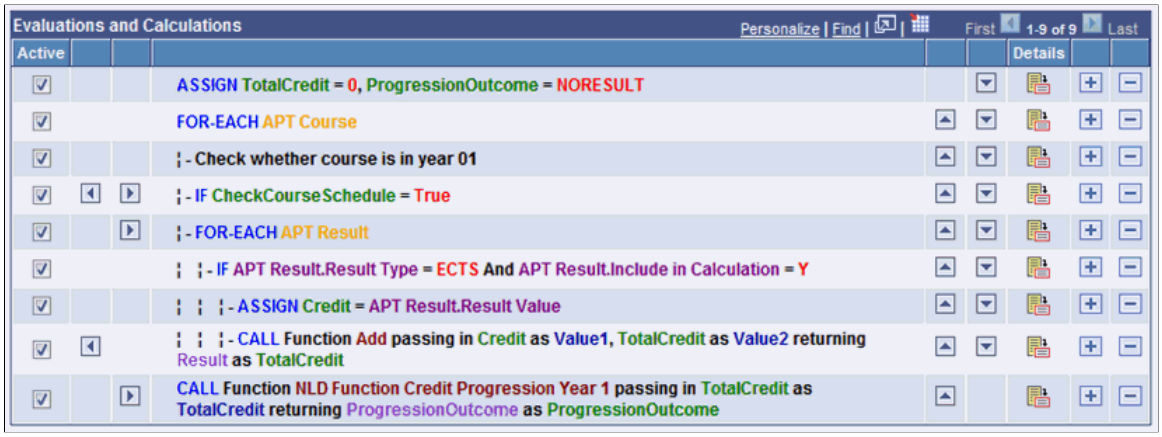
  Use the Up and Down Toggle fields to move a Statement before or after other Statements, changing the order in which Statements are executed. The type of Statement and where it is located in relation to other Statements determines if it can be moved up and/or down.

- By Indentation

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.

Image: Example Showing Statements with Indentations and Up and Down Toggles

This example illustrates the fields and controls on the Example Showing Statements with Indentations and Up and Down Toggles. You can find definitions for the fields and controls later on this page.



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

Applying Rule Concepts and Adding Statements to a Rule

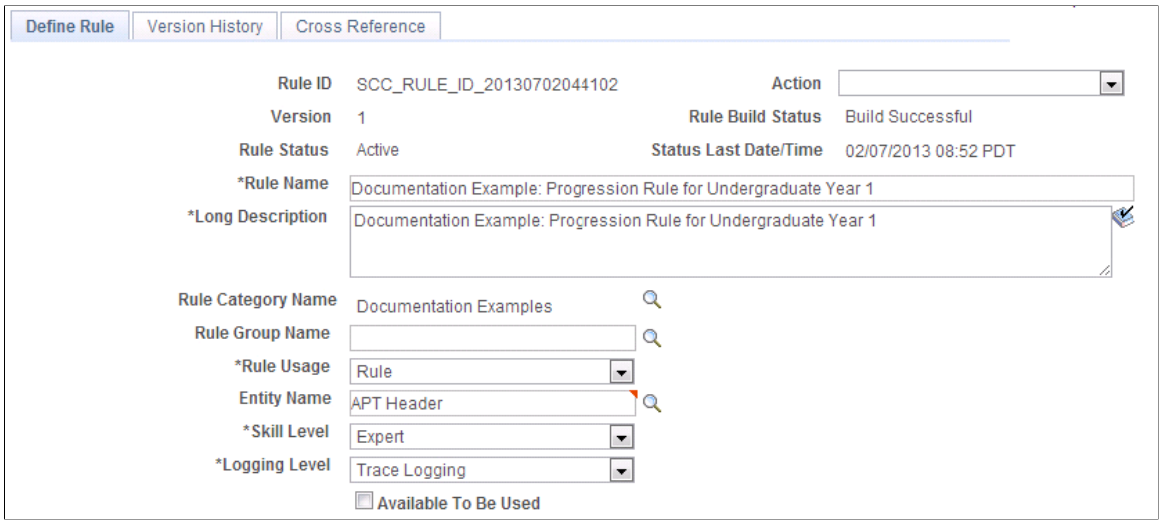
In this section, an example Rule illustrates how Rule Options, Variables, Criteria, and Statements are used together. Previous sections of the documentation describe creating Variables, using Criteria to define a data set, and the features of delivered Statements for Rule evaluation and calculation.

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.

Image: Example of Define Rule page for Progression Rule

This example illustrates the fields and controls on the Example of Define Rule page for Progression Rule. You can find definitions for the fields and controls later on this page.

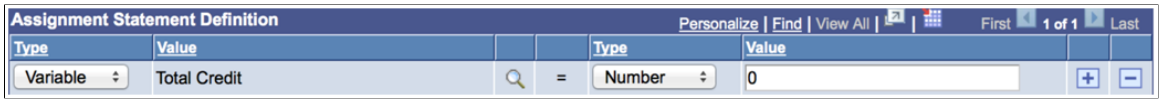


Add an ASSIGN Statement to Set the TotalCredit Field to Zero

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):

Image: Example of Assignment Statement Definition

This example illustrates the fields and controls on the Example of Assignment Statement Definition. You can find definitions for the fields and controls later on this page.



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:

Image: Example of ASSIGN Rule Text

This example illustrates the fields and controls on the Example of ASSIGN Rule Text. You can find definitions for the fields and controls later on this page.

For more information about ASSIGN statements, see “Understanding Statement-Specific Attributes, ASSIGN Statements.”

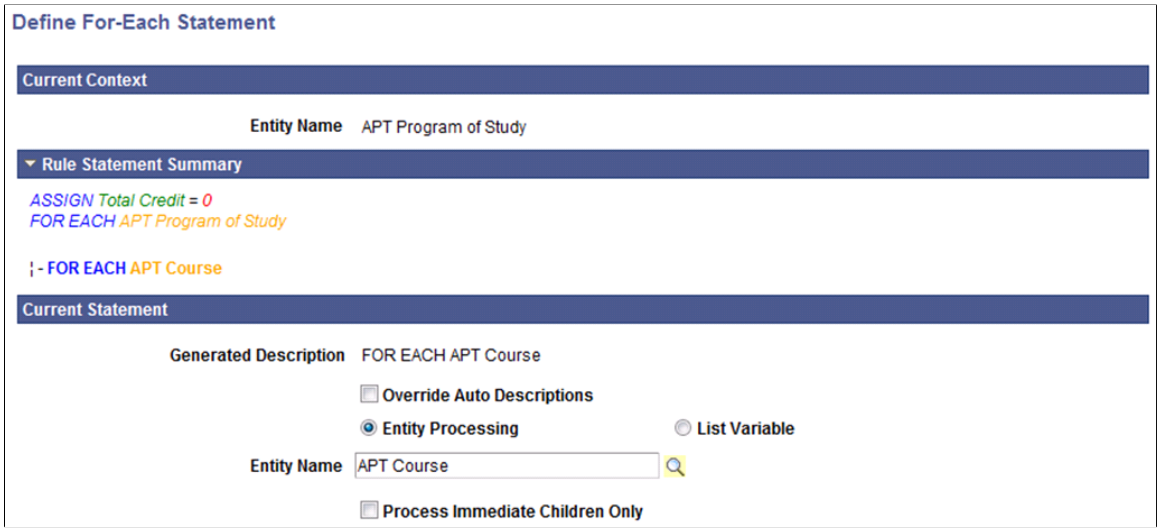
Add a FOR EACH Statement to Select All Courses

The next step is to retrieve all courses for the academic program of study which is associated with the Academic Progress tracker through the APT header. You need to retrieve the correct APT Program

of Study before selecting all associated courses. Select the APT program of study using a FOR-Each statement.

Image: Example of FOR EACH Statement Definition

This example illustrates the fields and controls on the Example of FOR EACH Statement Definition. You can find definitions for the fields and controls later on this page.

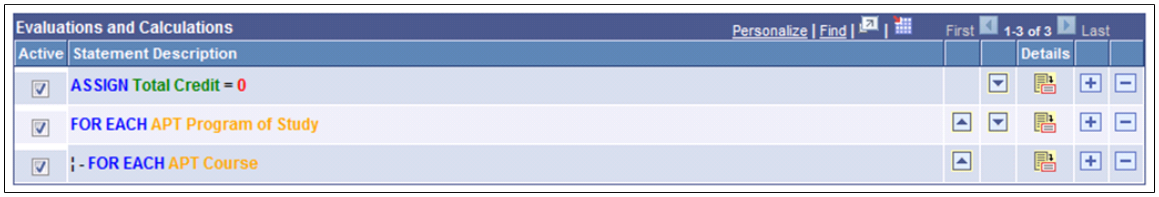


Select Entity Processing to activate the Entity Name field and prompt. Select the Search icon and choose the *APT Course* Entity. By selecting *APT Course*, you can retrieve all courses that are part of the APT instance selected as Base Entity. Do *not* select Process Immediate Children Only so the Rule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

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:

Image: Example of FOR EACH Rule Text

This example illustrates the fields and controls on the Example of FOR EACH Rule Text. You can find definitions for the fields and controls later on this page.

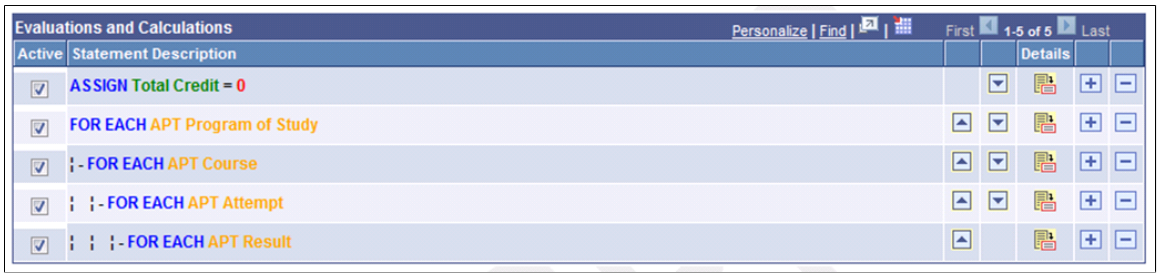


Add a FOR EACH Statement to Select a Result for Each Course

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.

Image: Example of Indented FOR EACH APT Result Rule Text

This example illustrates the fields and controls on the Example of Indented FOR EACH APT Result Rule Text. You can find definitions for the fields and controls later on this page.



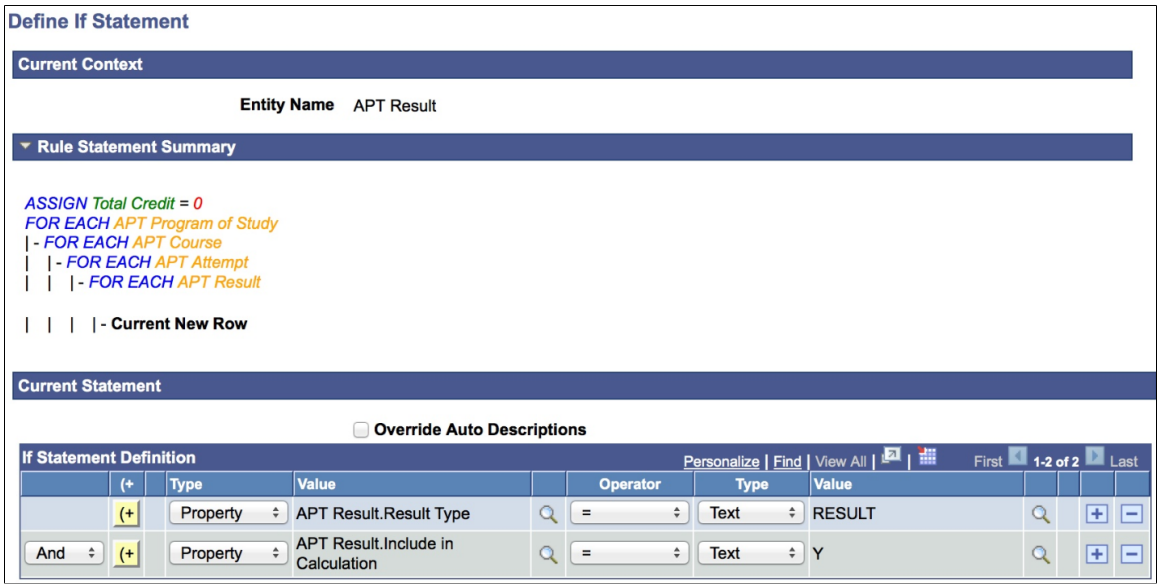
Note: The Rules Engine Manager has logic which validates FOR EACH statements upon saving the Rule. A FOR EACH statement cannot be added to a Rule that is invalid, but it is possible to invalidate an existing FOR EACH statement by removing a statement from the Evaluations and Calculations grid or by deactivating a line on the Evaluations and Calculations grid. This would make the overall Rule invalid. For example, if an APT Attempt row is removed from the Evaluations and Calculations grid or moved to a new position after having been added, the overall Rule becomes invalid.
The APT Result can only exist in the rule as a child to APT Course and not as a child to APT attempt. This allowed while the Rule is being built because the action may have been done in the course of correcting the rule. However, when the rule is saved, the whole Rule is validated and appropriate warning messages are displayed. If a nested FOR EACH statement is used in combination with a CREATE ENTITY statement, the same logic applies.

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:

Image: Example of Define IF Statement Definition

This example illustrates the fields and controls on the Example of Define IF Statement Definition. You can find definitions for the fields and controls later on this page.



Add a CALL Statement to Sum Credits

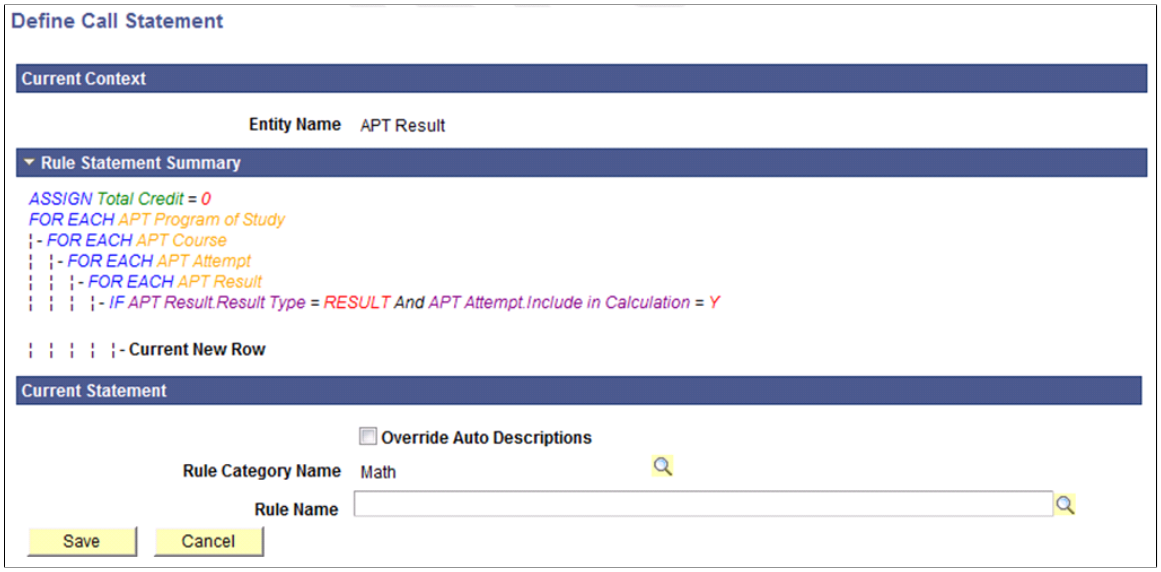
After adding specified criteria for selecting results, use a CALL statement to call a Function to sum the retrieved credits. In the Define CALL Statement page, pre-delivered Functions are available that are restricted to those which are available in the Rule Categories, including common math functions.

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:

Image: Example of Define CALL Statement

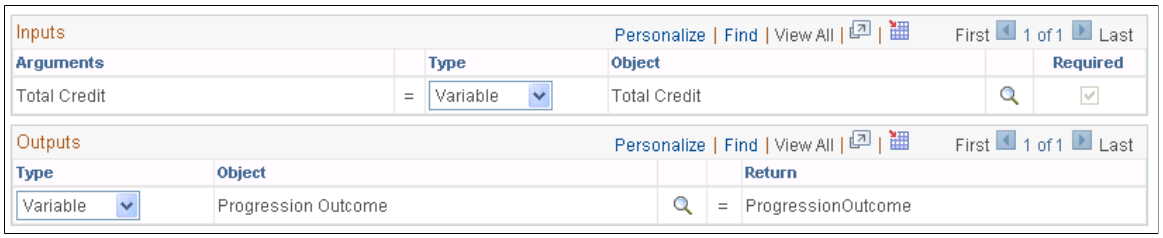
This example illustrates the fields and controls on the Example of Define CALL Statement. You can find definitions for the fields and controls later on this page.



When calling a Function, the Function itself indicates which Input and Output parameters are required. Only the valid Property and Variable Types are shown. In this case, only numeric Variables and Properties can be selected:

Image: Example of CALL Statement Inputs and Outputs

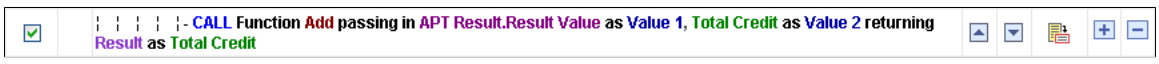
This example illustrates the fields and controls on the Example of CALL Statement Inputs and Outputs. You can find definitions for the fields and controls later on this page.



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:

Image: Example of CALL Statement Rule Text

This example illustrates the fields and controls on the Example of CALL Statement Rule Text. You can find definitions for the fields and controls later on this page.



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:

Image: Example of Academic Progression Function (1 of 2)

This example illustrates the fields and controls on the Example of Academic Progression Function (1 of 2). You can find definitions for the fields and controls later on this page.

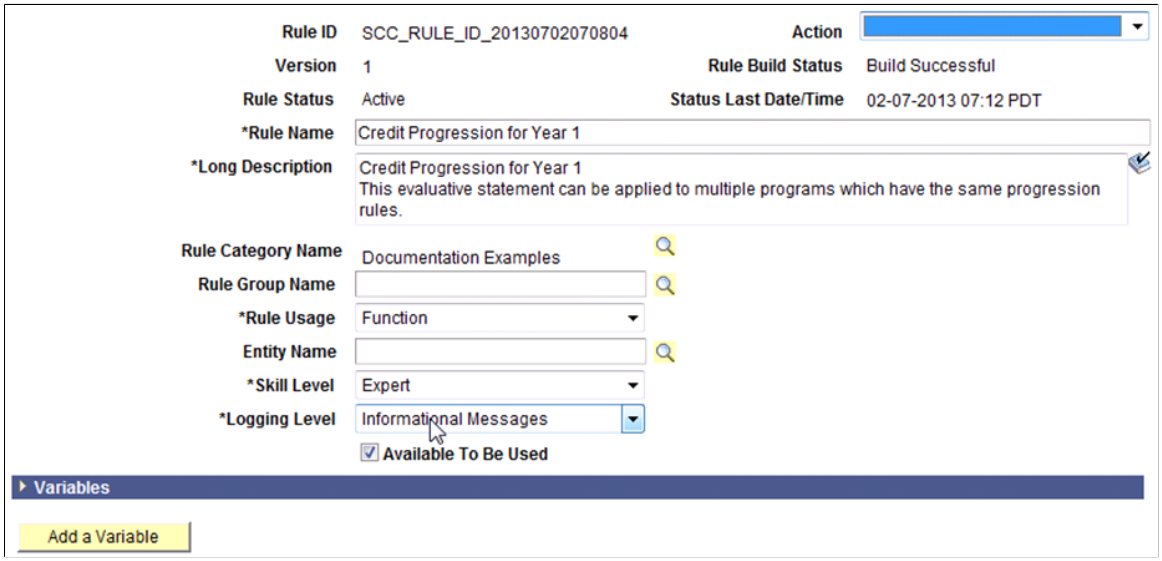
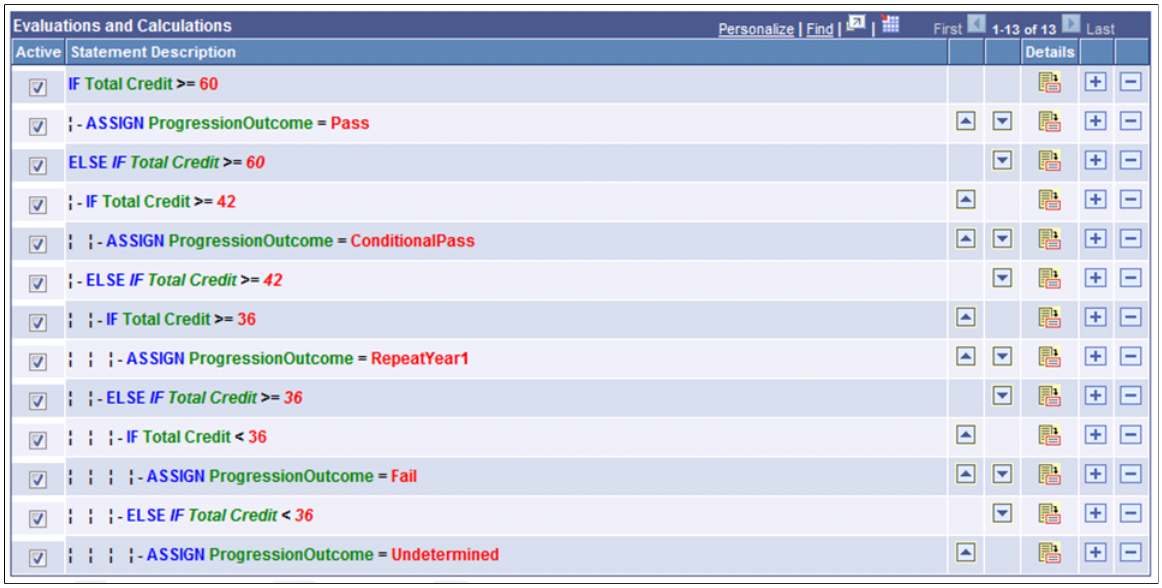


Image: Example of Academic Progression Function (2 of 2)

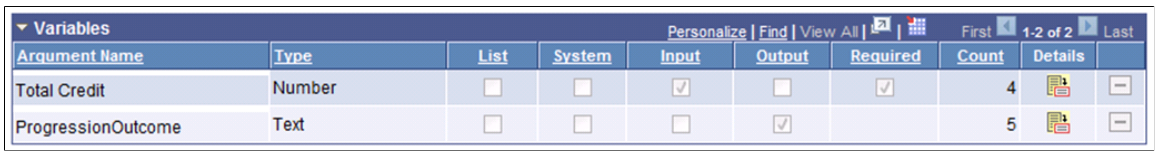
This example illustrates the fields and controls on the Example of Academic Progression Function (2 of 2). You can find definitions for the fields and controls later on this page.



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:

Image: Example of Academic Progression Function Input and Output

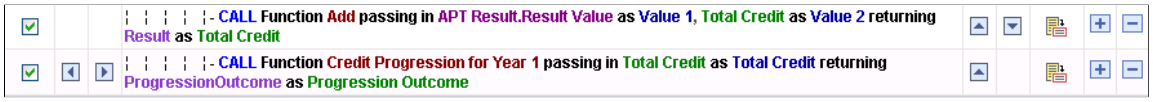
This example illustrates the fields and controls on the Example of Academic Progression Function Input and Output. You can find definitions for the fields and controls later on this page.



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:

Image: Example of Academic Progression Rule Text with Progression Function Incorrectly Indented

This example illustrates the fields and controls on the Example of Academic Progression Rule Text with Progression Function Incorrectly Indented. You can find definitions for the fields and controls later on this page.



With the CALL Function in the current position, the evaluation takes 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

- Call Function Credit progression Year 1

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

Image: Example of Academic Progression Rule Text with Progression Function Correctly Indented

This example illustrates the fields and controls on the Example of Academic Progression Rule Text with Progression Function Correctly Indented. You can find definitions for the fields and controls later on this page.

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. Selects all relevant Courses.
2. For each Course, determines whether en enrollment record exists in Year 1 of the academic program according to the APT schedule record using user defined Function.
3. Selects the appropriate result which is stored in the result record as Result Type “ECTS”.
4. Uses the result value obtained in the previous step to calculate the sum of credit using delivered math Function “Add” in Category “Math”.

5. Evaluates whether total credit is enough to progress to year 2 of the academic program using user defined Function.

Image: Example of Progression Rule for Contextual Reference Example (1 of 2)

This example illustrates the fields and controls on the Example of Progression Rule for Contextual Reference Example (1 of 2). You can find definitions for the fields and controls later on this page.

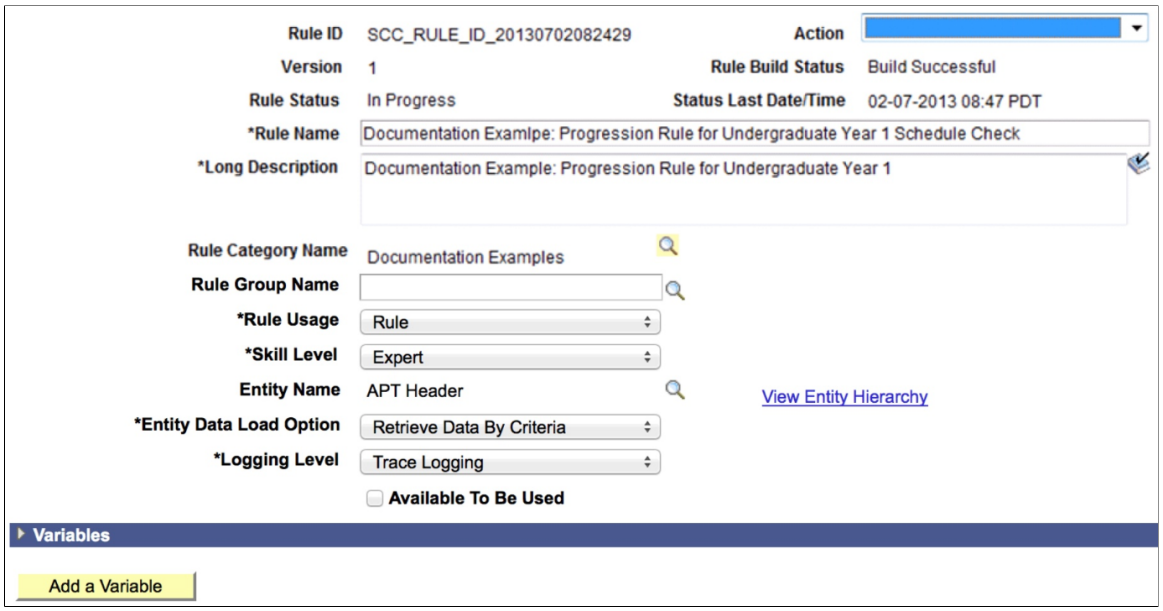
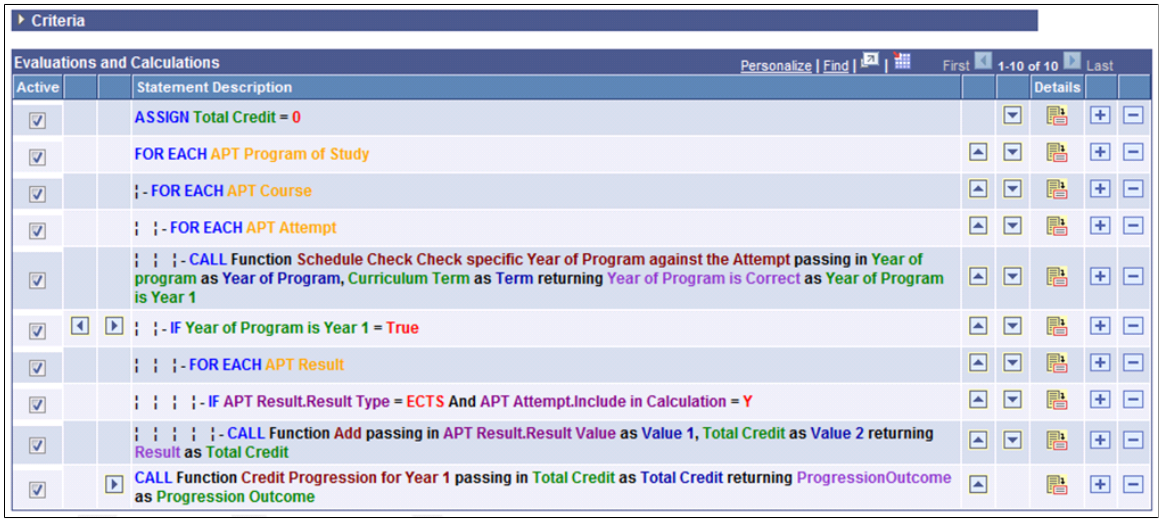


Image: Progression Rule for Contextual Reference Example (2 of 2)

This example illustrates the fields and controls on the Progression Rule for Contextual Reference Example (2 of 2). You can find definitions for the fields and controls later on this page.



The Functions used to perform summing of Total Credit (Steps 4 and 5) receive either an Input value of type Variable or Property directly from the Calling Rule. An input Variable is all that these Functions need to execute the required processing logic. 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.

Shown below is an example of the custom Function “NLD Function Credit Progression Year 1”.

Image: Example of Progression Function for Contextual Reference (1 of 2)

This example illustrates the fields and controls on the Example of Progression Function for Contextual Reference (1 of 2). You can find definitions for the fields and controls later on this page.

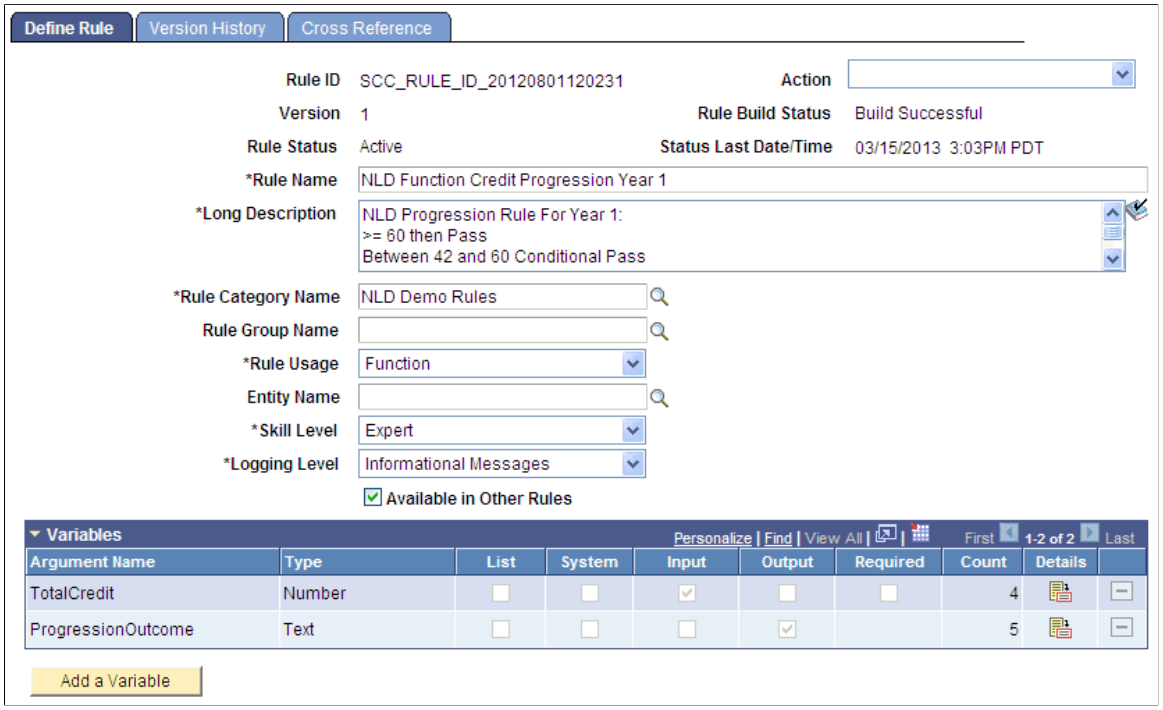
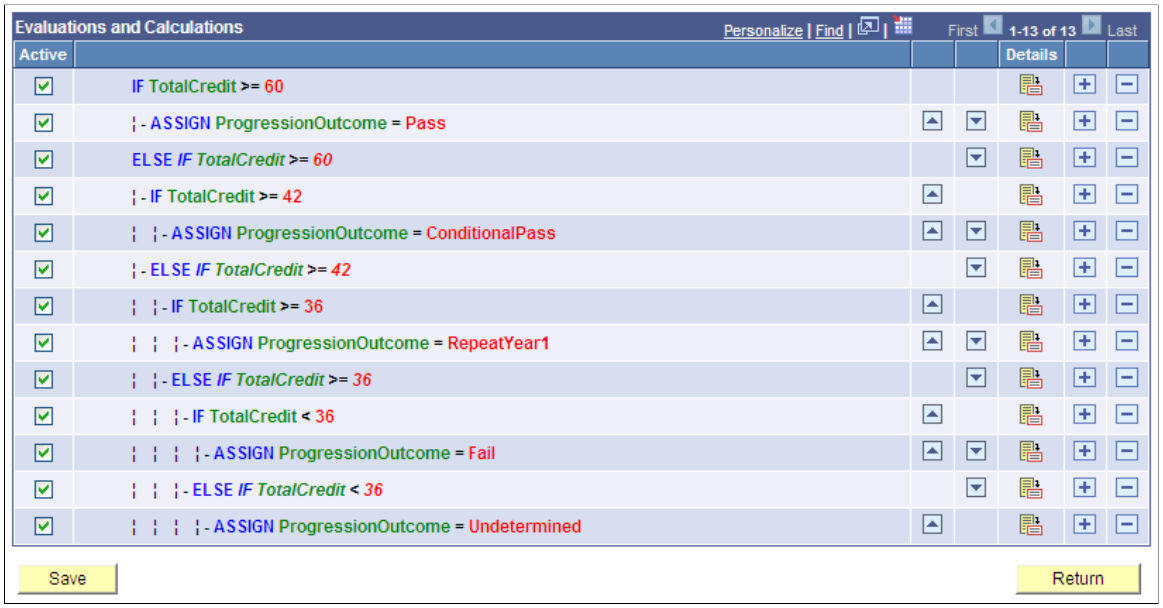


Image: Example of Progression Function for Contextual Reference (2 of 2)

This example illustrates the fields and controls on the Example of Progression Function for Contextual Reference (2 of 2). You can find definitions for the fields and controls later on this page.



Next is an example of the Function “Schedule Check, Check specific Year of Program against the Attempt”. This Function does have its own Base Entity “APT Attempt”, but only Input and Output

parameters are defined in the Call Statement shown after the Function example. The criteria identifying the exact APT Attempt are not explicitly passed from the Calling Rule to the Called Function.

However, when the Function is called, the exact Attempt 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 Attempt which is the same context that the Calling Rule is working from when retrieving Attempts using the FOR-EACH loop.

Image: Example of Program Course Function for Contextual Reference

This example illustrates the fields and controls on the Example of Program Course Function for Contextual Reference. You can find definitions for the fields and controls later on this page.

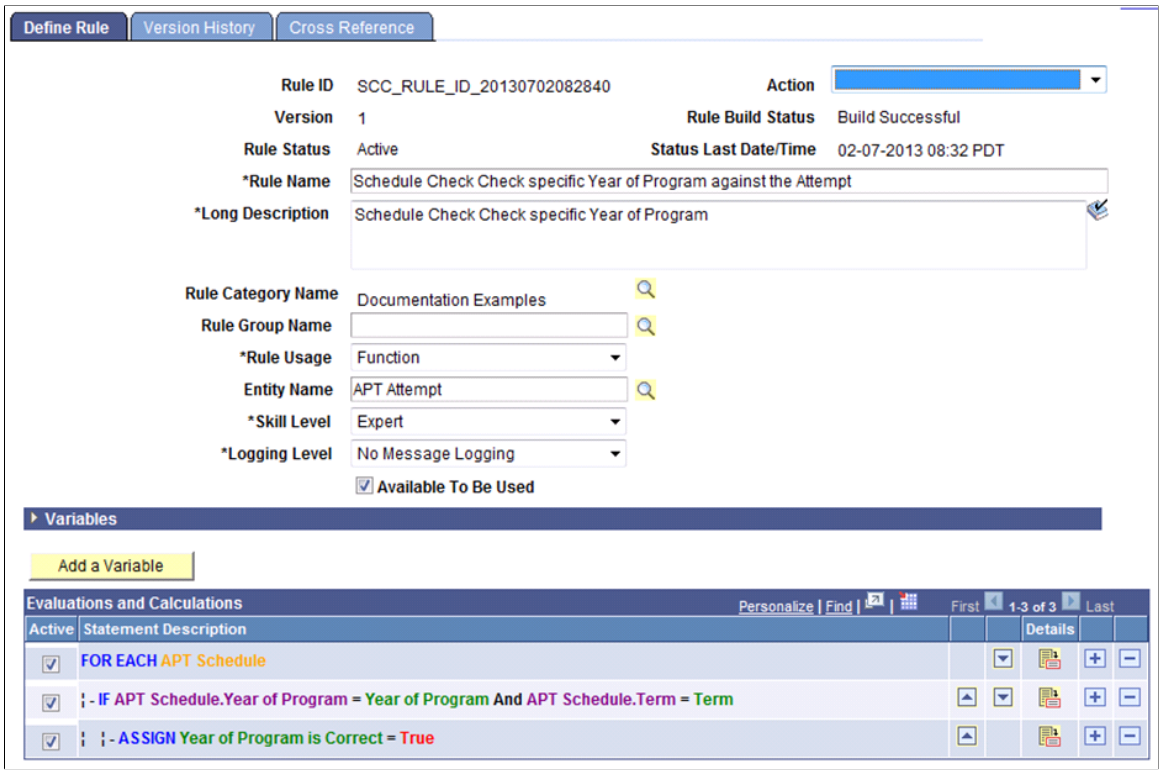
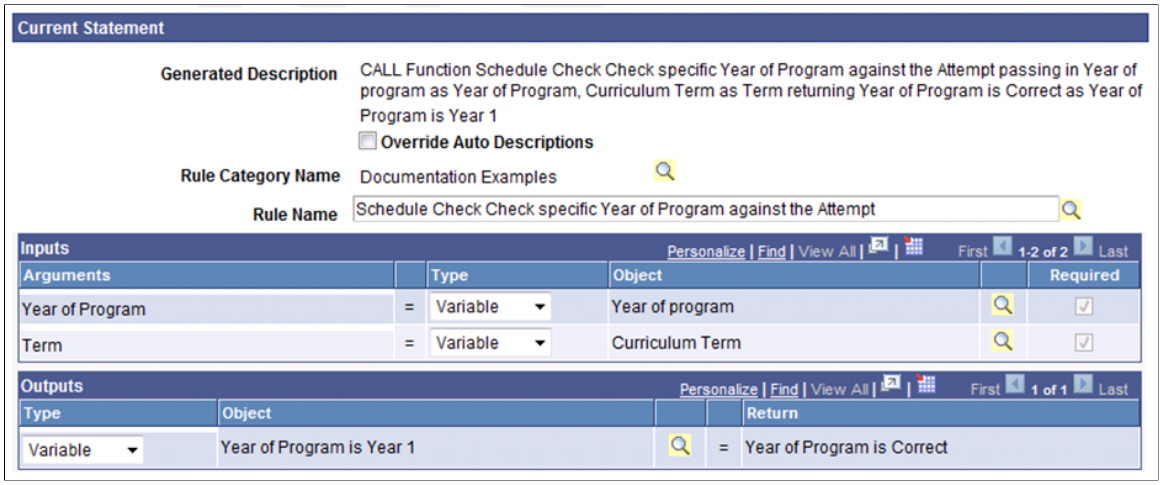


Image: Example of Call Statement for Contextual Reference

This example illustrates the fields and controls on the Example of Call Statement for Contextual Reference. You can find definitions for the fields and controls later on this page.



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.

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

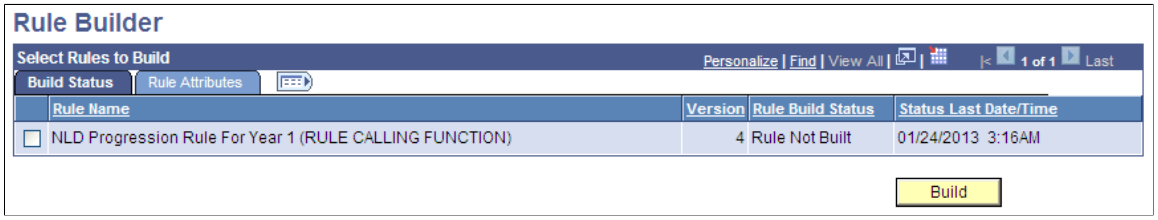
Access the Define Rule page (Set up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule, Define Rule).

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:

Image: Rule Builder Page

This example illustrates the fields and controls on the Rule Builder Page. You can find definitions for the fields and controls later on this 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).

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:

Image: Rules Engine Tester page

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

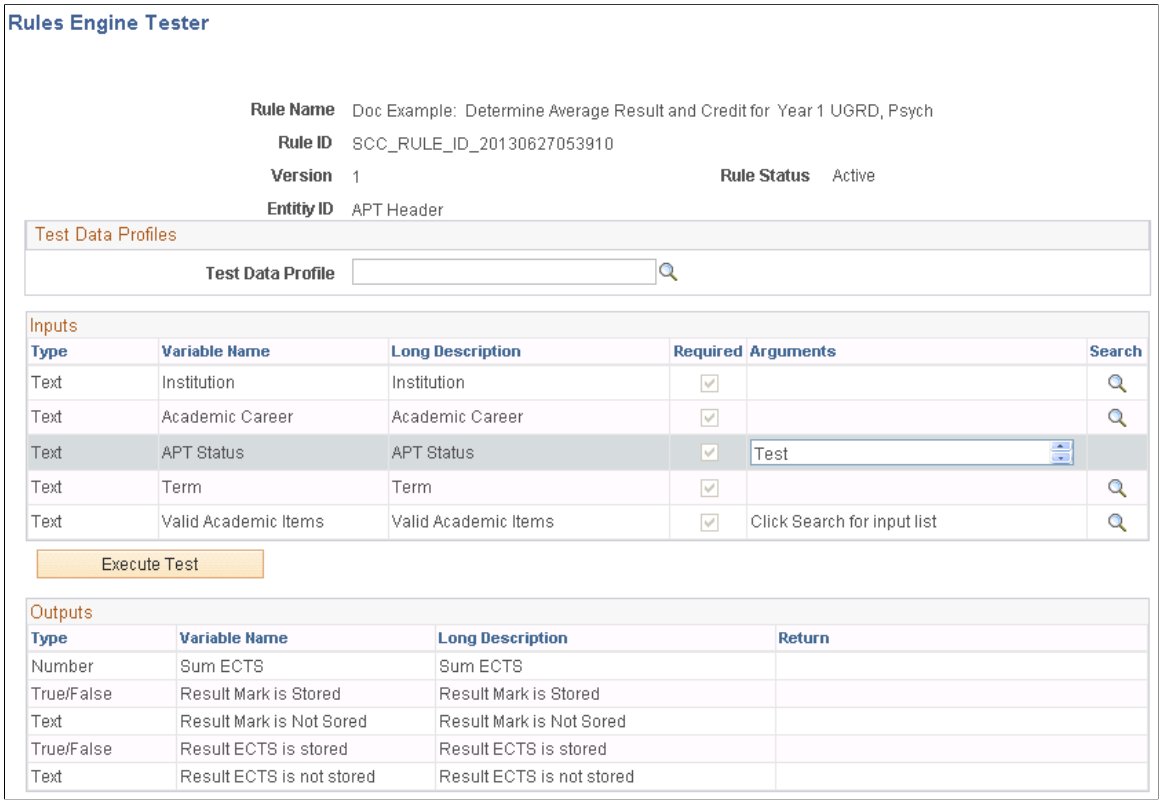


Image: Rules Engine Tester page (Inputs and Outputs)

This example illustrates the fields and controls on the Rules Engine Tester page (Inputs and Outputs). You can find definitions for the fields and controls later on this page.

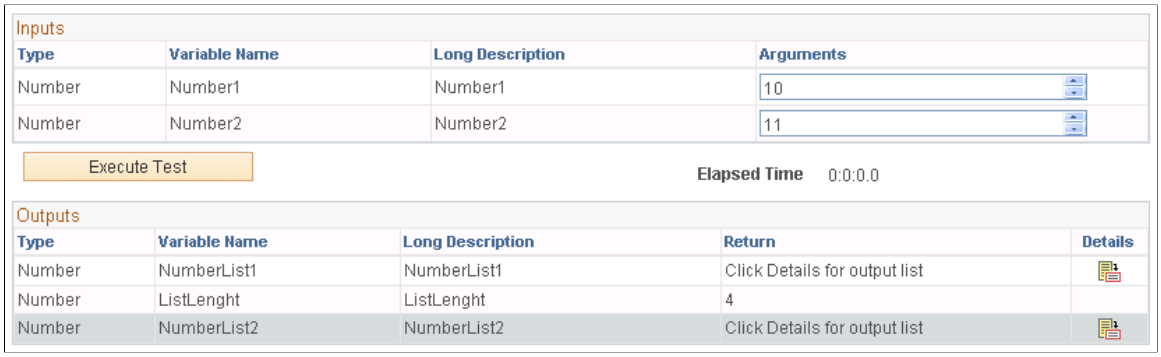
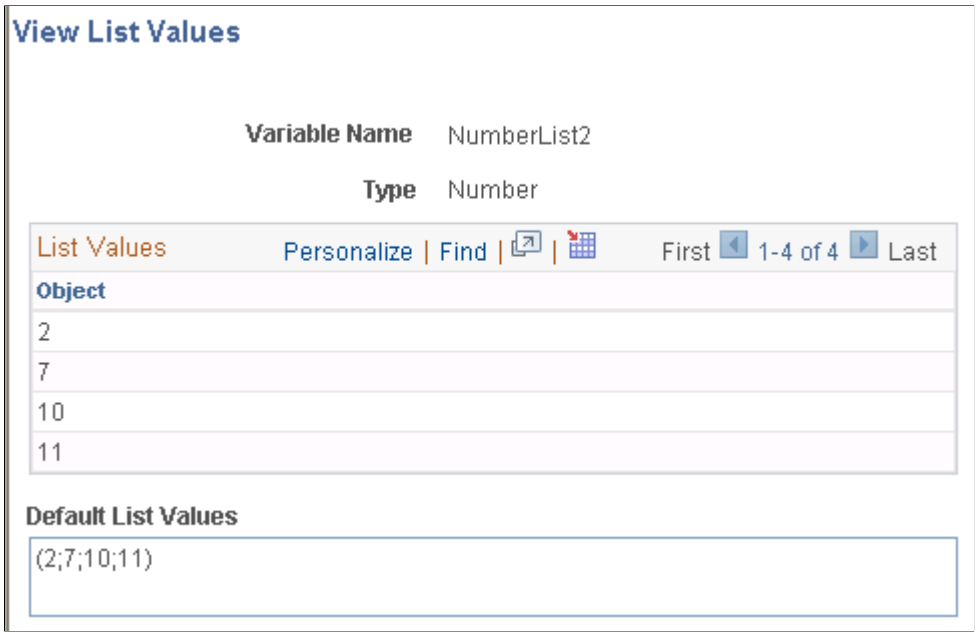


Image: Example of Rules Engine Tester List Output

This example illustrates the fields and controls on the Example of Rules Engine Tester List Output. You can find definitions for the fields and controls later on this page.



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 Profile.
Delete	Click this button to delete a stored Test Data Profile.
Update	Click this button to update a stored Test Data Profile with new parameters.
Add a New Test Data Profile and **Add**	Enter a name if you want to save a new Test Data Profile and click the Add button. A Test Data Profile can be added after the Rule has been executed using the listed parameters
Inputs	Displays all Parameters/Variables which have been listed as Input. **Note:** The text *Click Search for input list* appears next to *List* Input Variables. Click the Search icon to provide input.
Arguments	Enter the required Arguments needed to test the Rule.
Execute Test	Select this button to execute the Rule. If a Rule needs parameters and none are provided, the Rule cannot be executed.
Outputs	Displays all output Parameters/Variables with the output result when the Rule executes successfully.

Note: The text *Click details for output list* appears in the Output grid for *List* Variables only. Click on the Details icon to view *List* Variable output on a secondary page.

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.

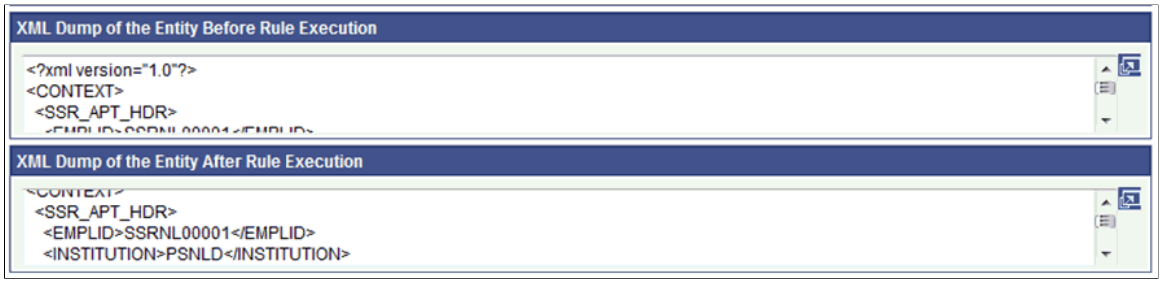
Return

Select this button to return to the Rules Engine Manager page.

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

Image: Example of XML Dump in Rule Tester

This example illustrates the fields and controls on the Example of XML Dump in Rule Tester. You can find definitions for the fields and controls later on this page.



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.

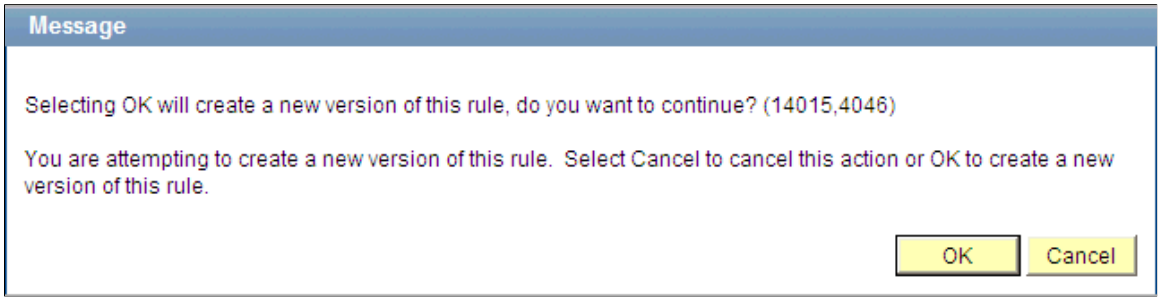
Creating a New Version of a Rule

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:

Image: Example of Create New Rule Warning Message

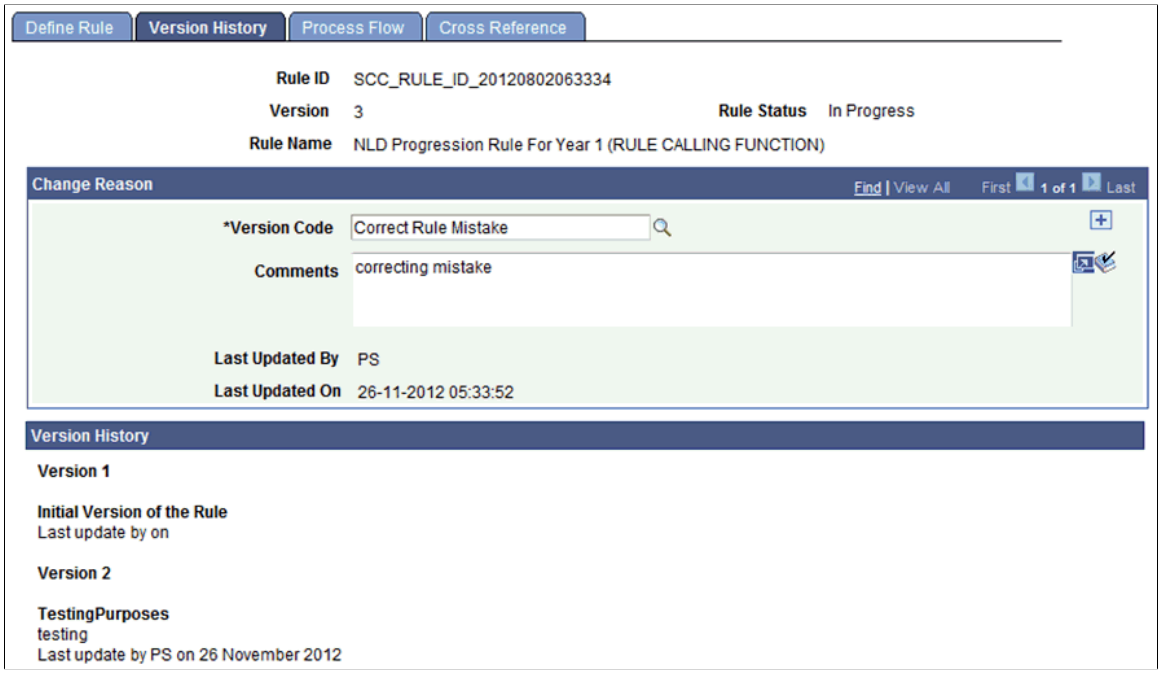
This example illustrates the fields and controls on the Example of Create New Rule Warning Message. You can find definitions for the fields and controls later on this page.



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

Image: Version History page

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



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

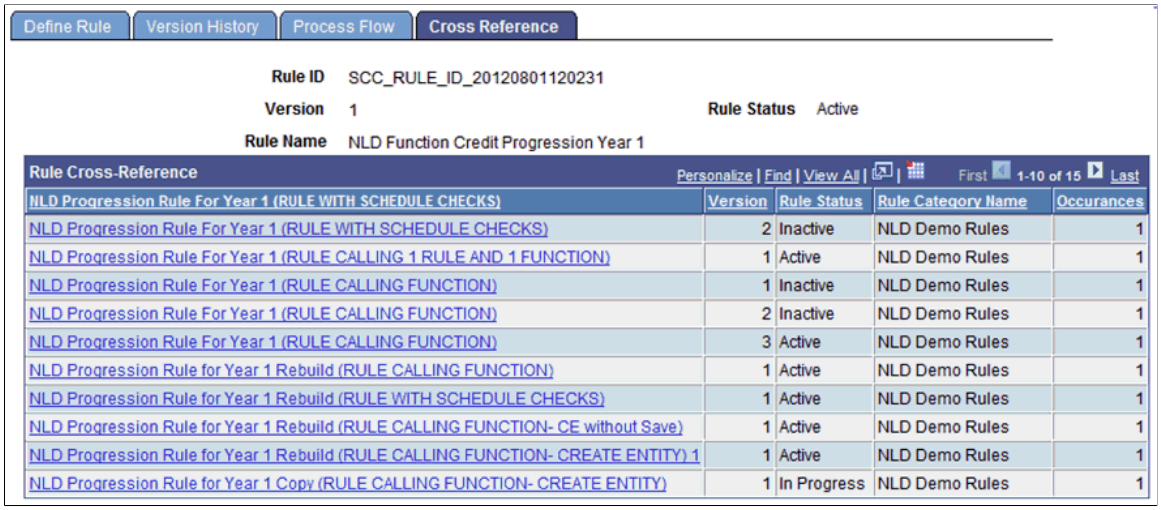
Note: It is not possible to reactivate “old” versions of Rules.

Viewing Rule Cross References

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

Image: Cross Reference page

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



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

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.”

Running Rules in Batch

Active Rules can be run in batch. Run Rules as a one off process or scheduled them to run at regular intervals.

Page Used for Running Rules in Batch

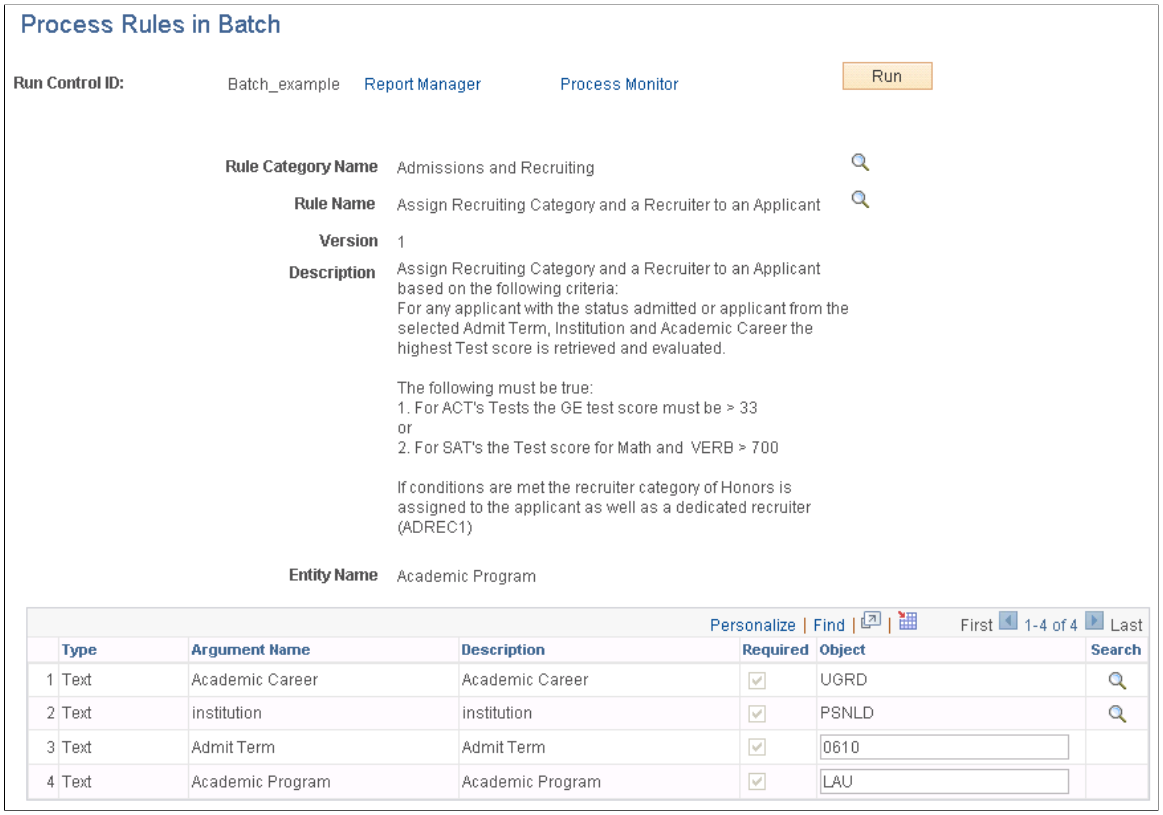
Page Name	*Definition Name*	*Navigation*	*Usage*
Process Rules in Batch	SCC_RULE_RC_BAT	Set Up SACR, System Administration, Rules Engine, Batch Processes, Run Rules in Batch	Use this page to run Active Rules in batch.

Running Rules in Batch

Access the Run Rules in Batch page (Set Up SACR, System Administration, Rules Engine, Batch Processes, Run Rules in Batch).

Image: Process Rules in Batch page

This example illustrates the fields and controls on the Process Rules in Batch page. You can find definitions for the fields and controls later on this page.



Rule Category Name Select a Rule Category. The Rule Category search is restricted to those for which you are authorized.

Rule Name Select a Rule.

Available Rules are filtered using the following criteria:

- Rule Usage is *Rule*.
- Rule Status is *Active*.
- Available to be Used is *True* (selected).

Version Displays the version of the selected rule

Description Displays the Description of the Rule, if any.

Entity Name Displays the Base Entity of the Rule, if any.

Rule Group Name Displays the Rule Group Name of the Rule, if any.

Variables

After selecting a Rule Name, the required variables are displayed in a grid. You must provide an Object input value for each required Variables before running or scheduling a Rule. If LOV prompting has been added to a Variable, the same LOV prompting functionality can be used to provide an Object input for the Variable.

Batch Process Logging

The Rules Engine Batch process generates log files for the Rule and any called Rules or Functions if the Rules and or called Functions have been built using one of these Rules Engine logging levels:

- Informational Messages
- Trace Logging
- Error Messages
- Warning Messages

With Trace Logging, three files are generated:

- RuleDebugFile – A log file with a transcript of Call Statements, Variable Maps and Contents, and Statements.
- RuleEntityPostExecution – An XML file containing a dump of the Entity structure *prior to* processing.
- RuleEntityPreExecution – An XML file containing a dump of the Entity structure *after* processing. Any updates and or inserts into the system are logged in the XML file.

Integrating User Interfaces with the Rules Engine

In addition to the system-delivered interfaces with the Rules Engine (Rules Engine Tester, Rules Engine Batch Processing, and selected user interfaces in Program Enrollment, Activity Management, Research Tracking, and Evaluation Management System), you can create custom integrations to the Rules Engine for user interfaces throughout Campus Solutions.

In this topic, we are illustrating the creation of a custom interface to the Rules Engine to meet the following business requirement:

Provide a button on component Records and Enrollment > Student Term Information > Term History > Student Special GPA that calculates a Special Grade Point Average based on courses from a student's major and displays the result on the page.

The steps to meet this requirement are:

1. Build and test a Functional Rule.
2. Generate a Trigger to call the Functional Rule.
3. Generate boilerplate code.

4. Attach the generated code to the component event.

Building and Testing a Functional Rule

The first step is to build and the test a Functional Rule to integrate with Student Special GPA page; in this example, a Rule that calculates a Special Grade Point Average based on a student's Academic Program, Academic Plan, or Academic Sub-Plan. Using the functional business requirement as a starting point, start by looking at how to retrieve the needed data.

Data and Entity Considerations

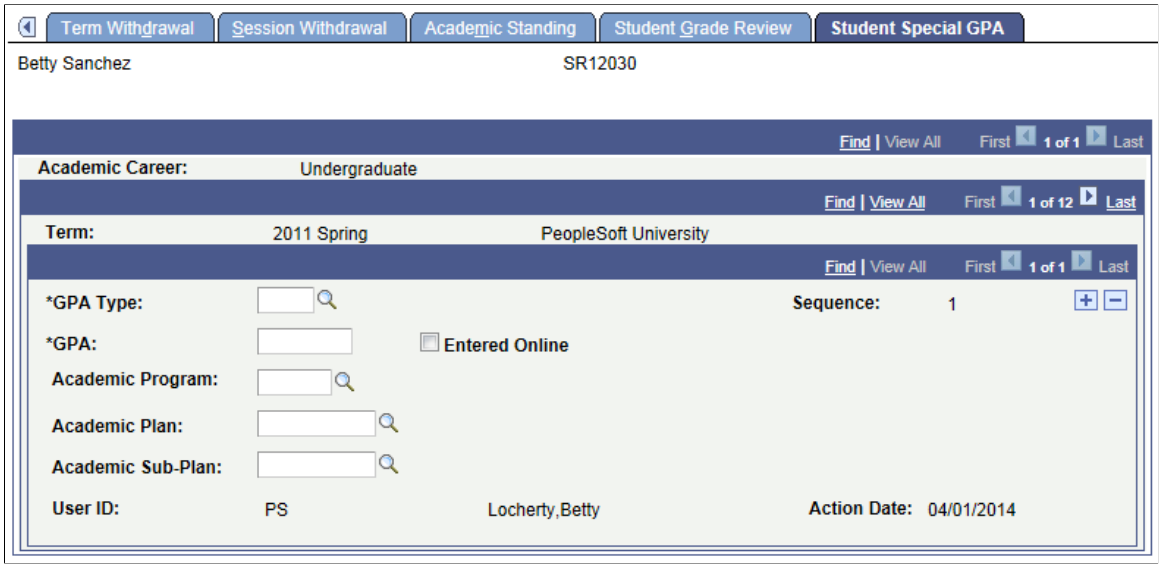
To determine how to retrieve the data, decide which role to use to build the Rule: Functional Expert Role or Developer Role. This example uses the Functional Expert Role instead of the Developer Role to create an *Expert Rule*. This assumes that Functional Experts want to create similar grade point average calculation Rules at a later stage and/or adjust the business logic for the created Rule over time. Rules created by a Developer Role are created in Application Package PeopleCode by developers. So, while Functional Experts are able to use such Rules, they have no means to adjust their business logic. By creating an Expert Rule, Functional Experts are able to clone or adjust the Rule.

Since this is an Expert Rule, an Entity is required to be in place for this data. For this example, system-delivered Entities are used, but new Entities can be created if needed. For information on creating Entities, see Creating a New Entity.

In this example, the Student Grade Inquiry page (Records and Enrollment > Student Term Information > Student Grades) is based on the record view CLASS_TBL_SE_VW, which contains all the information needed for this Rule. Since record tables are not being updated with this Rule, a record view is suitable.

Image: Student Special GPA page before Customization for Rule Engine User Interface Example

This example illustrates the Student Special GPA page before a custom integration with the Rules Engine.

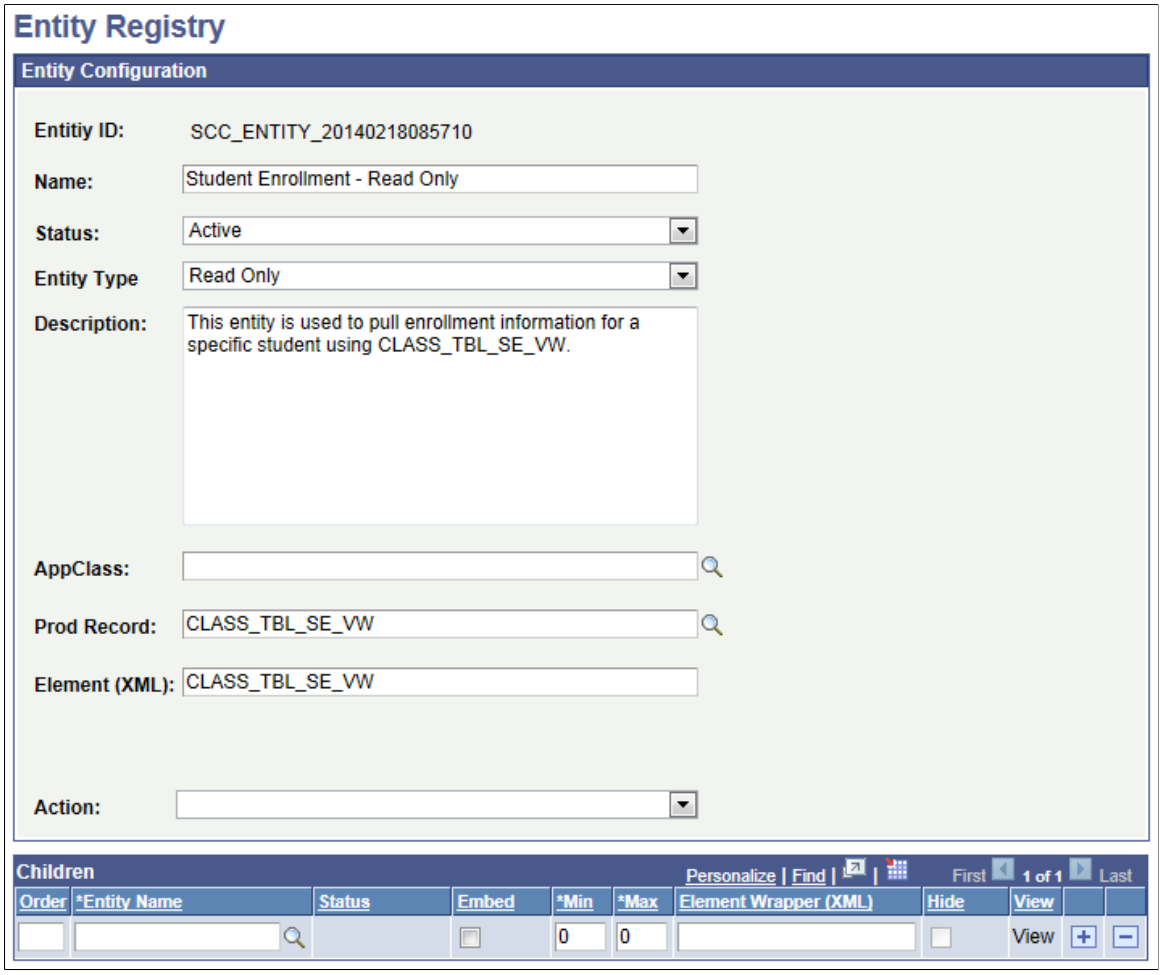


Navigate to the Entity Registry (Set Up SACR, System Administration, Entity , Entity Registry) to find that there is an existing matching Entity, *Student Enrollment – Read Only*, for record view CLASS_TBL_SE_VW. For this Entity, Entity Type is set to *Read Only* for Prod Record *CLASS_TBL_SE_VW*. The *Read Only* Entity Type does not allow an update or save to be executed by the Rules Engine. The Entity Status is set to *Active*. No AppClass (application class) is needed for Read Only

Entities. The AppClass is reserved for Application Class PeopleCode which controls the save Validation logic for the Entity. Use the Action drop down and select *View Hierarchy* to view the Entity structure and its properties.

Image: Entity Registry page for Student Enrollment - Read Only for Rules Engine User Interface Example

This example illustrates the Entity Registry page for the Student Enrollment – Read Only Entity.



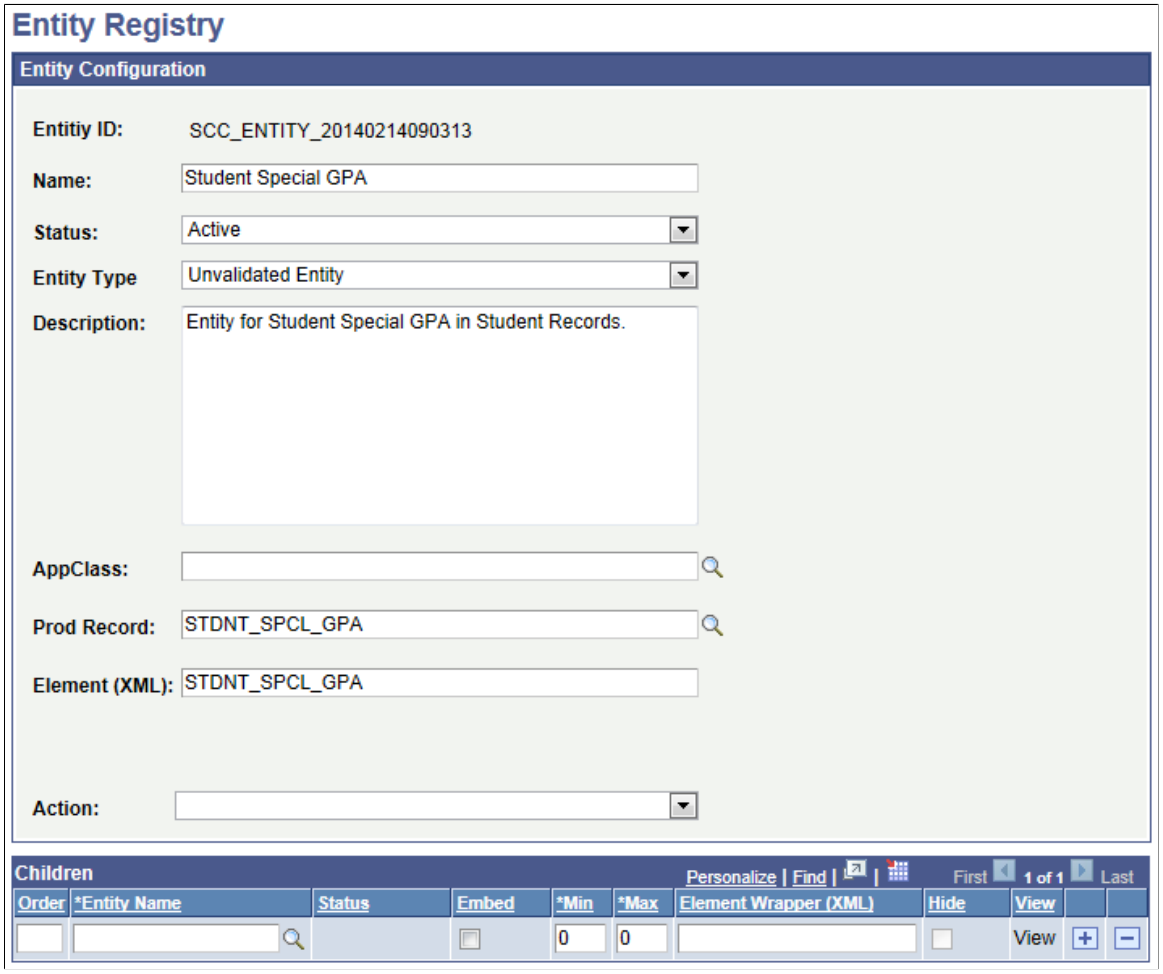
Although this Rule is not updating CLASS_TB_SE_VW, it does access and update the calculated Special GPA. The production record for Special GPA is STDNT_SPCL_GPA.

Navigate to the Entity Registry (Set Up SACR, System Administration, Entity , Entity Registry) to find that there is an existing matching Entity, *Student Special GPA*, for production record STDNT_SPCL_GPA. For this Entity, Entity Type is set to *Unvalidated Entity* for Prod Record *STDNT_SPCL_GPA*. The *Unvalidated Entity* Type allows data to be inserted and updated with *no* validation. The Entity Status is set to *Active*. No AppClass (application class) is needed for Unvalidated Entities. The AppClass is reserved for Application Class PeopleCode which controls the save Validation

logic for the Entity. Use the Action drop down and select *View Hierarchy* to view the Entity structure and its properties.

Image: Entity Registry page for Student Special GPA for Rules Engine User Interface Example

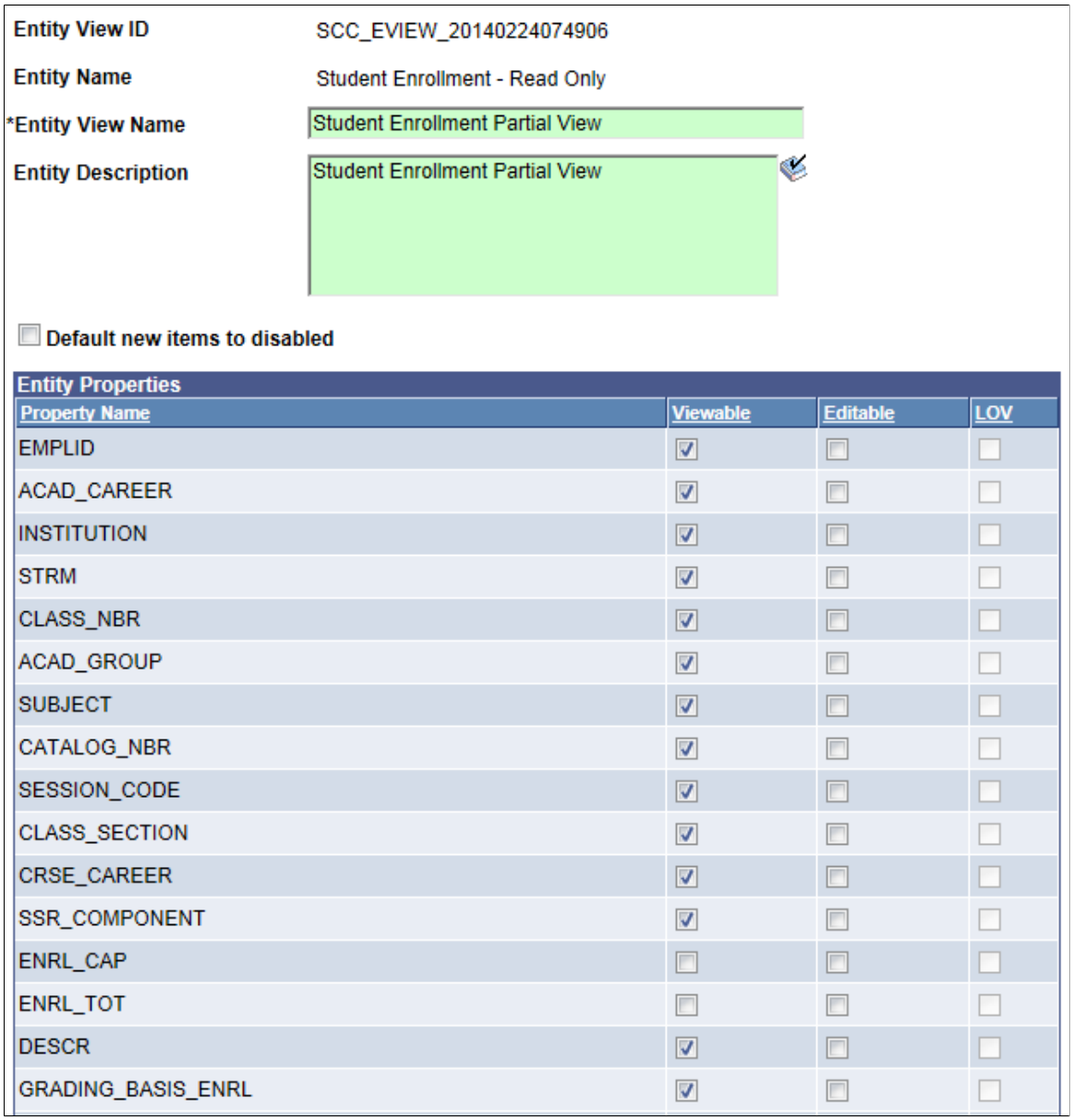
This example illustrates the Entity Registry page for the Student Special GPA Entity.



If you want to limit access to only certain properties of any Entity that you are using, you can do so by creating an Entity View. Do this by selecting *Entity View* from the Action drop down on the Entity Registry page.

Image: Entity View Editing page for the Student Enrollment Partial View of the Student Enrollment – Read Only Entity for Rules Engine Interface Example

This example illustrates the Entity View Editing page for the Student Enrollment Partial View of the Student Enrollment – Read Only Entity.



Entity View ID SCC_EVIEW_20140224074906

Entity Name Student Enrollment - Read Only

***Entity View Name** Student Enrollment Partial View

Entity Description Student Enrollment Partial View

☐ **Default new items to disabled**

Entity Properties

Property Name	Viewable	Editable	LOV
EMPLID	☑	☐	☐
ACAD_CAREER	☑	☐	☐
INSTITUTION	☑	☐	☐
STRM	☑	☐	☐
CLASS_NBR	☑	☐	☐
ACAD_GROUP	☑	☐	☐
SUBJECT	☑	☐	☐
CATALOG_NBR	☑	☐	☐
SESSION_CODE	☑	☐	☐
CLASS_SECTION	☑	☐	☐
CRSE_CAREER	☑	☐	☐
SSR_COMPONENT	☑	☐	☐
ENRL_CAP	☐	☐	☐
ENRL_TOT	☐	☐	☐
DESCR	☑	☐	☐
GRADING_BASIS_ENRL	☑	☐	☐

Provide the Entity View with a name and deselect the property options as required:

- Viewable – The property can be viewed by not changed.
- Editable – The property is changed when saved.

Note: When a new Entity is created from a production record and saved, the Entity Registry is automatically updated. The record fields become available as Entity Properties and existing Yes/No, translate value, and prompt table validations on the RECORD.FIELD definition are made available for prompting without specifically defining a LOV. .

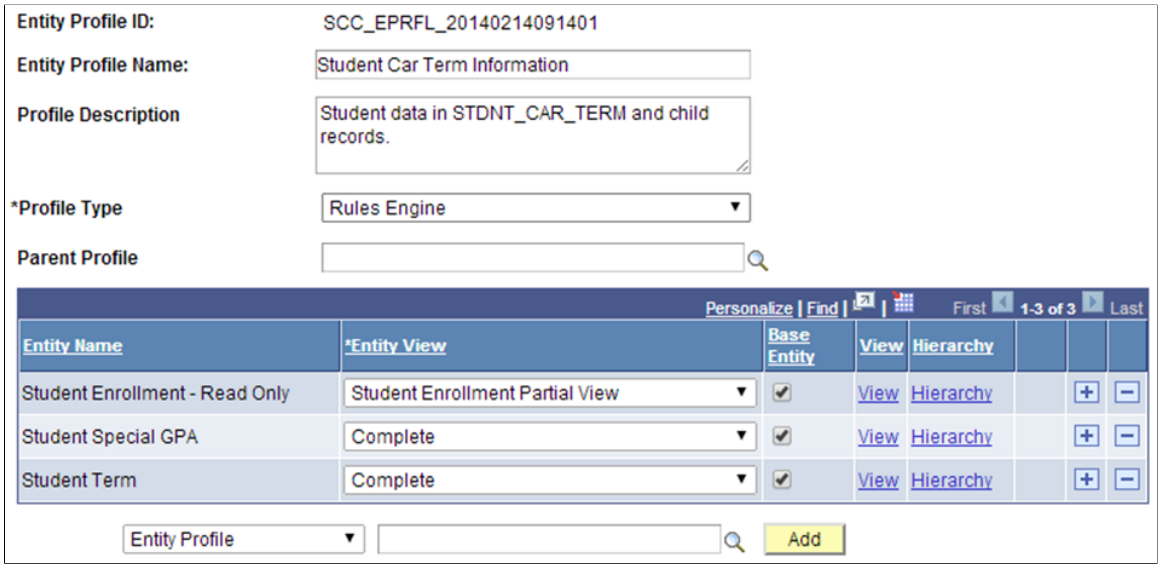
Providing Entity Security Access

Before an Entity can be used in the Rules Engine, you must create the appropriate security setup. This is done using the Entity Profile and Define Rule Category setups.

The Entity Profile controls how the Entity can be accessed by selection of a Profile Type. In this example, *Rules Engine*. The Entity Profile also controls whether the Rule is allowed to use the Entity Structure as a starting point for data access in the Rule. This is controlled by the selecting *Base Entity* for each Entity Name in the Entity Profile to which you want to have access.

Image: Entity Profile page for Student Car Term Information for Rules Engine User Interface Example

This example illustrates the Entity Profile page for Student Car Term Information for the Rules Engine User Interface Example.



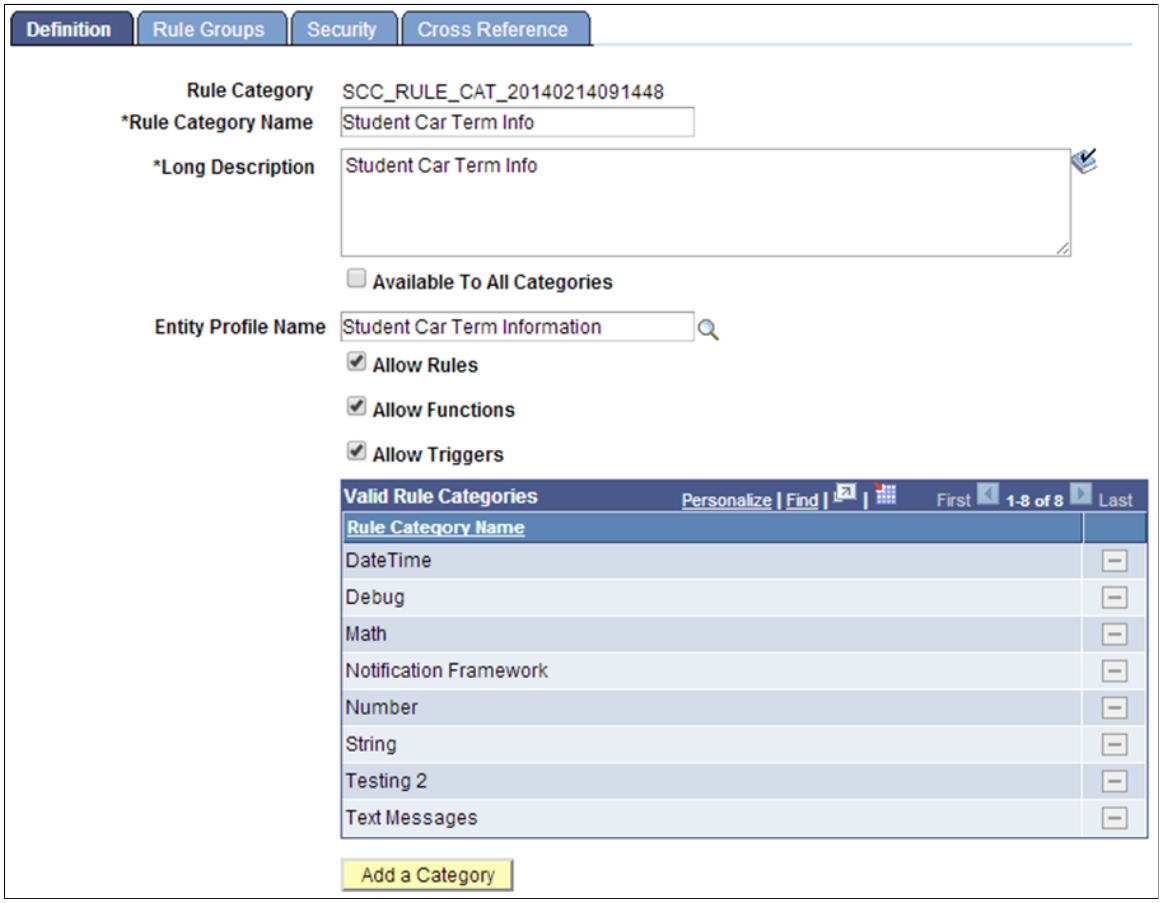
The next part of Entity Security is setup using Rule Category Definition. Here, you can:

- Create a Category in which to create the Functional Rule. In this example, we want all Student Car Term related Rules created in one Rule Category.
- Grant access to the Entity Profile *Student Car Term Information.*
- Control access to types of Rules. we want to create Functions, Rules and Triggers.
- Control which from Valid Rule Categories business logic can be accessed. We are creating a Rule that uses business logic from other Rule Categories; for example, to perform calculations (category Math) .

- Control who can create Rules for Student Car Term Information; this may be one or more specified users or all users in associated Role.

Image: Define Categories page for Rules Engine User Interface Integration Example

This example illustrates the Define Categories page for Rules Engine User Interface Integration Example.



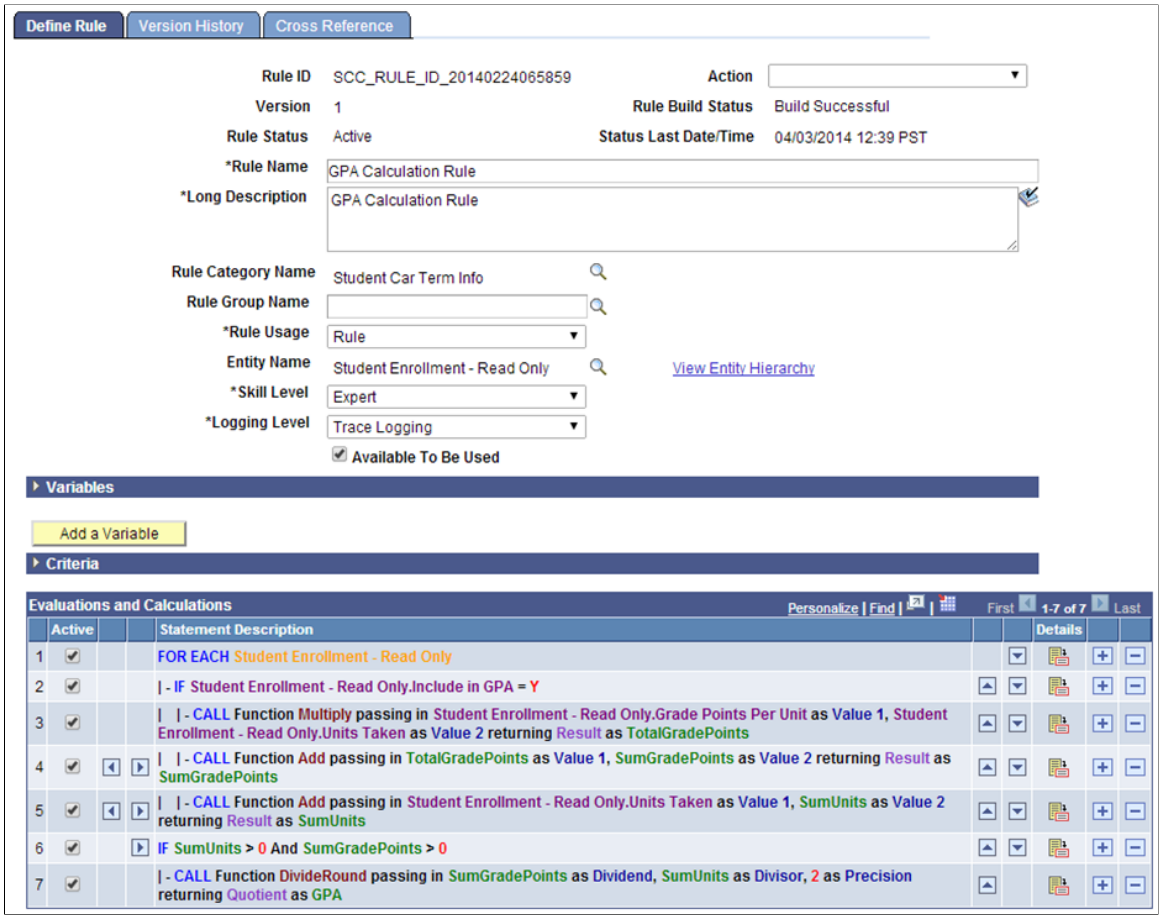
Build and Test the Functional Rule

With Entity security in place, t the Functional Rule to calculate the Special GPA for our business case can be created and tested.

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

Image: Define Rule page for Functional Rule for Rules Engine User Interface Example

This example illustrates the Define Rule page for Functional Rule for Rules Engine User Interface Example



This Functional Rule uses the *Student Enrollment – Read Only* Entity, created from production record CLASS_TBL_SE_VW. Assumed is the selection of data using the following Criteria:

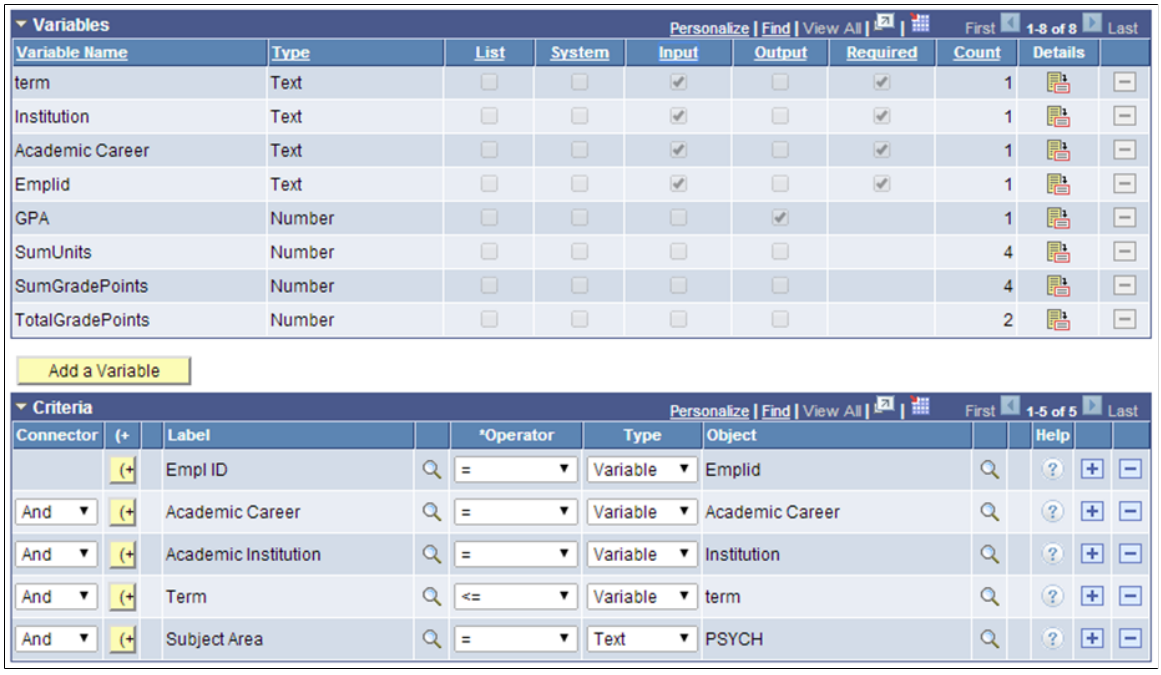
- EmplID
- Academic Career
- Institution
- Term
- Subject

Using these Criteria, the Rule returns all Psychology class results for the specified Student ID, Academic Career, Institution and those Academic Terms which are less than or equal to the specified Term. The

Criteria need to be passed into this Rule. The needed Values are obtained from Variables designated as required input:

Image: Variables and Criteria for Define Rule Page for Functional Rule for Rules Engine User Interface Example

This example illustrates the Variables and Criteria for Define Rule Page for Functional Rule for Rules Engine User Interface Example.



Variables — Personalize | Find | View All | First 1-8 of 8 Last

Variable Name	Type	List	System	Input	Output	Required	Count	Details
term	Text			✓		✓	1	
Institution	Text			✓		✓	1	
Academic Career	Text			✓		✓	1	
Emplid	Text			✓		✓	1	
GPA	Number				✓		1	
SumUnits	Number						4	
SumGradePoints	Number						4	
TotalGradePoints	Number						2	

Add a Variable

Criteria — Personalize | Find | View All | First 1-5 of 5 Last

Connector	(+	Label	*Operator	Type	Object	Help
	(+	Empl ID	=	Variable	Emplid	
And	(+	Academic Career	=	Variable	Academic Career	
And	(+	Academic Institution	=	Variable	Institution	
And	(+	Term	<=	Variable	term	
And	(+	Subject Area	=	Text	PSYCH	

Test the Rule (select *Test Rule* from the Action drop down on the Define Rule page) to ensure it is producing the desired results before proceeding with the next step of generating a Trigger to call the Rule.

Create a Trigger to Call the Functional Rule

Note: This example describes the creation of a Functional Rule and a separate Trigger Rule. However, the separation of business logic and user interface integration, although beneficial, is not strictly necessary. You may create one Rule that performs both functions.

The Trigger is the connection between the user interface and the Functional Rule. The Trigger calls the Functional Rule and passes the information from the user interface to the Functional Rule. Using a Trigger Rule allows you to separate the functional business logic (in this case the GPA calculation) from the logic which dictates when and how that Functional Rule is called. Over time, the Functional Rule may change due to changes in the business case whereas the Trigger Rule controls the interaction with the user interface and the how and when of calling the Functional Rule.

In this example, the Trigger must pass the specified input values (EmplID, Academic Career, Institution, Term, and Subject area) from the user interface to the Functional Rule to successfully select Grade Points and Units from the Student Enrollment Entity and calculate the GPA. A Component Event using PeopleCode (in this example, clicking a button) calls the Trigger which then calls the Functional Rule. This is discussed further in Generate Boilerplate Code.

There are two approaches to creating a Trigger in this example:

- Method A: Pass the needed Variables to the Functional Rule by retrieving and passing the Entity to the Rule.

  Considerations for Using Method A

  - The Rule Usage is Function or Trigger and has a Base Entity. A Function or Trigger does not have criteria. This means that for this Function to work with the Base Entity attached, it needs Entity information obtained by the calling Rule or the Component Event PeopleCode which calls the Trigger or Function.
  - There is a need to act upon multiple sections of a flexible Entity Structure (for example Entities for Program Enrollment) or Entity Tree.
  - You do not want to lock-down the Input Variables for the Trigger or Function as the data needed may change over time.
  - The execution of the Rule depends on the data in the component, which may not be the same as the data saved in the database. Any changes in the Rule update the component from which the data is retrieved and does not save the data to the database directly.

- Method B: Pass the needed Variables to the Functional Rule using Variables taking their values from the user interface.

  Considerations for Using Method B

  - The Rule Usage is Trigger so the component needs to pass input Variables to a called Rule.
  - The Trigger Rule should not have a base entity defined.
  - The Input Variables are not expected to change over time.
  - Any data needed by the rules and not passed in using input Variables must be retrieved in a rule called from the trigger rule as the base entity defined by the criteria of a rule.
  - If the Rule changes values in the component, the data must be returned to the component as output Variables.

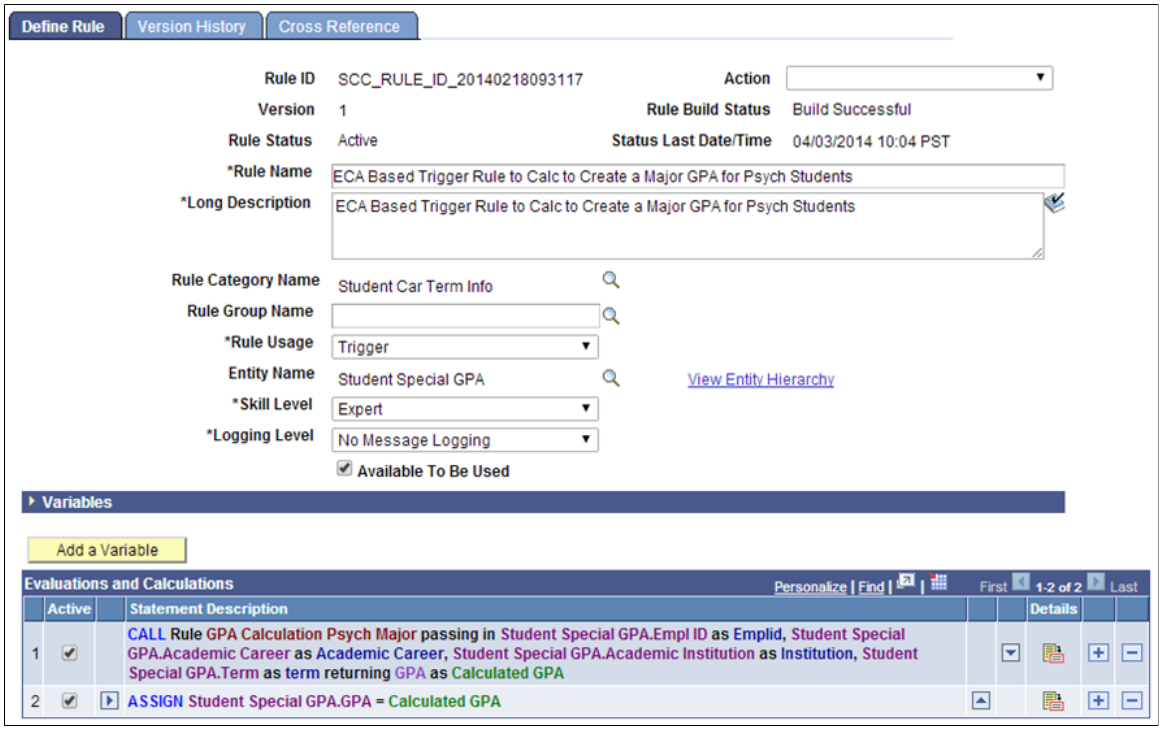
Creating a Trigger Using Method A: Passing an Entity using an Entity

In Method A, the Trigger is defined to work with the Student Special GPA Entity as a Base Entity. The production record for this Base Entity, STDNT_SPCL_GPA, is available on the user interface from

which the Trigger is called. The needed information from the Entity can be passed directly to the Trigger. Although the Trigger has no Criteria, the correct Entity is provided to the Trigger.

Image: Define Rule page for Creating Method A Trigger for Rules Engine User Interface Integration Example

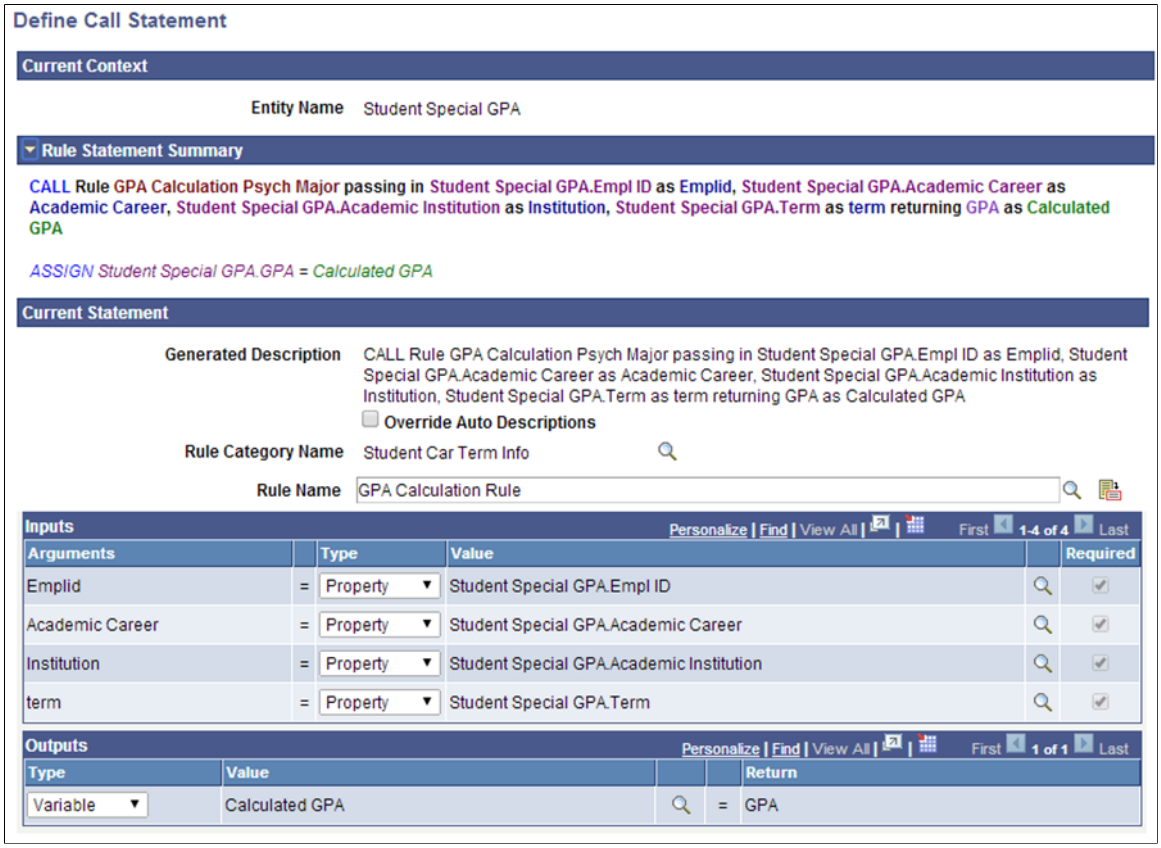
This example illustrates the Define Rule page for Creating Method A Trigger for Rules Engine User Interface Integration Example.



This Trigger updates the Student Special GPA Entity with the calculated GPA. This update is displayed on the Special GPA page. This is the Define Call Statement to the GPA Calculation Rule:

Image: Define Call Statement page for Creating Method A Trigger for Rules Engine User Interface Integration Example

This example illustrates the Define Call Statement page for Creating Method A Trigger for Rules Engine User Interface Integration Example



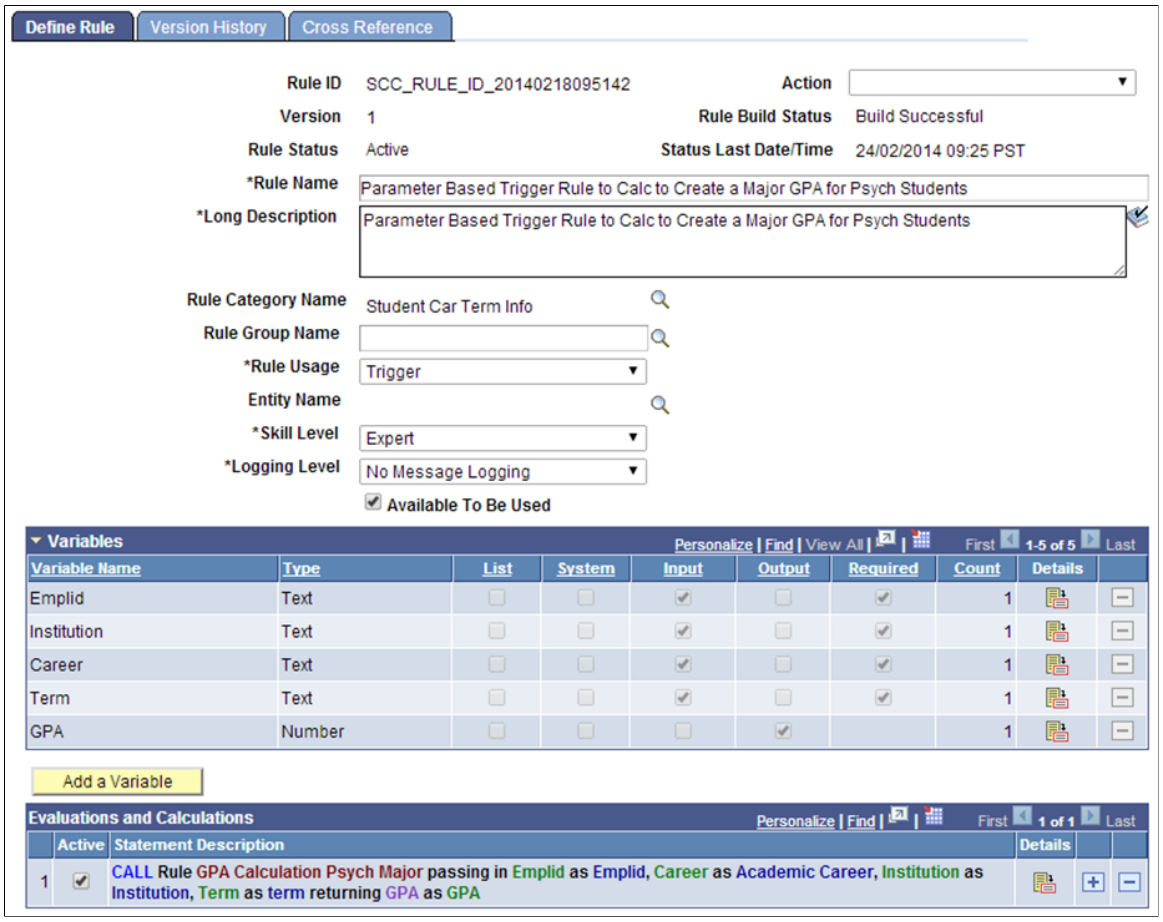
From the call to the Functional GPA rule, it is apparent that the input values needed are passed in using Entity Properties from Student Special GPA Entity.

Creating a Trigger Using Method B: Passing Parameters

In Method B, the Trigger is defined to receive all input values as Variables from the user interface. There is no Base Entity as it is not needed to provide the called GPA Functional Rule with the values it needs.

Image: Define Rule page for Creating Method B Trigger for Rules Engine User Interface Integration Example

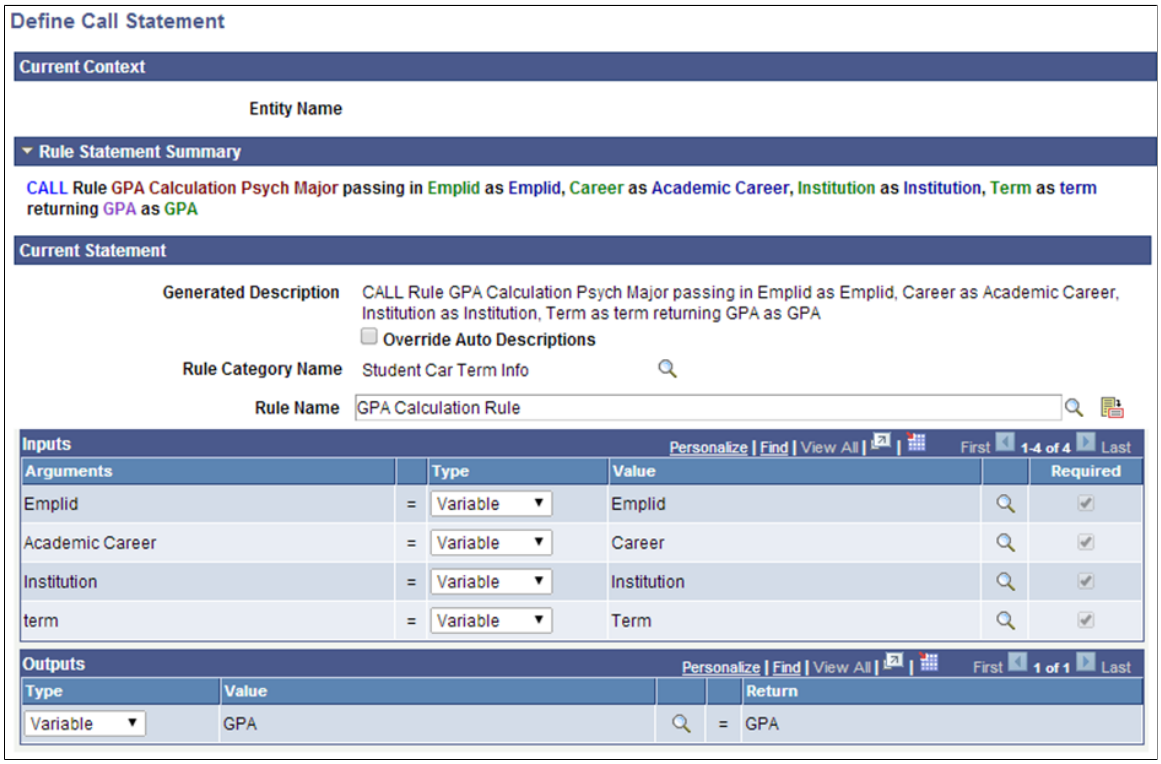
This example illustrates the Define Rule page for Creating Method B Trigger for Rules Engine User Interface Integration Example



The calculated GPA is passed into a GPA output Variable. The calling PeopleCode event takes this value and updates the GPA on the Special GPA page. This is the Define Call Statement to the GPA Calculation Rule:

Image: Define Call Statement page for Creating Method B Trigger for Rules Engine User Interface Integration Example

This example illustrates the Define Call Statement page for Creating Method B Trigger for Rules Engine User Interface Integration Example.



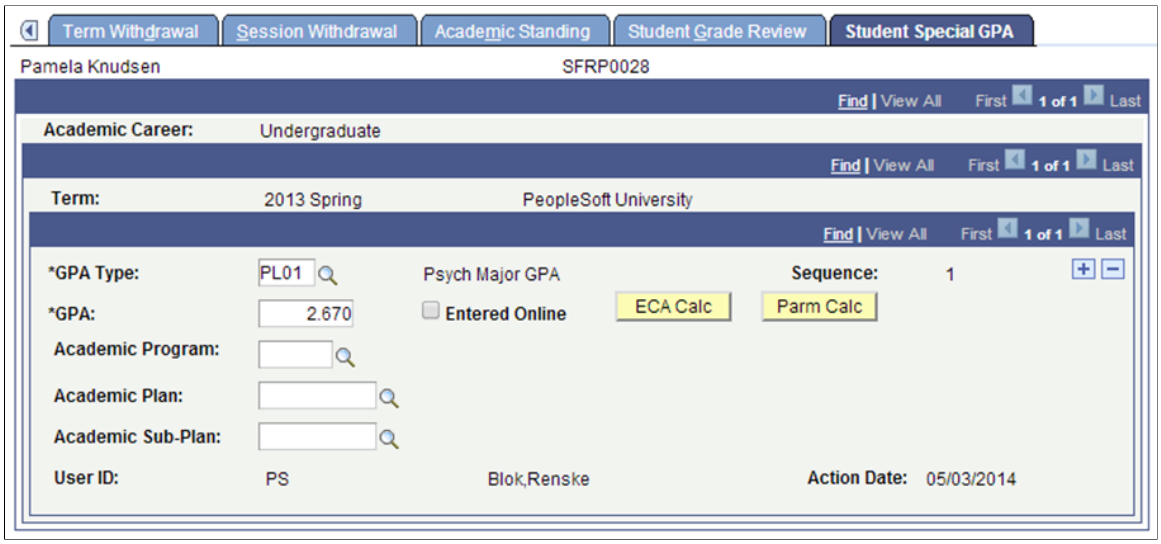
Generate Boilerplate Code

With a Trigger created, the boilerplate code can be generated and connected it to a Component Event. In this example, the code calls the Rule to calculate the GPA by clicking a button on the Student Special GPA page. The example shows two new buttons added: ECA Calc, representing the Method A approach

to the Trigger, and Parm Calc, representing the Method B approach to the Trigger. You would likely only use one approach in the customization of your system and would, therefore, only have one button.

Image: Student Special GPA page after Customization for Rule Engine User Interface Example

This example illustrates the Student Special GPA page after Customization for Rule Engine User Interface Example.



The act of clicking a button is the *FieldChange* Event to which the PeopleCode should be added to call the Trigger.

Use the Define Rule Triggers page (Set up SACR > System Administration > Rules Engine > Setup > Define Rule Triggers) to generate boilerplate code and attach that code to the Component Event. The setup created here also acts as an administrative reference to Events and associated Rules. You must minimally specify a PeopleCode Event, a Record, and a Field. The following PeopleCode Events can be selected:

- FieldChange
- FieldEdit
- SavePostChange
- SavePreChange

You may also add a Component for reference. Once you have the setup, click the Generate Code button to generate the boilerplate PeopleCode. The Generate Code option uses information from the Trigger to create a Code example which can be used to call the Trigger from the Component.

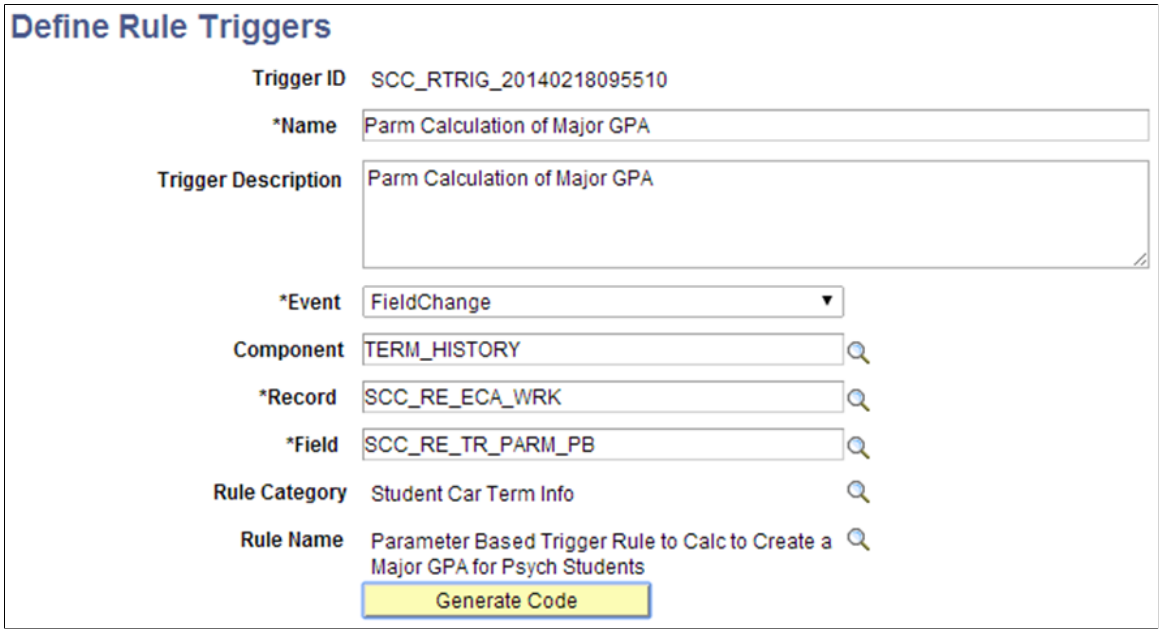
Generating Code for the ECA Calc

This example shows a Trigger based on Method A, which contains an Entity and assumes that an Entity is passed to the Trigger. The calculated GPA is updated directly on the Entity.

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

Image: Define Rule Triggers page for Parm Calc for Rules Engine User Interface Integration Example

This example illustrates the Define Rule Triggers page for Parm Calc for Rules Engine User Interface Integration Example.



Here is the example generated code:

```
* ---------------------  About This Example Code  -------------------------- */
/* The following Example Code has been created as a starting point from       */
/* which to create Rule -UI Integration.  In the code you will find that the  */
/* rule refers to the Trigger Rule provided on the component as well as its   */
/* Arguments.  The code below will need to be adjusted before it can be used  */
/* to call the (Trigger) Rule.  For more information please refer to the      */
/* PeopleBooks documentation available for Rules Engine Functionality.        */
/*                                                                            */
/* ----------------  Understanding this Example Code  ----------------------- */
/* There are two ways to bring information from a Page or Component into a    */
/* Rule:                                                                      */
/*                                                                            */
/* Method 1: Providing Arguments from Rowset                                  */
/* Provide Local Strings with Values obtained from the appropriate Component  */
/* Rowset.  This method may be appropriate when calling a Rule or Function    */
/* without an Entity which requires a set of input Arguments (Variables) or   */
/* for a Rule with a Base Entity which requires a set of parameters which can */
/* be easily obtained and passed into the Rule.                               */
/*                                                                            */
/* Method 2: Providing Entity using EntityComponentAdapter                    */
/* Use the Entity Component Adapter to obtain Entity values from the          */
/* Component.  This method should be used when calling a Function with a Base */
/* Entity which needs to work with that Entity information from the UI.       */
/* -------------------------------------------------------------------------- */

import SCC_RULES_ENGINE:Util:RuleFactory;
import SCC_RULES_ENGINE:Util:RuleInterface;
import SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract;
import SCC_COMMON:ENTITY:IEntity;

Local RuleFactory &sccRuleFactory = create RuleFactory();
Local RuleInterface &SCC_RTRIG_20140218092909= &sccRuleFactory.getRule("SCC_RULE_ID⇒
```



```
_20140218093117", true);

Local EntityComponentAdapterAbstract  &objECA;
Local array of SCC_COMMON:ENTITY:IEntity &arrIEntity;
Local SCC_COMMON:ENTITY:IEntity &objIEntity;
Local row &row;
Local number &i;

/* Make sure you are using the proper Entity Component Adapter implementation for y⇒
our Component */
/* Remove the comment markers below to implement the ECA */

/* &objECA=CreateObject("SCC_COMMON:ENTITY:COMPONENT:ECAErrorGrid"); */

/*  If the current row is not the row where the entity exists, replace logic below ⇒
to get the correct row in the component */

&row = getrow();

/* Bind the row of data in the comonent to an entity */
/* Be sure to correctly bind the component to the entities used by the rule */

/* &arrIEntity = CreateArrayRept(&objIEntity, 0); */
/* &arrIEntity.Push(&objECA.BindToUIFromRowWithEntityID(&row, "SCC_ENTITY_201402140⇒
90313")); */
/* &SCC_RTRIG_20140218092909.Context = &arrIEntity; */


/* Execute the Rule */

&SCC_RTRIG_20140218092909.Invoke();
```

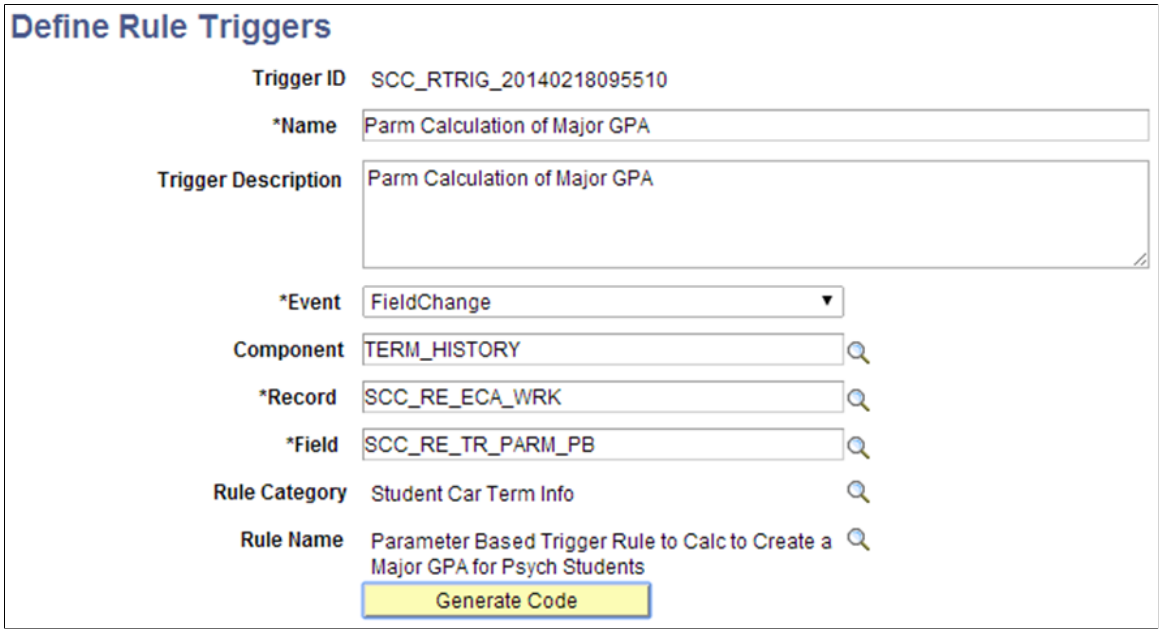
Generate Code for the Parm Calc

This example shows a Trigger based on Method B, which only contains Input Variables. The calculated GPA is passed back to the Component Event which takes care of updating the GPA value.

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

Image: Define Rule Triggers page for Parm Calc for Rules Engine User Interface Integration Example

This example illustrates the Define Rule Triggers page for Parm Calc for Rules Engine User Interface Integration Example.



Here is the example generated code:

```
/* ---------------------  About This Example Code  -------------------------- */
/* The following Example Code has been created as a starting point from       */
/* which to create Rule -UI Integration.  In the code you will find that the  */
/* rule refers to the Trigger Rule provided on the component as well as its   */
/* Arguments.  The code below will need to be adjusted before it can be used  */
/* to call the (Trigger) Rule.  For more information please refer to the      */
/* PeopleBooks documentation available for Rules Engine Functionality.        */
/*                                                                            */
/* ----------------  Understanding this Example Code  ----------------------- */
/* There are two ways to bring information from a Page or Component into a    */
/* Rule:                                                                      */
/*                                                                            */
/* Method 1: Providing Arguments from Rowset                                  */
/* Provide Local Strings with Values obtained from the appropriate Component  */
/* Rowset.  This method may be appropriate when calling a Rule or Function    */
/* without an Entity which requires a set of input Arguments (Variables) or   */
/* for a Rule with a Base Entity which requires a set of parameters which can */
/* be easily obtained and passed into the Rule.                               */
/*                                                                            */
/* Method 2: Providing Entity using EntityComponentAdapter                    */
/* Use the Entity Component Adapter to obtain Entity values from the          */
/* Component.  This method should be used when calling a Function with a Base */
/* Entity which needs to work with that Entity information from the UI.       */
/* -------------------------------------------------------------------------- */

import SCC_RULES_ENGINE:Util:RuleFactory;
import SCC_RULES_ENGINE:Util:RuleInterface;
import SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract;
import SCC_COMMON:ENTITY:IEntity;

Local RuleFactory &sccRuleFactory = create RuleFactory();
Local RuleInterface &SCC_RTRIG_20140218095510= &sccRuleFactory.getRule("SCC_RULE_ID⇒
```



```
_20140218095142", true);

Local EntityComponentAdapterAbstract  &objECA;
Local array of SCC_COMMON:ENTITY:IEntity &arrIEntity;
Local SCC_COMMON:ENTITY:IEntity &objIEntity;
Local row &row;
Local number &i;

/* Make sure you are using the proper Entity Component Adapter implementation for y⇒
our Component */
/* Remove the comment markers below to implement the ECA */

/* &objECA=CreateObject("SCC_COMMON:ENTITY:COMPONENT:ECAErrorGrid"); */

/* Input Variable Declaration */

Local String &Emplid;
Local String &Institution;
Local String &Career;
Local String &Term;
/* Output Variable Declaration */

Local Number &GPA; /* Maps to Rule Output GPA*/

<*

/* Assign values from the component to the rule variables */
/* Replace ? with the appropriate values */
/* Required variables must have values passed into the rule */
/* If optional variables do not need values assigned, leave the line commented out ⇒
*/

&Emplid = ?;  /* Maps to Rule Input Emplid (Required) */
&Institution = ?;  /* Maps to Rule Input Institution (Required) */
&Career = ?;  /* Maps to Rule Input Career (Required) */
&Term = ?;  /* Maps to Rule Input Term (Required) */

*>

ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0001" , &Emplid);
ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0003" , &Institution);
ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0005" , &Career);
ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0007" , &Term);/* Execute the Rule *⇒
/

&SCC_RTRIG_20140218095510.Invoke();

/* Assign values from the rule back to the component */
/* Replace ? with appropriate values */

&GPA = ObjectGetProperty(&SCC_RTRIG_20140218095510,"V0013");
```

Understanding the Generated Code

The generated code contains a reference to the *EntityComponentAdapterAbstract* because the Trigger uses a Base Entity. The Entity information must be passed from the Component to the Trigger Rule. The Entity Component Adapter provides the following functionality:

- The *EntityComponentAdapterAbstract* (ECA) is code that maps the data in the component to the Base Entity attached to the Rule.
- If the Base Entity has child Entities, the ECA also maps the Entity children.
- Rather than pulling data from the database, the ECA uses data in the component which may include unsaved changes. This means that the Rule can interact directly with data from the component.

- Changes made to the Entities in the Rule are made to the component data also.

The *ECAErrorGrid* handles any Errors which may be generated by the EntityComponentAdapter. This code is required whenever an ECA is used and provides the necessary Error Handling for the Entity Component Adapter. Two error methods exist:

- ECAErrorGrid – This code takes care of the error messages coming back from the ECA.
- ECAErrorPoPup – Opens a popup page which displays the errors coming back from the ECA.

The *IEntity* array needs to be instantiated as an Array object before you can map the data from a Component to the base Entity of a rule. Errors are generated if the Entity is directly pulled into the code without a push to the array first. In this example, the data is being retrieved and mapped using the *BindToUIFromRowWithEntityID* method.

BindtoUIfromRowWithEntity selects the correct Entity based on the Base Entity ID associated with the Trigger Rule on the Trigger Component. This generated code has already placed the *BaseEntityID* in the *BindtoUIfromRowWithEntity* construct.

Other methods are available through which to retrieve the Entity. They can be added but are not included in the Generated Code:

- *BindToUI:* This grabs the Entity Based on the record on level 0. This would be the first non–Work, non–View encountered in the Component.

  Note: For any Component which uses, for example, a record such as INSTALLATION on level 0, this does not work well. Binding at level 0 also implies you may be creating a relatively large Entity Tree. Performance is a consideration.

- *BindToUIfromRow:* This picks the first Entity on that rowset level for example level 1. It assumes that the Component structure and the Entity structure are going to be similar or the same. It "walks" the Component and tries to match what is in the Component with what is in the Entity.

  Note: This returns the first Entity found based on the RowSet indicated. Problems may occur when the there are multiple RowSets in the Component on that level.

- *BindToUIfromRecord:* This picks the Entity from the user interface based on the record indicated. This record is assumed to be the Production Record from which the Entity is mapped.

  Note: Multiple Entities could be mapped to the same Production Record. In this case, the first Entity is returned which may not be the right one.

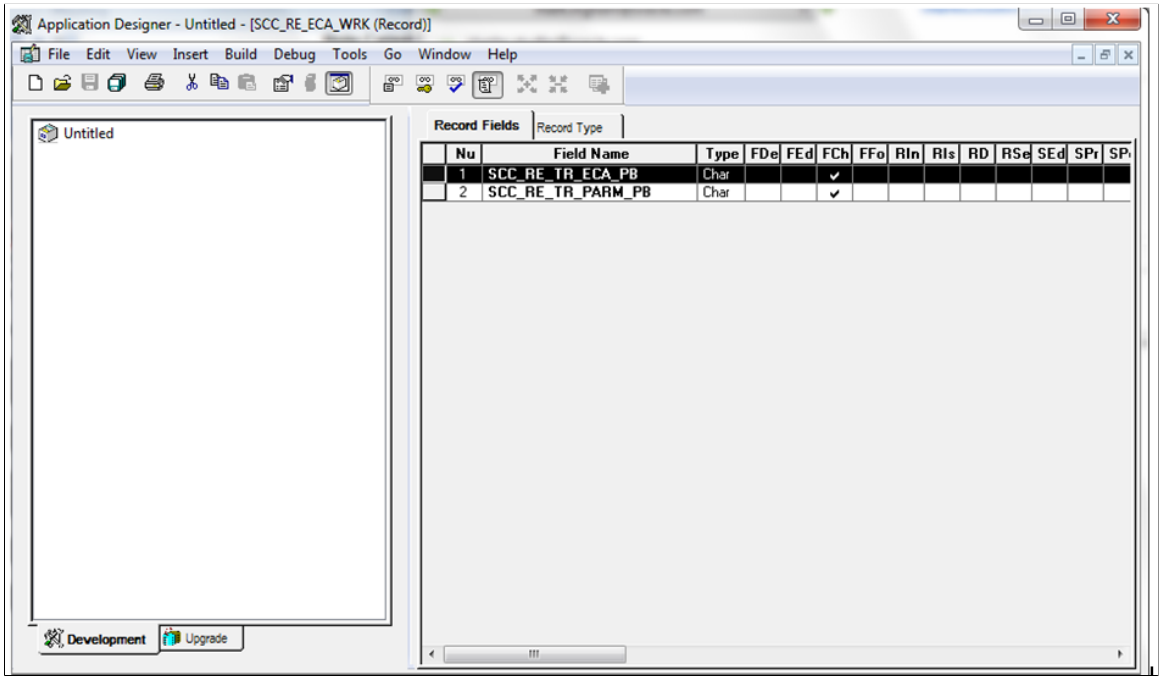
For more extensive and technical information regarding these Entity Methods, see Setting Up Entity Registry.

Attach Generated Code to the Component Event

The code can be copied from the Generated Code window and pasted into the PeopleCode Event. Open Application Designer and navigate to the PeopleCode Event. In this example, the Event is added to the *FieldChange* event of a *WorkRecord* field:

Image: Application Designer Example for Rules Engine User Interface Integration Example

This example illustrates the Application Designer Example for Rules Engine User Interface Integration Example.



Open the *FieldChange* event and paste the Generated Code for the ECA Trigger into the appropriate event. Adjust the code to retrieve the correct row from which to select the Entity. This is the adjusted PeopleCode on the *FieldChange* Event of field *SCC_RE_TR_ECA_PB* which is the field for the ECA Calc button:

```
import SCC_RULES_ENGINE:Util:RuleFactory;
 import SCC_RULES_ENGINE:Util:RuleInterface;
 import SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract;
 import SCC_COMMON:ENTITY:IEntity;

 Local SCC_RULES_ENGINE:Util:RuleFactory &sccRuleFactory = create SCC_RULES_ENGINE:⇒
Util:RuleFactory();
 Local SCC_RULES_ENGINE:Util:RuleInterface &SCC_RTRIG_20140218092909 = &sccRuleFact⇒
ory.getRule("SCC_RULE_ID_20140218093117", True);

 Local SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract &objECA;
 Local array of SCC_COMMON:ENTITY:IEntity &arrIEntity;
 Local SCC_COMMON:ENTITY:IEntity &objIEntity;
 Local Row &row;
 Local number &i;

 /* Make sure you are using the proper Entity Component Adapter implementation for ⇒
your Component */
 /* Remove the comment markers below to implement the ECA */
 &objECA = CreateObject("SCC_COMMON:ENTITY:COMPONENT:ECAErrorGrid");

 /*  If the current row is not the row where the entity exists, replace logic below⇒
 to get the correct row in the component */
```

```
 &row = GetRow();

 /* Bind the row of data in the comonent to an entity */
 /* Be sure to correctly bind the component to the entities used by the rule */


 &arrIEntity = CreateArrayRept(&objIEntity, 0);
 &arrIEntity.Push(&objECA.BindToUIFromRowWithEntityID(&row, "SCC_ENTITY_20140214090⇒
313"));
 &SCC_RTRIG_20140218092909.Context = &arrIEntity;



 &SCC_RTRIG_20140218092909.Invoke();
```

Based on the level 1 row in this example, the Entity can be selected from the current row. References to Variable and Calculated_GPA have been removed from code. As we are using the Rule to update the GPA Entity Property, there is no need to declare or assign a value to the &Calculated_GPA variable in this PeopleCode.

This is the adjusted PeopleCode on the *FieldChange* Event of field *SCC_RE_TR_PARM_PB* which is the field for the Parm Calc button:

```
import SCC_RULES_ENGINE:Util:RuleFactory;
 import SCC_RULES_ENGINE:Util:RuleInterface;
 import SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract;

 Local SCC_RULES_ENGINE:Util:RuleFactory &sccRuleFactory = create SCC_RULES_ENGINE:⇒
Util:RuleFactory();
 Local SCC_RULES_ENGINE:Util:RuleInterface &SCC_RTRIG_20140218095510 = &sccRuleFact⇒
ory.getRule("SCC_RULE_ID_20140218095142", True);

 Local SCC_COMMON:ENTITY:COMPONENT:EntityComponentAdapterAbstract &objECA;
 Local Row &row;
 Local number &i;

 /* Make sure you are using the proper Entity Component Adapter implementation for ⇒
your Component */
 /* Remove the comment markers below to implement the ECA */
 /* &objECA=CreateObject("SCC_COMMON:ENTITY:COMPONENT:ECAErrorGrid"); */

 /*  If the current row is not the row where the entity exists, replace logic below⇒
 to get the correct row in the component */
 &row = GetRow();

 /* Bind the row of data in the comonent to an entity */
 /* Be sure to correctly bind the component to the entities used by the rule */
 /* &SCC_RTRIG_20140218095510.Context = CreateArrayRept(&objECA.BindToUIFromRowWith⇒
EntityID(&row, "SCC_ENTITY_20140214090313"), 1); */

 /* Input Variable Declaration */

 Local string &Emplid;
 Local string &Institution;
 Local string &Career;
 Local string &Term;
 Local number &Sequence_Number;
 Local string &GPA_Type;

 /* Output Variable Declaration */

 Local number &GPA; /* Maps to Rule Output GPA*/

 /* Assign values from the component to the rule variables */
 /* Replace ? with the appropriate values */
 /* Required variables must have values passed into the rule */
 /* Optional variables do not need values assigned, leave the line commented out */
```



```
 &Emplid = &row.STDNT_SPCL_GPA.EMPLID.Value; /* Maps to Rule Input Emplid (Required⇒
) */
 &Institution = &row.STDNT_SPCL_GPA.INSTITUTION.Value; /* Maps to Rule Input Instit⇒
ution (Required) */
 &Career = &row.STDNT_SPCL_GPA.ACAD_CAREER.Value; /* Maps to Rule Input Career (Req⇒
uired) */
 &Term = &row.STDNT_SPCL_GPA.STRM.Value; /* Maps to Rule Input Term (Required) */
 &Sequence_Number = &row.STDNT_SPCL_GPA.SEQ_NUM.Value; /* Maps to Rule Input Sequen⇒
ce Number (Required) */
 &GPA_Type = &row.STDNT_SPCL_GPA.GPA_TYPE.Value; /* Maps to Rule Input GPA Type (Re⇒
quired) */

 ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0001", &Emplid);
 ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0003", &Institution);
 ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0005", &Career);
 ObjectSetProperty(&SCC_RTRIG_20140218095510, "V0007", &Term);

 &SCC_RTRIG_20140218095510.Invoke();

 /* Assign values from the rule back to the component */
 /* Replace ? with appropriate values */

 &GPA = ObjectGetProperty(&SCC_RTRIG_20140218095510, "V0013");

 &row.STDNT_SPCL_GPA.LS_GPA.Value = &GPA;
```

The PeopleCode has been adjusted to provide a *FieldValue* for each Variable to be provided to the Trigger. For example, &Emplid =&row.STDNT_SPCL_GPA.EMPLID.Value. In this example, the &GPA value retrieved is updated on the component by updating &row. STDNT_SPCL_GPA.LS_GPA.Value.

When the Events PeopleCode has been adjusted, the user interface integration can be tested from the Component page.

Library of System-Delivered Rules Engine Objects

This section lists all objects, including Lists of Values, Functions, Rules, Categories, Data Sets, and Entity Profiles that are delivered as System Data for Rules Engine. These System Functions, Rules, Categories and Entity Profiles are delivered as part of Rules Engine feature and have a specific purpose. Those which have a similar purpose or intended usage are grouped together into the same Rules Engine Category. Categories with Rules that access data are delivered with an appropriate entity profile.

Lists of Values

The following Lists of Values can be used in the Rules Engine when referenced by a Rules Engine Variable. The LOV provides Prompt Table functionality when variables are used in Rule criteria or evaluative statements.

LOV ID	*Description*	*LOV Type*	*LOV Prompt Record*	*LOV Prompt Field*
SCC_LOV_ 20130225065505	RE Honors	Table	HONOR_AWARD_ TBL	AWARD_CODE
SCC_LOV_ 20131220070639	NFK Notification	Translate Values	PSXLATITEM	SCC_NTFREQ_TYPE

LOV ID	Description	LOV Type	LOV Prompt Record	LOV Prompt Field
SCC_LOV_ 20140123052943	NFK Announcement Audience	Translate Values	PSXLATITEM	SCC_NTF_AUDCE
SCC_LOV_ 20140130120917	Yes/No Prompt	Ad Hoc Values	N/A	Y (Yes) N (No)
SCC_LOV_ 20140204053537	Rules Engine NFK Templates	Table	SCC_RULE_NTF_ VW	SCC_NTFREQ_ TMPLTID
SCC_LOV_ 20140204054709	Rules Engine Notification ID	Table	SCC_NTF_CON_ CFG	SCC_NTFREQ_ TMPLTID

System Variables

The following System Variables can be used in the Rules Engine when referenced by a Rules Engine Variable. The Rule Category Name is here for reference only as System Variables are considered available for use in any Rule.

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.
Current Time	DateTime	Returns the Current Time in format HH:MM:SS.
Operator Id	Text	Returns the value for the Current Operator ID.
User Language	Text	Returns the User Language based on User Defaults.

Data Sets

These System Data Sets can be attached to Rules Engine Variables and used as “placeholders” for multiple combined values. Data Sets can be used as lists.

Data Set ID	Name	Long Description	Purpose
SCC_ENTITY_ 20130418162950	Message Catalog Data Set	Message Catalog Data Set	This Data Set can be used in combination with a function to retrieve formatted Message Catalog Text from the system.
SCC_ENTITY_ 20130306064022	AIR Courses	AIR Courses	This data set can be used to as a placeholder for AIR course values

Data Set ID	Name	Long Description	Purpose
SCC_ENTITY_ 20130528134954	AM Results	AM Weights and Marks used for calculating the Primary Course Result	This data set is used in an primary Course result calculation Rule.
SCC_ENTITY_ 20130711074744	Academic Term	Academic Term	This Data Set is used in function Getterminformation and provides a combined set of Term setup table values.
SCC_ENTITY_ 20130711075100	Academic Session Information	Academic Session Information	This data set is used in a function which provides information for a term session.
SCC_ENTITY_ 20140120043511	ActiveResearchSupervisors	Active Research Supervisors	This data set is used in a function which provides information for a Research Supervisor.
SCC_ENTITY_ 20140120083337	ActiveResearchAdminProfiles	Active research administrator profiles	This data set is used in a function which provides information for a Research Affiliation Profile.
SCC_ENTITY_ 20140226150042	EMS Evaluation Information	EMS Evaluation Information	This data set aggregates relevant information about the Evaluation. Used to pass Evaluation data to Rules.
SCC_ENTITY_ 20140226150400	EMS Evaluation Keys	EMS Evaluation Keys	This data set contains evaluation key data (for example, Evaluation Instance, EMPLID, INSTITUTION). Used to pass key data to Rules in a generic fashion (for example, KEY_1, KEY_2, etc). **Note:** EMPLID and INSTITUTION properties have been added to the EMS Evaluation Information Dataset to simplify access to those fields.
SCC_ENTITY_ 20140226150717	EMS Evaluation XRef Keys	EMS Evaluation Cross-Reference Keys	This data set contains the Cross-Reference Keys of the Evaluation. Used to pass key data to Rules in a generic fashion (for example, KEY_1, KEY_2, etc).
SCC_ENTITY_ 20140226152106	EMS Rating Component	EMS Rating Component	This data set contains Rating Component fields. Used to pass Rating Component Information to Rules.

Data Set ID	Name	Long Description	Purpose
SCC_ENTITY_ 20140130004615	Text Catalog Data Set	Data set for Text Catalog based messages	This data set can be used in combination with System Function “CreateTextCatalogMessage “. This function retrieves message text from the Text Catalog and populates the data set.
SCC_ENTITY_ 20140224074528	NFK Template Variables	NFK Template Variables	This data set is used by delivered Notification Framework rules to populate required notification variables.
SCC_ENTITY_ 20140224073623	Email Attachments	Email Attachments	This data set is used by Notification Framework Rules to populate Email Attachment values.

System Delivered Categories

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20130121071454	SystemTest	This Category exists to build and test Delivered Developer functions and system variables. The System Test Category consists of self–sustaining mini test suites that can test whether delivered functions work the way they should.	System Test Functions which should be used on test environments for system testing only. These Rules can be on test environments to test basic functionality of the Rules Engine. System Tests include a basic test run of all delivered functions and Rules and includes data delivered as test profile where possible. The test profile data is based on Campus Solutions demo data and may need to be changed for use on your test environment.
SCC_RULE_CAT_ 20130211055805	APT Functions	APT Functions	General purpose functions which can be used by institutions as an example of how to build similar Rules.
SCC_RULE_CAT_ 20130510122554	Student Records Generic	Student Records Generic Functions	Student Records Generic Functions.
SCC_RULE_CAT_ 20120711124111	Math	All Generic Math Functions	These functions can be used to perform math functions. For example Add, Divide, Subtract.

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20120711141824	String	All Generic String Functions	These functions can be used to perform manipulations of string values. For example, Uppercase which converts lowercase text to uppercase text.
SCC_RULE_CAT_ 20120711155409	DateTime	All Generic Date, Time, and DateTime Functions	These functions can be used to perform different Date and DateTime manipulations. For example AddDaystoDate, which adds a specified number of days to date and returns the new date.
SCC_RULE_CAT_ 20120802213716	Entity Functions	Functions for dealing with the entity context. Save, Validate, Add, Remove etc.	These functions can be used to Save, Validate, Add, and Remove Entities. These are the functions that are commonly used to insert and/ or update data into data tables.
SCC_RULE_CAT_ 20120807013921	Debug	Functions to assist in debugging issues	These functions can be called for debugging purposes. The functions put debugging information into a log file.
SCC_RULE_CAT_ 20121203135246	Number	All Generic Number Functions	Currently no number functions have been delivered. This Category is a place holder for future usage.
SCC_RULE_CAT_ 20130530144109	Text Messages	Text Messages	These functions can be used to populate Message text from either the Message Text Catalog or the Text Catalog. Use functions from this category if you want to retrieve formatted Message text into your Rule.

Category for Activities Management

Rule Category for the Activities Management feature.

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20130425123523	AM Calculation Rules	AM Calculation Rules	These Rules are delivered as part of the Activity Management (AM) feature.

Categories for Program Enrollment

Rule Categories for the Program Enrollment feature.

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20130219025032	AIR Functions	AIR Functions	General purpose functions that access the academic item registry and can be used by institutions as an example of how to build similar Rules or be leverages as-is into their own Rules
SCC_RULE_CAT_ 20130211055805	APT Functions	APT Functions	General purpose functions that access the academic progress tracker and can be used by institutions as an example of how to build similar Rules or be leverages as-is into their own Rules

Categories for Research

Rule Categories defined for Research are listed below. They are used for Service Requests components, Research Self Service components, and for online edits in Research administrative components.

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20140120040657	Research Functions	Research Functions	These functions are used in Rules belonging to the other Research categories.
SCC_RULE_CAT_ 20140116030817	Research Candidates	Research Candidate Rules This includes Consumption and Thesis Rules as the primary keys for these entities are the same as for a candidate.	These Rules are most likely used in the Research Rule Types Component.
SCC_RULE_CAT_ 20130823033945	Research Supervisors	Research Supervisor Rules	These Rules are most likely used in the Research Rule Types Component.
SCC_RULE_CAT_ 20140122021007	Research Topics	Research Topic Rules	These Rules are most likely used in the Research Rule Types Component.
SCC_RULE_CAT_ 20140205035611	Research Assignments	Research Assignment Rules	These Rules are most likely used in the Research Rule Types Component

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20140128233029	Service Request Functions	Service Request Functions	These Functions would be used in Service Request Category Rules.
SCC_RULE_CAT_ 20131028041439	Service Requests	Research and non research service request Rules	These Rules are used in Rule Trigger definitions for Research Service Requests Components for student and administrators.
SCC_RULE_CAT_ 20140115030355	Research Self Service	Task Rules used to determine the recipient of the notification, when the student performs a Service on the Research Candidate self service page	These Rules are used in the Research Service ID Setup component to determine the Notification Recipient EmplID.

Category for Evaluation Management System (EMS)

Rule Category defined for Evaluation Management System are listed below. They are used for calculating Rating Component Values and Rating Scheme values.

Category ID	Name	Long Description	Purpose
SCC_RULE_CAT_ 20140226144850	Evaluation Management System	Evaluation Management System	Contains Rule Groups and Data Sets required to implement Rules Engine processing for EMS; along with several example Rules.

Entity Profiles

Profile ID	Name	Long Description	Purpose
SCC_EPRFL_ 20130329103345	System Profile AIR & APT	System Profile AIR & APT	Used by Rules Engine Category AIR Functions and APT functions which need to access System Academic Items.
SCC_EPRFL_ 20130306044322	System Data Set	System Data Set	Data Set Profile which is used to deliver System Data Sets.
SCC_EPRFL_ 20130510082227	System Profile AM	System Profile Activity Management	This profile is used by system —delivered Categories for Activity Management.

Profile ID	Name	Long Description	Purpose
SCC_EPRFL_ 20140214091401	Student Car Term Information	Student data in STDNT_CAR _TERM and child records.	This profile is used to illustrate the proof of concept for Rules Engine user interface integration.

Entities

The following Entities are delivered as Rules Engine objects.

Entity ID	Name	Long Description	Purpose
SCC_ENTITY_ 20140218085710	Student Enrollment - Read Only	This entity is used to pull enrollment information for a specific student using CLASS _TBL_SE_VW.	This Entity is used to illustrate the proof of concept for Rules Engine user interface integration.
SCC_ENTITY_ 20140214090313	Student Special GPA	Entity for Student Special GPA in Student Records using STDNT_SPCL_GPA.	This Entity is used to illustrate the Proof of concept for Rules Engine user interface integration.

Rule Groups

Rule Groups provide a template in which Category, Base Entity, Rule Usage and Input and output Variables are predefined. Each Rule created in a Rule Group inherits the same characteristics as the Rule Group. Rule Groups are commonly used to provide a template for User Interface integration or when there is the need to call multiple Rules in the same manner using the same characteristics.

Rule Group ID	Name	Long Description	Purpose
SCC_RULEGR_ID_ 20140226153734	EMS Rating Component Calculation	EMS Rating Component Calculation	All Rules calculating Rating Component values within a Rating Scheme must use this Rule group.
SCC_RULEGR_ID_ 20140226154255	EMS Rating Scheme Calculation	EMS Rating Scheme Calculation	All Rules calculating Rating Scheme Overall Rating values must use this Rule group.

Rule Group ID	Name	Long Description	Purpose
SCC_RULEGR_ID_ 20140224091018	Academic Progress Tracker Item (version 1.1)	Academic Progress Tracker Item Rule Group	This Rule Group provides a uniform set of input and output parameters to all Rules attached. The uniform template enables interaction with the Academic Progress Tracker user interface functionality delivered for Program Enrollment. This Rule group replaces Rule Group “Academic Progress Tracker Item” with ID SCC_RULEGR_ID_ 20130603144145
SCC_RULEGR_ID_ 20130603144145	Academic Progress Tracker Item	Academic Progress Tracker Item Rule Group	This Rule Group has been Inactivated.
SCC_RULEGR_ID_ 20140122023122	Research Assignments	Research Assignments Rule Group	This Rule Group provides a uniform set of input and output parameters to all Research Assignment Rules attached to this Rule Group.
SCC_RULEGR_ID_ 20140116025712	Research Candidate	Research Candidate Rule Group	This Rule Group provides a uniform set of input and output parameters to all Research Candidate Rules attached to this Rule Group.
SCC_RULEGR_ID_ 20140206012041	Research Thesis	Research Thesis Rule Group	This Rule Group provides a uniform set of input and output parameters to all Research Thesis Rules attached to this Rule Group.
SCC_RULEGR_ID_ 20140115031545	Research Self Service Task	Research Self Service Task	This Rule Group provides a uniform set of input and output parameters to all Research Self Service Rules attached to this Rule Group.
SCC_RULEGR_ID_ 20131202050022	Research Supervisor	Research Supervisor	This Rule Group provides a uniform set of input and output parameters to all Research Supervisor Rules attached to this Rule Group.
SCC_RULEGR_ID_ 20140205033942	Research Topic	Research Topic Rule Group	This Rule Group provides a uniform set of input and output parameters to all Research Topic Rules attached to this Rule Group.

Rule Group ID	Name	Long Description	Purpose
SCC_RULEGR_ID_ 20131028050041	Service Request	Service Request Rule Group based on Service Request Header entity	

System Test Category Functions

Rule ID	Name	Long Description
SCC_RULE_ID_20130218075844	System Test: Simple Evaluative Statement Test (Honors Example)	This system test Rule tests a simple evaluative statement. The basis for this test is a simplified example of an Honors Classification Evaluation Rule (Judicium).
SCC_RULE_ID_20130129055908	Test Math Functions (CALL all Math Functions)	This Function has been created to test through all delivered math functions. Testing includes the following functions in Category "Math": • Add • Subtract • Multiply • Divide • Round • Truncate • Mod
SCC_RULE_ID_20130129073324	Test String Functions (CALL all String Functions)	This function Tests through all delivered String Functions.

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130212065258	System Test: APT Entities: Example Simple Judicium/Honors Rule	This Rule has been created to Test the Rules Engine interacting with System Delivered APT Academic Items. The Rule uses Entities created from System Delivered academic Item Types in APT and tests whether delivered functions and statements interact correctly with these entities. **Note:** This Rule should NOT be run on a production Environment. Example Rule: The student has completed a Bachelor's program and thinks that based on his grades that he should be able to request an Honors classification on his degree. To determine whether an Honors classification applies, a number of things need to be evaluated: • Total Credit needs to be equal to or above specified amount • Average Mark must be above specified minimum • Lowest Mark must be above specified minimum PREREQUISITE: You need to have three distinct Result Types defined which can be used to enter Results against planned academic item types in the Academic Progress Tracker. These Result Types Names should be added to the following variables as defaults: • Variable Result Type: For numeric results stored against the Course academic Item. • Variable Result Type Credits: For numeric credit results stored against the Course academic item. • Variable Honor Result Type: For alphanumeric result which holds the Honors Classification. Use a Text field and not a prompt table.

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130121071654	Test List Functions (Copy List Via Assign)	The purpose of this System test function is to run through several scenario's in which list functions are tested: 1. Assignment of complete list to List (Text). 2. Assignment of complete list to List (Number), 3. Assign value from list to individual Variable in For-Each and evaluate content. 4. Evaluate value from list in For-Each without assigning to Variable first.
SCC_RULE_ID_20130524045353	Test List Functions II (call All list functions)	The purpose of this System test function is to run through several scenarios in which list functions and variables are tested: 1. Add to list. 2. Length of list. 3. Sort list. 4. Assign list. 5. Clear list.
SCC_RULE_ID_20130516005432	Test List Functions III (Data Set list)	The purpose of this System test function is to run through several scenarios in which list functions using data set variables are tested: 1. Add to list. 2. Length of list. 3. Sort list. 4. Assign list. 5. Clear list.

Rule ID	Name	Long Description
SCC_RULE_ID_20130205094910	Test Date Functions (CALL all DateTime Functions)	Test Date Function (CALL all DateTime Functions) • Add Days to Date • Add Months to Date • Retrieve Day from CurrentDate • Retrieve Hour, Minute, Seconds from Time • Add Years to Date • Return Day from Date • Get Current Date • Compare Current Dates • Return Hour from Time • Return Minute from Time • Subtract Days and Months and Years from Date • Retrieve Current Year from Date

AIR Category Functions

Rule ID	Name	Long Description
SCC_RULE_ID_20130219023428	SystemTest: Rule Retrieves AIR and Course Catalog information	SystemTest: Rule Retrieves AIR and Course Catalog Course information based on a specified Academic Item ID, an institution value and an effective date. The information retrieved is passed into a Data Set.
Bundle 43. New AIR Category function. SCC_RULE_ID_20160425080318	AIR description	Get AIR header setup description filtering by Academic Institution, Academic Item ID and Date.

APT Category Functions

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130212023549	Retrieve Maximum Attempt	Functions Tested: This Rule retrieves the Maximum Attempt row for a specified academic item in the student's Academic Progress Tracker. Input variables needed are the following key fields which are needed to identify the specific Academic Item Attempt row. The information can be retrieved from the Academic Item for which you need to retrieve the maximum attempt: • Student ID • Institution • APT Instance • APT Item sequence Example Use: You want to add a new attempt row into to the Academic Progress Tracker for Course Math101 because the student failed the previous attempt, but you need to know what the maximum registered attempt is prior to inserting.
SCC_RULE_ID_20130305043300	SystemTest Function: Assign APT Result Values and Save	This Function can be used to assign a Result Type and a Numeric Result or Alphanumeric Result to an APT Result row in the Student Academic Progress Tracker. The Save Entity Statement is called as part of this function to immediately save the created or updated Row. Attempt Outcome is set to “Conditional" status 20. Include in Calculation Flag is set to *Yes*.
Bundle 43. New APT Category function. SCC_RULE_ID_20160524125846	Get Begin Date for APT Curriculum Term.	Get Begin Date for APT Curriculum Term.

APT Category Function Rules

Bundle 43. Updated information.

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130607153122	Check for Sub-Plan Condition: Marketing (Version 3 is Active.)	Checks APT/Program stack for MARKETING Sub-Plan as a condition for selecting an APT option.
SCC_RULE_ID_20130603165523	Precondition for 2155: Expert (Version 3 is Active.)	Student must have satisfied the requirements for 'Economics Year 1 Option List A' (1743), and must have completed MGMT 1001 with at least a grade of C.
SCC_RULE_ID_20140320070913	Check for Sub-Plan Condition: Marketing TEXT CATALOG (Version 2 is Active.)	Checks APT/Program stack for MARKETING Sub-Plan as a condition for selecting an APT option using TEXT CATALOG.

Activity Management Calculation Category Rules

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130508133749	AM One Time Late Penalty	The AM One Time Late Penalty subtracts the one time penalty from the student's earned mark and inserts a new result row for the student.
SCC_RULE_ID_20130425124530	AM Primary Course Result	AM Primary Course Result.
SCC_RULE_ID_20130521103202	AM Capped Late Penalty	The AM Capped Late Penalty evaluates the earned mark entered. If the earned mark is less than the capped mark parameter, the student's earned mark is retained. If the earned mark is greater than the capped mark, the student receives the capped mark.
SCC_RULE_ID_20130521095817	AM Daily Late Penalty	The AM Daily Late Penalty evaluates the earned mark entered and the number of days the submission is late. The late penalty is inserted on a new result row for the student.
SCC_RULE_ID_20130521094603	AM Fixed Penalty Rule	The AM Fixed Late Penalty inserts on a new result row for the student with the value indicated on the fixed late penalty Rule.
SCC_RULE_ID_20130516135412	AM Expires to Zero Penalty	AM Expires to Zero Penalty.

Rule ID	Name	Long Description
SCC_RULE_ID_20130515141726	AM Weekly Late Penalty	The AM Weekly Late Penalty evaluates the earned mark entered and the number of weeks the submission is late. The late penalty is inserted on a new result row for the student.
SCC_RULE_ID_20130515130457	AM Late Penalty Online Driver Rule	This Rule applies late penalties automatically on the IAM Result pages (Result Roster, including both the administrative and WorkCenter rosters); and IAM Result Details) when a mark is entered with a submission date after the due date for a specific assessment item. The Rule returns a late penalty value and a new row indicating the penalty is inserted on the student's result. Late penalty parameters are setup on either the Activity Registry or the Activity Manager. The late penalty options include: a one-time penalty, a daily penalty, a weekly penalty, a fixed penalty or a capped penalty. Depending on the chosen parameter this Rule calls the associated penalty Rule for the calculation of the penalized mark

Date and Time Category Functions

Functions in the following category can be used to execute business logic using Date and Time values. For example, you can use the functions to convert Strings to Date or to calculate the number of days between two Date values.

Rule ID	Name	Long Description
SCC_RULE_ID_20120711162453	GetCurrentDate	Gets the Current Date from the System.
SCC_RULE_ID_20130513065254	DaysToWeeks	This function takes the number of days and convert it to weeks, truncating the calculated value.
SCC_RULE_ID_20130509114010	YearsBetweenDates	This function takes two dates and calculates the number of years, truncating any partial year between the two dates. If the Date From value is later than the Date To field, the result is a negative number.
SCC_RULE_ID_20130508165644	MonthsBetweenDates	This function takes two dates and calculates the number of months truncating any partial month between the two dates. If the Date From value is later than the Date To field, the result is a negative number.

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20130508164259	WeeksBetweenDates	This function takes two dates and calculates the number of weeks truncating any partial week between the two dates. If the Date From value is later than the Date To field, the result is a negative number.
SCC_RULE_ID_20130508152325	DaysBetweenDates	This function takes two dates and calculates the number of days between the two dates. If the Date From value is later than the Date To field, the result is a negative number.
SCC_RULE_ID_20120711181602	Year	Returns the year value from a date.
SCC_RULE_ID_20120711180936	Month	Returns the month value from a date.
SCC_RULE_ID_20120711180404	Hour	Returns the hour value from a time.
SCC_RULE_ID_20120711171519	SubtractYearsFromDate	Subtract a specific number of years from a date returning the calculated date.
SCC_RULE_ID_20120711171113	SubtractMonthsFromDate	Subtract a specific number of months from a date returning the calculated date.
SCC_RULE_ID_20120711165819	SubtractDaysFromDate	Subtracts a specific number of days from a date returning the calculated date.
SCC_RULE_ID_20120711165049	AddYearsToDate	Add a specific number of years to a date returning the calculated date.
SCC_RULE_ID_20120711164350	AddMonthsToDate	Add a specific number of months to a date returning the calculated date.
SCC_RULE_ID_20120711163227	AddDaysToDate	Add a specific number of days to a date returning the calculated date.
SCC_RULE_ID_20120725170243	Second	Returns the second value from a time.
SCC_RULE_ID_20120725165438	Minute	Returns the minute value from a time.
SCC_RULE_ID_20120725163802	Day	Returns the day value from a date
SCC_RULE_ID_20131119192253	StringToDate	Convert a String in the format YYYY-MM-DD to a Date.
SCC_RULE_ID_20131119193042	StringToDateTime	Convert a String to a Date Time value.
SCC_RULE_ID_20131119193911	StringtoTime	Convert a String to a Time value.

Debug Category Functions

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20120807014818	Debug Current Context	Outputs the current context as XML.
SCC_RULE_ID_20120807014738	Debug Context	Outputs the current full context in xml to the debug log in informational logging level.

Entity Category Functions

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20120802214353	SaveAllEntity	Saves all the entities in the Context and their children. It performing all validation and pre-save logic and deleting any entities marked for deletion.
SCC_RULE_ID_20120806111753	ValidateEntity	Runs entity validation, providing the error/warning state.
SCC_RULE_ID_20120802213958	SaveEntity	Saves the Current Entity Context and all Children. It performs all validation and pre-save logic and deleting any entities marked for deletion.
SCC_RULE_ID_20120802215742	UnDeleteEntity	Marks the current entity and all it's children for to be undeleted. This simply marks the entity for undeletion, to finish the undelete call save.
SCC_RULE_ID_20120802215730	DeleteEntity	Marks the current entity and all it's children for deletion. This simply marks the entity for deletion, to finish the delete call save.

Math Functions

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20120711125620	Subtract	Subtract one value from another giving a result. Result = Sum(Value 1 - Value 2)
SCC_RULE_ID_20120711124755	Add	Add two values into a results Result = Add(Value 1 + Value 2)
SCC_RULE_ID_20120716174126	SubtractTruncate	Subtract one value from another giving a result truncated to the precision. Result = Sum(Value 1 - Value 2), Precision

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20120716173508	SubtractRound	Subtract one value from another giving a result rounded to the precision. Result = Round(Value 1 - Value 2), Precision
SCC_RULE_ID_20120716173227	AddTruncate	Add two values together giving a result rounding to the given precision. Result = Round(Value 1 + Value 2)
SCC_RULE_ID_20120712162241	AddRound	Add two values together giving a result rounding to the given precision. Result = Round(Value 1 + Value 2)
SCC_RULE_ID_20120712161717	MultiplyTruncate	Multiply two values together giving a result truncating to the given precision. Result = Truncate(Value 1 * Value 2), Precision
SCC_RULE_ID_20120712160911	MultiplyRound	Multiply two values giving a result rounding it to the given precision. Result = Round(Value 1 * Value 2)
SCC_RULE_ID_20121204164104	AverageTruncate	Sum the contents of a list and divide by the number of items in the list, truncating the result by the precision. Ave = Truncate(Sum(List)/Len(List), Precision)
SCC_RULE_ID_20121204163908	AverageRoundDown	Sum the contents of a list and divide by the number of items in the list, rounding down the result by the precision. Ave = RoundDown(Sum(List)/Len(List), Precision)
SCC_RULE_ID_20121204163746	AverageRoundUp	Sum the contents of a list and divide by the number of items in the list, rounding up the result by the precision. Ave = RoundUp(Sum(List)/Len(List), Precision)
SCC_RULE_ID_20121204163355	AverageRound	Sum the contents of a list and divide by the number of items in the list, rounding the result by the precision. Ave = Round(Sum(List)/Len(List), Precision)

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20121204161244	Average	Sum the contents of a list and divide by the number of items in the list. Ave = Sum(List)/Len(List)
SCC_RULE_ID_20120712155703	DivideRoundDown	Divide one value from another giving a result rounding down to the given precision. Result = Divide(Value 1 / Value 2)
SCC_RULE_ID_20120712151532	DivideRoundUp	Divide on value into another giving a result rounding up to the given precision. Result = RoundUp(Value 1 / Value 2)
SCC_RULE_ID_20120712130748	DivideTruncate	Divide one value from another giving a result truncating it to the given precision. Result = Divide(Value 1 / Value 2)
SCC_RULE_ID_20120712125721	DivideRound	Divide one value from another giving a result rounding it to the given precision. Result = Divide(Value 1 / Value 2)
SCC_RULE_ID_20120712121748	RoundDown	Round a numeric value to a specific number of digits. Result = RoundDown(Value, Decimal Places) This function performs rounding where value less than 1.0 would result the value being truncated unless the number is negative, then it is rounded down.
SCC_RULE_ID_20120712114110	RoundUp	Round a numeric value to a specific number of digits. Result = RoundUp(Value, Decimal Places) This function performs rounding where value greater than 0.0 would result the value being rounded to 1, and values at 0.0 would result in the value not being rounded.
SCC_RULE_ID_20120730124647	SubtractRoundUp	Subtract one value from another giving a result rounded up to the precision. Result = RoundDown(Value 1 - Value 2), Precision

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20120730123301	AddRoundDown	Add two values together giving a result rounding down to the given precision. Result = Round(Value 1 + Value 2), Precision
SCC_RULE_ID_20120730121901	MultiplyRoundDown	Multiply two values giving a result rounding down to the given precision. Result = RoundDown(Value 1 * Value 2), Precision
SCC_RULE_ID_20120730121234	MultiplyRoundUp	Multiply two values giving a result rounding up to the given precision. Result = RoundUp(Value 1 * Value 2), Precision
SCC_RULE_ID_20120730124327	SubtractRoundDown	Subtract one value from another giving a result rounded down to the precision. Result = RoundDown(Value 1 - Value 2), Precision
SCC_RULE_ID_20120730123457	AddRoundUp	Add two values together giving a result rounding up to the given precision. Result = RoundUp(Value 1 + Value 2), Precision
SCC_RULE_ID_20120711130546	Multiply	Multiply Value 2 values giving a result. Result = Multiply(Value 1 x Value 2)
SCC_RULE_ID_20120711140717	Mod	This function performs the modulus math function, returning the remainder when dividing one number by another. Result = Modulus(Value 1 / Value 2)
SCC_RULE_ID_20120711134943	Truncate	Truncate the decimal number to a specific precision. Result = Truncate(Value, Precision)
SCC_RULE_ID_20120711133255	Round	Round a numeric value to a specific number of digits. Result = Round(Value, Decimal Places) This function performs natural rounding, where value of 0.5 or higher would result the value being rounded to 1, and values lower than 0.5 would result in the value being rounded to 0.

Rule ID	Name	Long Description
SCC_RULE_ID_20120711131114	Divide	Divide one value from another giving a result. Result = Divide(Value 1 / Value 2)
SCC_RULE_ID_20131119134507	StringToNumber	Convert a string to a number.
SCC_RULE_ID_20131120095322	Max	Retrieves the maximum value from a list of numbers.
SCC_RULE_ID_20131120095954	Min	Retrieves the minimum value from a list of numbers.

String Category Functions

Functions in the following category can be used to execute business logic using string values. For example, you can use the functions to convert DateTime or Numbers to String.

Rule ID	Category Name	Name	Long Description
SCC_RULE_ID_ 20130419101915	String	Concatenate	Concatenate two strings into a single string.
SCC_RULE_ID_ 20120711154104	String	UpperCase	Converts a string to upper case.
SCC_RULE_ID_ 20120711154450	String	LowerCase	Convert a string to lower case.
SCC_RULE_ID_ 20120711153301	String	Substring	Returns a string from a longer string based on position.
SCC_RULE_ID_ 20120711152058	String	StringLength	Returns the length of a string.
SCC_RULE_ID_ 20131119143620	DateTimeToString	Convert a DateTime value to a String	SCC_RULE_ID_ 20131119143620
SCC_RULE_ID_ 20131119140237	DateToString	Convert a Date to a String	SCC_RULE_ID_ 20131119140237
SCC_RULE_ID_ 20131119135320	NumberToString	Convert a Number To a String	SCC_RULE_ID_ 20131119135320
SCC_RULE_ID_ 20131119185227	TimeToString	Convert a Time value to a String	SCC_RULE_ID_ 20131119185227

Number Category Functions

Functions in the following category can be used to execute business logic using numeric values. For example, you can use the functions to convert numbers to string or retrieve a maximum number from a number list.

Rule ID	Name	Long Description
SCC_RULE_ID_20131119134507	StringToNumber	Convert a string to a number.
SCC_RULE_ID_20131120095322	Max	Retrieves the maximum value from a list of numbers.
SCC_RULE_ID_20131120095954	Min	Retrieves the minimum value from a list of numbers.

Using Number Category Functions

- Using Min and Max
    - Description: When using delivered statements Min or Max in the Number Category, you can retrieve a minimum or maximum Mark, respectively. A minimum or maximum number can be retrieved from a Variable of Type List which contains all numeric values or from Lists of type Data Set. In this case the minimum or maximum can be retrieved from the property of type number.
    - How to Use: You create a Data Set List which contains a list of courses with accompanying Course Mark. From this list, you want to retrieve the Course with the highest Course Mark. The function Max can be used to retrieve the Maximum result from the list.
- Using StringToNumber
    - Description: This function converts a numeric string into a true number. Because a numeric value obtained from a property of type string can only be assigned to a Variable of type String, it may be necessary to convert the numeric string to a number before it can be used.
    - How to Use: For example, a value, 77, is retrieved from a property of type string. You would like to use this value in a calculation. The Math function Add only accepts numeric values or values from a numeric variable as input. In order to use the function Add, the variable which contains value 77 must first be converted into a true number. This can be done with function StringToNumber.

Student Records Generic Category Functions

Rule ID	Name	Long Description
SCC_RULE_ID_20130510122816	GetTermInformation	This Rule returns all information from the TERM_TBL.
SCC_RULE_ID_20130511184959	ValidateStudentAPTProgram	Validate Student Program using the academic program information in the APT.
SCC_RULE_ID_20130511192230	ValidateStudentAPTCareer	Validate Student Career using the academic program information in the APT.

Rule ID	Name	Long Description
SCC_RULE_ID_20130511190534	ValidateStudentAPTSubplan	Validate Student Subplan using the academic program information in the APT.
SCC_RULE_ID_20130511183011	ValidateStudentAPTPlan	Validate Student Plan using the academic program information in the APT.
SCC_RULE_ID_20130510201105	ValidateStudentSubplan	Validate Student Subplan.
SCC_RULE_ID_20130510195809	ValidateStudentPlan	Validate Student Plan.
SCC_RULE_ID_20130510193556	ValidateStudentProgram	Validate Student Program.
SCC_RULE_ID_20130510180645	ValidateStudentCareer	Validate Student Career.
SCC_RULE_ID_20130510163902	GetSessionInformation	Get information for a term session.

Create Text Message Category Functions

Rule ID	Name	Long Description
SCC_RULE_ID_20130530144525	CreateTextMessage	Create a text message data set and populate it with a message from the message catalog. Up to 9 parameters for the message can be used.
SCC_RULE_ID_20140217002205	CreateTextCatalogMessage	Create a text catalog message data set and populate it with a message from Text Catalog. Up to 4 context keys and 5 parameters for the message can be used.

Using Delivered Text Message Rules

Functions in the Text Message Category allow you to retrieve a formatted Text Message from the System. Text messages can be retrieved from the following functionality:

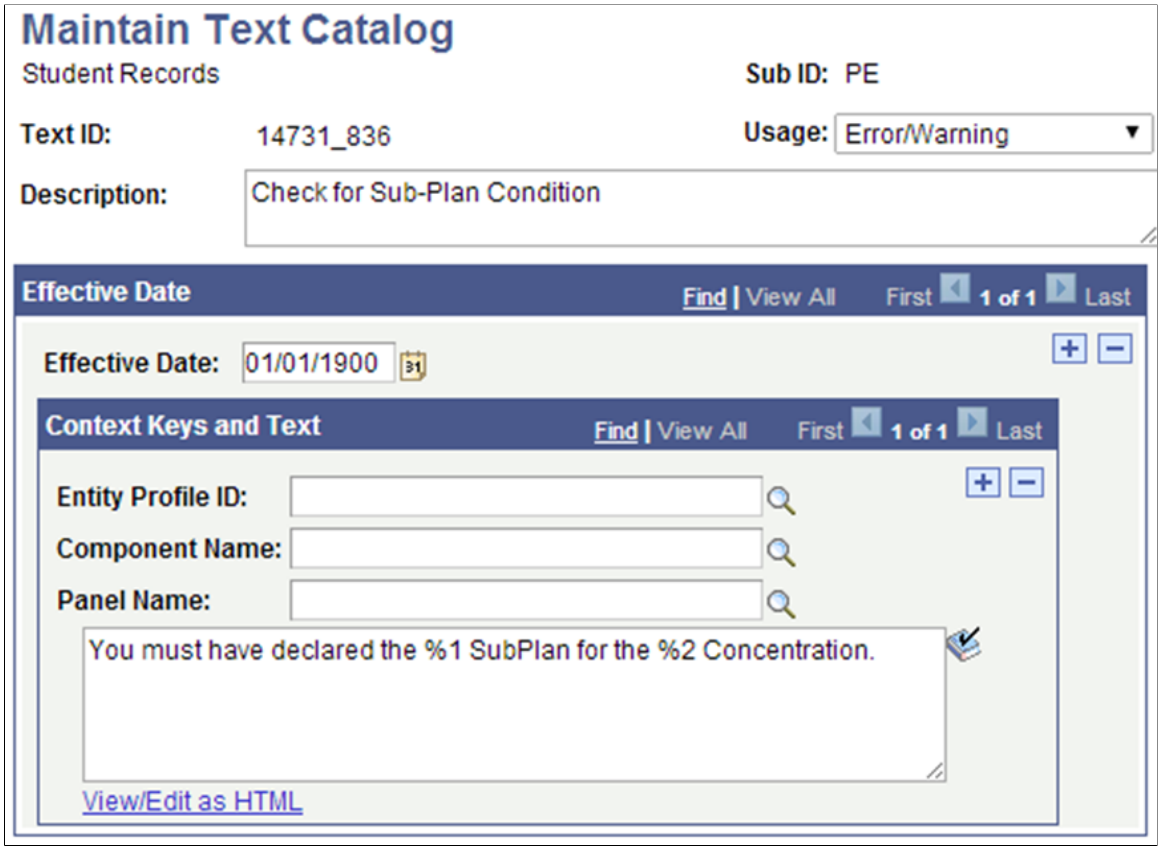
- Message Text Catalog
- Text Catalog

To use Message Text Catalog Text in your Rule, a message catalog text entry needs to exist.

Access Message Text catalog via PeopleTools, System Administration, Utilities, Administration, Message Catalog to view or create Message Catalog entries.

Image: Maintain Text Catalog Page for Rules Engine Create Text Message Example

This example illustrates the fields and controls for the Maintain Text Catalog Page for Rules Engine Create Text Message Example.

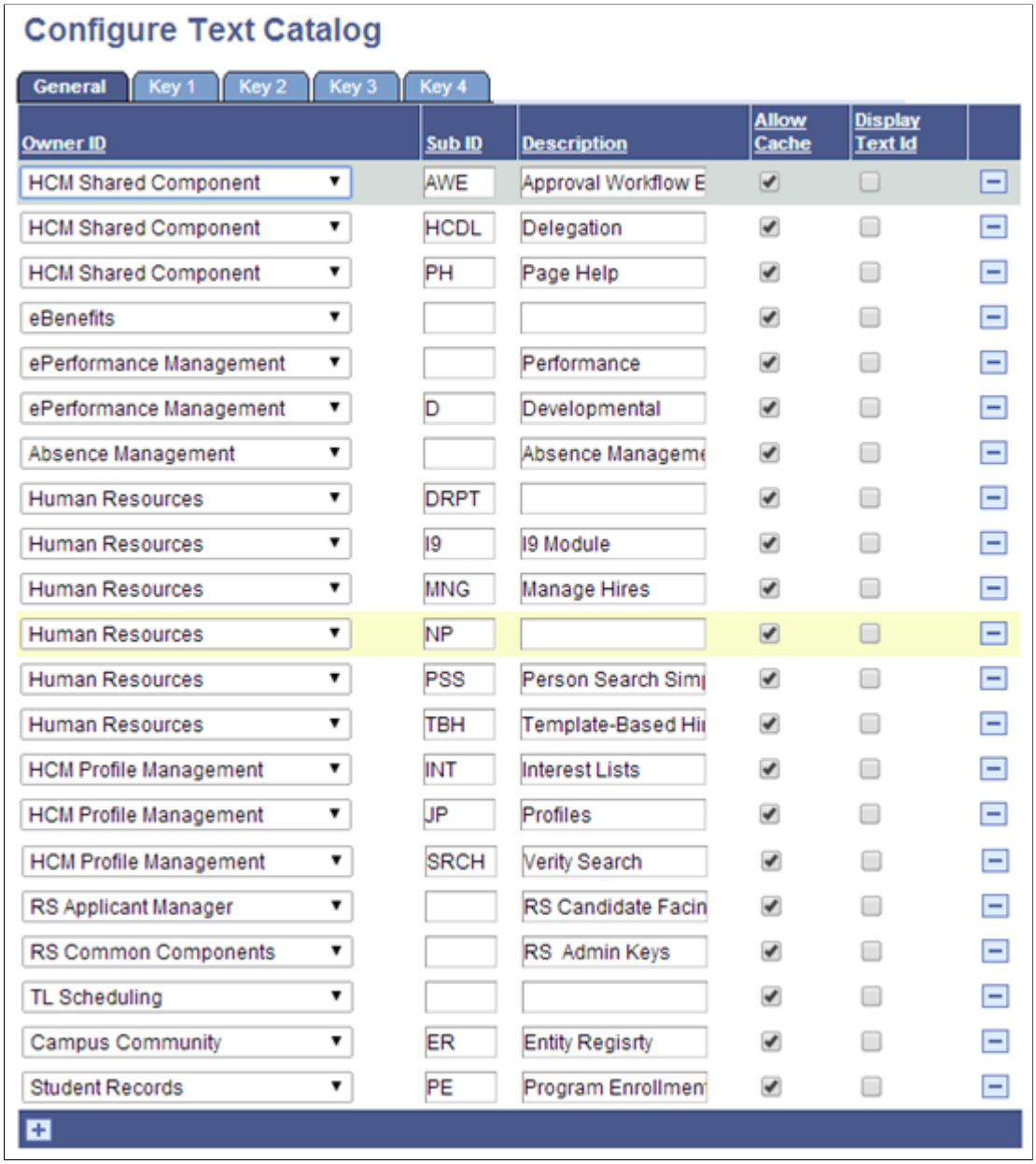


Sub ID’s must be created prior to creating a Text Catalog Text entry.

Access Set Up Common Objects, Common Definitions, Text Catalog and Notepad, Configure Text Catalog to set up Configure Text Catalog to add a relevant Sub ID for the relevant application area.

Image: Configure Text Catalog Page for Rules Engine Create Text Message Example

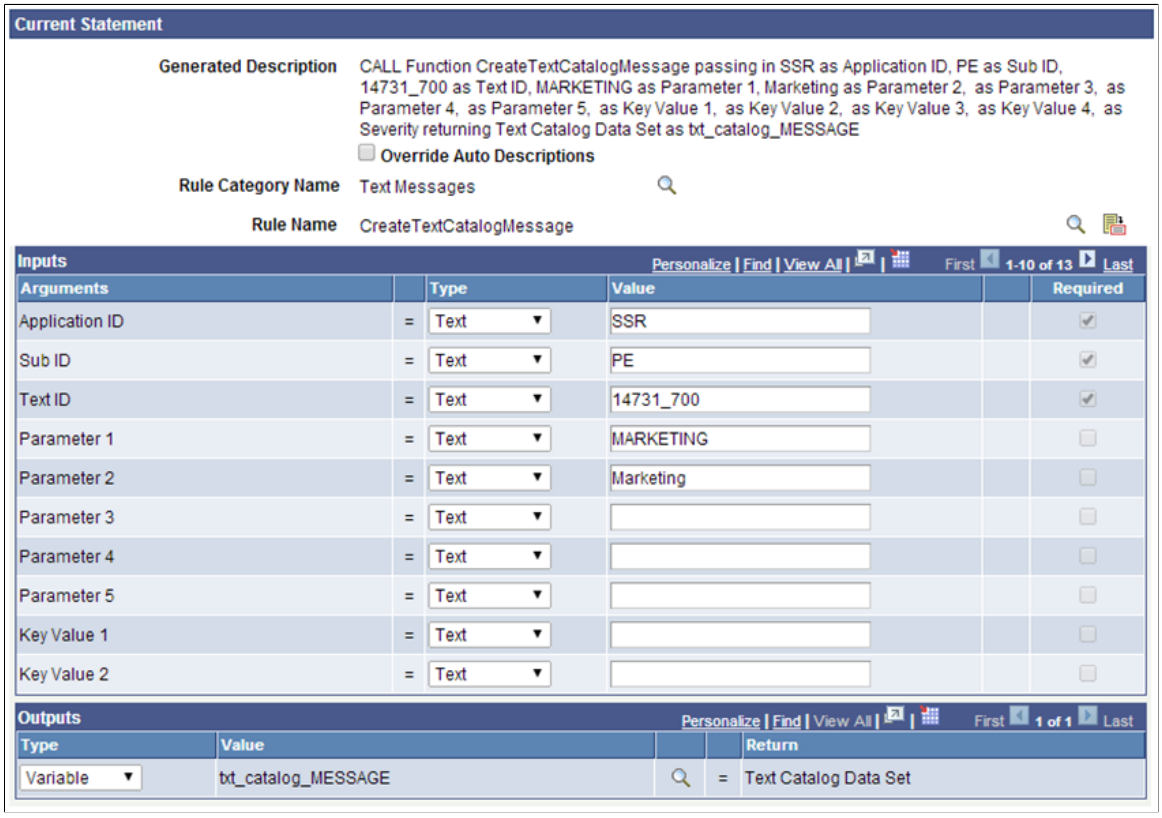
This example illustrates the fields and controls for the Configure Text Catalog Page for Rules Engine Create Text Message Example.



This is an example of a call to a Function that retrieves the Text Catalog:

Image: Example of a Function Call to the Text Catalog for Rules Engine Create Text Message

This example illustrates the fields and controls for an example of a Function Call to the CreateTextCatalogMessage rule that uses the Text Catalog.



To retrieve a formatted text message supply the appropriate input defined as arguments:

Argument Name	***Details***	***How does this relate to setup***
Application ID	A Text value. This argument is required.	This corresponds to Application ID in the Maintain Text Catalog set up from which a Text Message is retrieved.
Sub ID	A Text Value. This argument is required.	This corresponds to Sub ID in the Maintain Text Catalog set up from which a Text Message is retrieved.
Text ID	A Text Value. This argument is required.	This corresponds to the unique Text ID in the Maintain Text Catalog set up from which a Text Message is retrieved.
Parameter 1 through 5	Text. Optional	This value replaces Message Text variables %1 through %5 with a value. Provide a Variable, Text Value or (data set) property.

Argument Name	*Details*	*How does this relate to setup*
Key Value 1 through 4	Text, Optional	Context Keys can be used for granular selection of Context Catalog Text's added to a Text Catalog ID.
Text Catalog Data Set	Required	Provide a Data Set Variable which references system Data Set "Text Catalog Data Set".

Evaluation Management System (EMS) Category Rules

Rules in the following category are delivered as examples to demonstrate the use of an automated rating scheme as part of an Evaluation. All Rules created which interact with the Evaluation records should be built under this category to take advantage of the delivered data sets and Rule group which support Rules processing for Evaluations.

Rule ID	*Name*	*Long Description*
SCC_RULE_ID_20140306170755	Rating Scheme Calculation Example	This Rule can be used at the Evaluation Rating Scheme level as an alternate calculation of the Overall Rating value for the scheme. This example returns the sum of the component rating values.
SCC_RULE_ID_20140303114951	Test Score Rule Example	This Rule returns a rating value for the rating component based upon the highest score value in a comparison of ACT scores and SAT I scores. Calls a number of Rules to get the scores from the test score records, evaluate those results against a rating scale to obtain the rating value to populate the rating component.
SCC_RULE_ID_20140306173039	Academic Qualification Example	This Rule returns a rating value to the rating component based upon education data fields of percentile rank and converted GPA contained in External Academic Summary. Calls a number of other Rules which get the GPA and percentile values, evaluate those results against a rating scale to obtain the rating value to populate the rating component.
SCC_RULE_ID_20140306172622	Courses Completed Requirement Example	This Rule returns a rating value to the rating component based on a count of completed subjects in External Academic Subjects.
SCC_RULE_ID_20140304120320	Get ACT Rating Value Example	Returns the rating value for an ACT score.
SCC_RULE_ID_20140306141320	Get Course Count Rating Example	Gets the rating value for Course Count.

Rule ID	Name	Long Description
SCC_RULE_ID_20140306143945	Get Course Requirement Rating Example	Returns a rating value based on the number of Courses Completed for all Academic History entries that have External Academic Data rows where External Career = HS and Transcript Type = OFF.
SCC_RULE_ID_20140306161125	Get Courses Completed Count Example	Returns a count of completed courses in External Academic Subjects.
SCC_RULE_ID_20140304123947	Get GPA Rating Example	Returns the rating value for a GPA.
SCC_RULE_ID_20140304162158	Get High School Rating Example	Gets the highest rating value of the Converted GPA and Percentile values and passes to Get High School Rating Driver Rule.
SCC_RULE_ID_20140304170413	Get High School Rating Driver Example	Loops through External Academic Data where External Career = HS for an EMPLID and calls Get High School Rating. Returns the highest High School Rating.
SCC_RULE_ID_20140226160705	Get Highest Test Component Score Example	Returns the highest score for the specified EMPLID, TEST_ID, and TEST _COMPONENT.
SCC_RULE_ID_20140304155504	Get Percentile Rating Example	Returns the rating value for a Percentile Rank.
SCC_RULE_ID_20140304121526	Get SAT I Rating Value Example	Returns a rating value for a SAT I Test Score Total.
SCC_RULE_ID_20140226155857	Get Test Component Scores Example	Returns a list of test component scores for the specified EMPLID, TEST_ID and TEST_COMPONENT.

Research Self Service Task Category Rule

The Rule in this category can be used to Return Notification recipient EMPLID for the Research Candidate in context.

Rule ID	Name	Description
SCC_RULE_ID_20140115033240	Research Std SS task notification recipient	Returns the Active Primary Supervisor ID for the Research/Project Candidate in context.

Service Request Functions Category Rules

Rules and Functions in this category can be used to Return information relating to the Research Service Request in context.

Rule ID	Name	Description
SCC_RULE_ID_20131017052110	Service Request Assignment: getSetup	Returns the Advisor Type, Default EMPLID from the Service Request Assignment Setup for the Institution, service request type and subtype.
SCC_RULE_ID_20131018043858	Service Request Assignment: getTYPESetup	Returns the Service Request Category for the given Service Request Type.
SCC_RULE_ID_20131017060725	Service Request Assignment: getCATSetup	Returns the “SR Assigned to” EMPLID from the Service Request Category Setup for the Service Request in context.
SCC_RULE_ID_20131017014805	Service Request Assignment:Research Advisor	Returns the “Advisor” EMPLID from the Student Advisor History for the Service Request in context.
SCC_RULE_ID_20131016061345	Service Request Assignment:Research Supervisor	Derive the Primary Research supervisor for the Service Request in context.
SCC_RULE_ID_20131017014916	Default Service Request Assignment Logic	Returns EMPLID to which the service request will be assigned.

Service Requests Category Rule

The Rules in this category can be used to derive AssignedTO EMPLID for the Research Service Request in context.

Rule ID	Name	Description
SCC_RULE_ID_20140205065106	Service Request AssignedTo : Rule Trigger	This Trigger Rule is called from Rule Trigger definition to derive Service Request AssignedTo in Research Service Requests Student and admin components.

Research Functions Category Rules

Functions in this category can be used to return Active Supervisors and Admin Affiliation Profiles for the Research Service Request in context.

Rule ID	Name	Description
SCC_RULE_ID_20140206024642	Match Topic Title	Matches Title of active Research Topics of given status with the passed Thesis Title parameter. If topic records exist but title doesn’t match, then it returns false, else true.
SCC_RULE_ID_20140120052534	Get Active Research Supervisors	Returns the List of active Research Supervisors for the candidate in context.

Rule ID	Name	Description
SCC_RULE_ID_20140120083423	Get Active Admin Profiles	Returns the List of active Research/ Project Admin Affiliation Profiles for the candidate in context.

Research Candidates Category Rules

Rules and Functions in this category can be used to return true/false with a list of messages for the Research Candidate in context. These Rules can be used in the Research Rule Types and Execution Event Context components to display the warning messages in the Research Components.

Rule ID	Name	Description
SCC_RULE_ID_20140116235214	Candidate has active Supervisors	Checks if the candidate has active Supervisors. Returns message if not found.
SCC_RULE_ID_20140120084414	Candidate has non available research Supervisors	Checks if Candidate has any Research Supervisors who are not available anymore. If so returns a message.
SCC_RULE_ID_20140121001258	Candidate has non available Project Supervisors	Checks if Candidate Project has any Project Supervisors who are not available anymore. If so returns a message.
SCC_RULE_ID_20140206012715	Compare Thesis Title	Checks if Candidate Thesis Title matches any of the active Approved Topic Titles. If Not, returns a message.

Notification Framework Category Rules

Using delivered functions, you can create a notification in a specific channel. The Notification Rule can be called by other Rules. This makes it possible to send notifications conditionally based on an evaluation or calculation result for a specific selection of students in your database.

Rule ID	Name	Description
SCC_RULE_ID_20140203094013	Announcement Notification	An Announcement Notification can be created for the channel Alert. (for example, an informational message that appears on a portal homepage). Note that an Announcement is an Alert created not for one Recipient but for all recipients.
SCC_RULE_ID_20131219093336	Email Notification	A notification can be created for the Email Channel. Use this Notification to send Emails to one or more persons using the TO, CC and BCC email options. Attachments can be included.

Rule ID	*Name*	*Description*
SCC_RULE_ID_20140211033859	Push Notification	A notification can be created for the channel Push. Push Notifications are created for mobile apps on iOS/Android.
SCC_RULE_ID_20140211080755	SMS Notification	A notification can be created for the channel SMS.
SCC_RULE_ID_20140211033801	Worklist Notification	A notification can be created for the channel Worklist. This Rule creates a Worklist Item on a Portal Homepage with actionable hyperlink.
SCC_RULE_ID_20140211092724	Alert Notification	A notification can be created for the channel Alert.(for example, Informational message that appears on a portal homepage).

Using Notification Framework Rules

To use the Notification Rules, you must complete the Notification Framework setup. To facilitate usage of Notification Rules, Oracle delivers a Notification Consumer Setup.

Rules Engine Notification Consumer ID

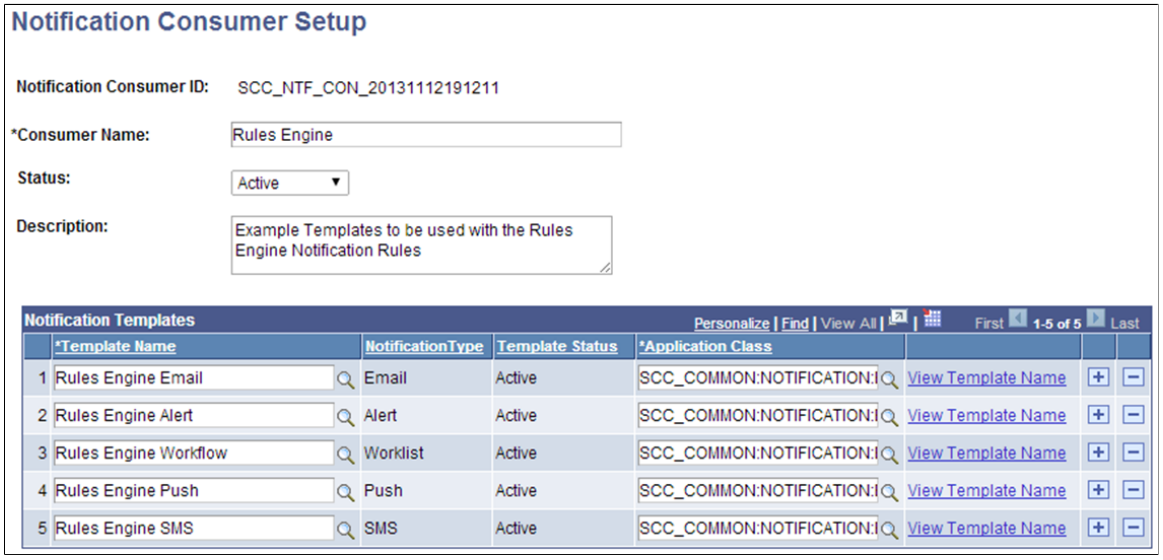
The Consumer ID controls which consumers can send Notifications through the Notification Framework. Oracle delivers a Notification Consumer ID for Rules Engine.

UID	*Name*	*Long Description*	*Purpose*
SCC_NTF_CON_ 20131112191211	Rules Engine	Notification Consumer for Rules Engine. Attached Templates are delivered as examples and can be used to send Rules Engine Notifications.	The following Consumer ID is used to send Rules Engine Notifications. The Consumer ID is linked to active Notification Rules.

Navigate to Set Up SACR, System Administration, Utilities, Notifications, Notification Consumer Setup.

Image: Notification Consumer Setup Page for Rules Engine Example

This example illustrates the fields and controls for the Notification Consumer Setup Page for Rules Engine Example.



The Rules Engine can create and send a Notification in these Channels:

- Email
- Alert
- Workflow
- Push
- SMS

There is a corresponding Notification Setup Template for each channel supported by Rules Engine. The associated application class is a dummy class which has been added as a placeholder. The application class does not contain any logic or functionality but must be added as a default application class for any Institution Template added to Notification Consumer Setup for Rules Engine.

Rules Engine Templates

Templates allow you to set up the recipients for the Notifications. A template has been created for each notification supported by Rules Engine. Oracle delivers these Template ID's for Rules Engine:

Template ID	***Name***	***Long Description***	***Associated Generic Template***
SCC_NTF_TMP_ 20131112191858	Rules Engine Email	Example Template for sending Rules Engine Notifications of type Email	SCC_RULES_ENGINE_ EMAIL

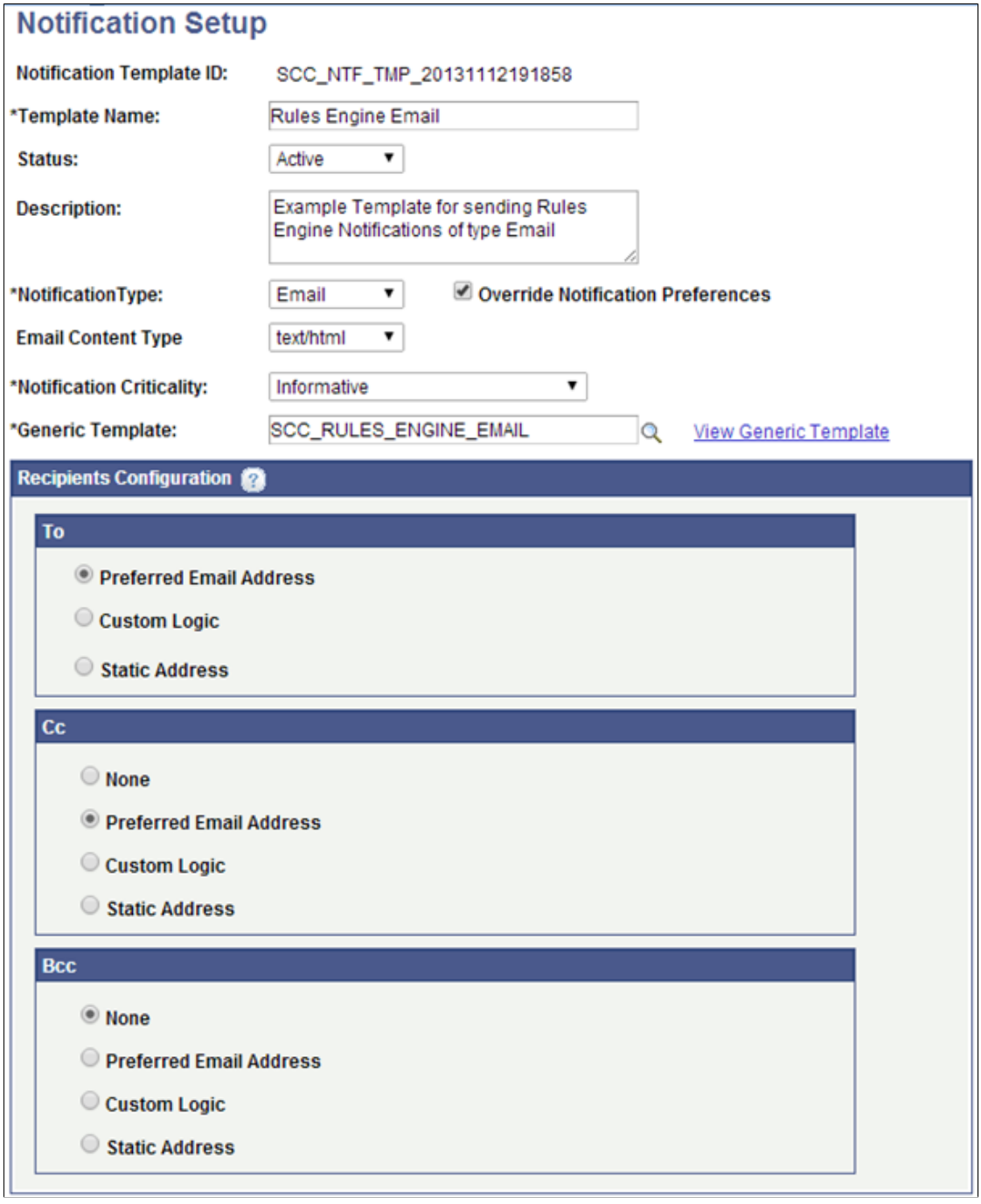
Template ID	*Name*	*Long Description*	*Associated Generic Template*
SCC_NTF_TMP_ 20140123062439	Rules Engine Alert	Example Template for sending Rules Engine Notifications of type Alert as well as Announcements	SCC_RULES_ENGINE_ ALERT
SCC_NTF_TMP_ 20140123070158	Rules Engine Workflow	Example Template for sending Rules Engine Notifications of type Worklfow	SCC_RULES_ENGINE_WL
SCC_NTF_TMP_ 20140204030502	Rules Engine Push	Example Template for sending Rules Engine Notifications of type Push	SCC_RULES_ENGINE_ PUSH
SCC_NTF_TMP_ 20140204030858	Rules Engine SMS	Example Template for sending Rules Engine Notifications of type SMS	SCC_RULES_ENGINE_SMS

Click on View Template Name on the Notification Consumer Setup Page or navigate to Set Up SACR, System Administration, Utilities, Notifications, Notification Setup to view delivered Notification IDs.

This screen shot shows the delivered Rules Engine Template for channel Email. You can adjust the Template to suit your needs; however, Oracle recommends that you create your own Templates.

Image: Notification Setup Page for Rules Engine Example

This example illustrates the fields and controls for the Notification Setup Page for Rules Engine Example.



Rules Engine Generic Templates

Each Template ID is associated with a Generic template. The Generic Template controls the Message Text and the Variables to include in the Notification Message. The attached Generic Template can be adjusted to suit Institution requirements or can be replaced by an institution specific Template. Access

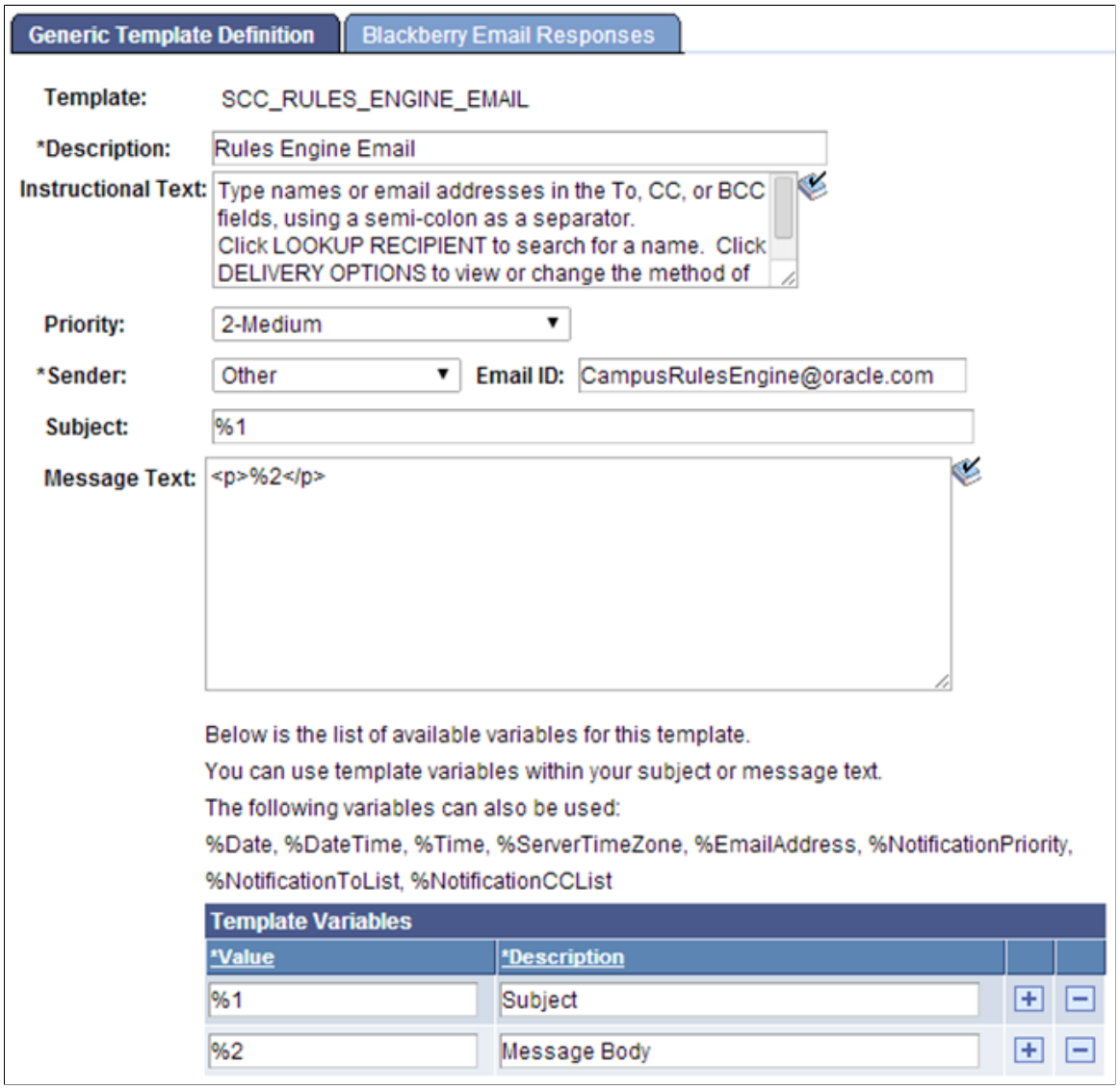
PeopleTools, Workflow, Notifications, Generic Templates to adjust Generic Templates or use the prompt option on the Template ID setup to access. Oracle delivers the following Generic Templates for Rules Engine:

Template ID	***Name***	***Template Variables***
SCC_RULES_ENGINE_EMAIL	Rules Engine Email	%1 Subject %2 Message Body
SCC_RULES_ENGINE_ALERT	Rules Engine Alert	%1 Subject %2 Message Body
SCC_RULES_ENGINE_WL	Rules Engine Workflow	%1 Subject %2 Message Body
SCC_RULES_ENGINE_PUSH	Rules Engine Push	%1 Subject %2 Message Body

Template ID	*Name*	*Template Variables*
SCC_RULES_ENGINE_SMS	Rules Engine SMS	%1 Subject %2 Message Body

Image: Generic Template Definition Example for Rules Engine

This example illustrates the fields and controls for the Generic Template Definition Example for Rules Engine.



Understanding the Interaction between the Notification Framework and the Notification Rule

The Rules delivered in the Notification Framework Category can be used to send a Notification in one or more notification channels. The following Rules are delivered:

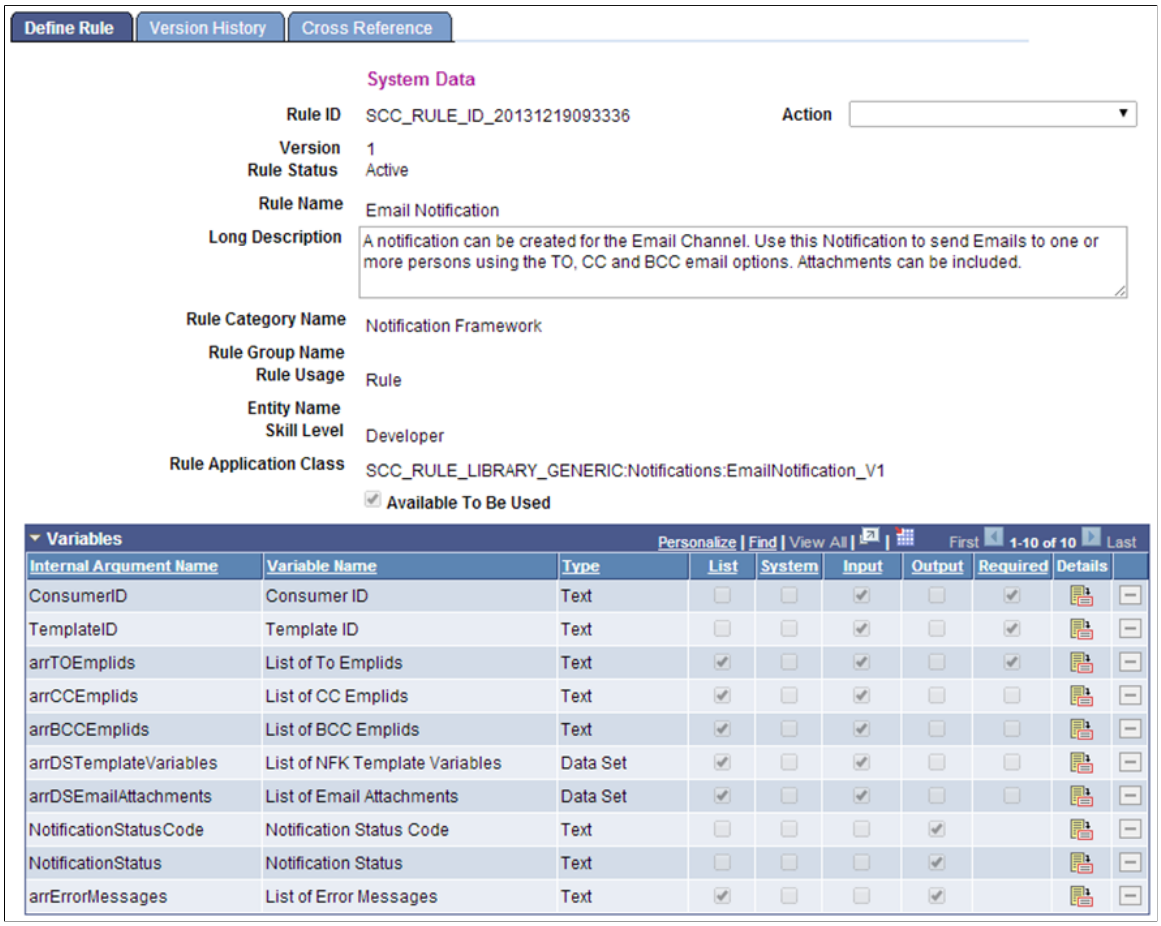
- Email Notification

- SMS Notification
- Alert Notification
- Announcement Notification
- Worklist Notification

Access the Rules Engine Manager to access Notification Rules (Set Up SACR, System Administration, Rules Engine, Rules Engine Manager, select Search for a Rule. Use the prompt to select Rule Category Name *Notification Framework.* Click Search and select the desired Notification Rule (Email, SMS, Alert, Push, or Worklist).

Image: Rules EngineDefine Rule Page for Rules Engine Notification Example

This example illustrates the fields and controls for the Define Rule Page for Rules Engine Notification Example.



When the Notification Rule completes, you should be able to review the Notification in the Notification Framework Admin Page *and* view the notification as output by channel (For example, by verifying that an email has been sent). The recipient can review the Notification via Self Service Notifications.

This table describes each variable and its relationship to the Notification Framework Rule functionality:

Variable Name	***Details***	***How does this relate to Notification Framework setup***
Emplid List of To Emplids List of CC Emplids List of BCC Emplids	EMPLID to which the Notification is to be sent. Valid for Notification Channels: ALERT, WORKFLOW, EMAIL, PUSH	The Rule uses the settings on the Notification Template to determine where the notification needs to be sent. A recipient is selected according to the settings which have been administered on the Notification Template. For example, Preferred Email address. For Template SCC_RULES_ENGINE_EMAIL this could be a Preferred Email Address, Custom Logic or a Static address. For Notification channel email a notification can be sent to the following recipients: • TO • CC • BCC • Other channels support only one recipient In the Notification Email Rule, it is required to provide an Emplid for the TO recipient. Emplid can be provided for recipients CC and BCC. Multiple recipients can be added for the Email channel.
List of Email Attachments Attachment File Name	File Name to be added to email Valid for Notification Channels: EMAIL	The Notification Rule for channel EMAIL allows you to specify an attachment file Path and File Name. The File name is retrieved from the specified file Path, added to the email sent by the Notification Framework, and sent to the listed Recipients. **Note:** Although not required, when an Attachment file name is specified, the Attachment file path should also be specified and vice versa. Multiple attachments can be added

Variable Name	***Details***	***How does this relate to Notification Framework setup***
List of Email Attachments Attachment File Path	File Location of the File which is to be added to email Valid for Notification Channels: EMAIL	Refer to Attachment File Name.
Consumer ID	The Notification Consumer ID of the consumer sending the notification. Valid for all Notification Channels.	A LOV prompt is attached to this LOV so that only the Consumer ID Rules Engine can be selected.
Template ID	The Notification Template ID of the template to be used for the Notification. The Notification Template holds information about the notification type and recipients configuration. Valid for all Notification Channels	A LOV Prompt is to this LOV so that only templates relevant for Consumer ID Rules Engine can be selected. The Notification Template controls the Notification Type and Recipient information.
Template Variables	The names of the Template Variables which are set up on the Notification Template as Variables. The Variable Names and Order should match those of the Template (for example %1). Valid for all Notification Channels.	Template Variables can be added to a List Variable. The Template Variables added to the list should correspond to the Variables which have been created on the Generic Notification Template for this specific Notification. The Variables are used to provide the Email with appropriate text for the Message Subject as well as body.
Template Variable Values	The values to be placed into the Variables which have been defined on the Notification Template. Valid for all Notification Channels.	Variables can be added to this List Variable. The Variables are used to populate the Template Variables as they have been created on the Generic Template. It is possible to add hard-coded text as well as Variables to the Template Variable Values list. The order in which the text and variables are added to the list must match the order of the Template Variables list.

Variable Name	*Details*	*How does this relate to Notification Framework setup*
Notification Type Created	If the Notification has been generated successfully, the Notification type is filled. Valid for all Notification Channels.	Possible Values: • ALT: An Alert has been sent to the Notification Framework. • EML: An email has been sent to the Notification Framework. • PSH: A Push notification has been sent to the Notification Framework. • SMS: A SMS has been successfully sent to the Notification Framework. • WKL: A Worklist has been successfully sent to the Notification framework. **Note:** The Rule itself does not create the Notification but sends a Notification to the Notification Framework which in turn handles the actual Notification. The Rule can only indicate whether the Notification was handed to the Notification Framework using the appropriate channel. The Rule cannot determine whether the notification (for example email) was sent successfully.
Error Message	The Notification was not generated successfully Valid for all Notification Channels.	If the notification has not been generated successfully the following output field contains an error message. Again this only concerns an error which may have occurred passing values to the Notification Framework.
Component	Component Name Valid for Notification Channels: WORK LIST	The component to which the Notification applies. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be : SCC_BIO_DEMO.

Variable Name	*Details*	*How does this relate to Notification Framework setup*
Menu Name	Menu Name Valid for Notification Channels: WORK LIST	The Menu name to which the Notification applies. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be : CC_BIO_DEMO_DATA_ STDNT.
Menu Bar Name	Menu Bar Name Valid for Notification Channels: WORK LIST	The Menu Bar name to which the Notification applies. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be : USE.
Menu Item Name	Menu Item Name Valid for Notification Channels: WORK LIST	The Menu Item name to which the Notification applies. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be : SCC_BIO_DEMO .
Page Name	Page Name Valid for Notification Channels: WORK LIST	The Page name to which the Notification applies. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be : SCC_BIO_DEMO_PERS.
Mode	Mode Valid for Notification Channels: WORK LIST	The Mode in which the component should be opened. For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be: U (Update/Display).
Market	Market Valid for Notification Channels: WORK LIST	The Market to which the environments portal applies For a notification referring to campus community, Personal Information (Student), Add/Update Person, Biographical Details the value would be GBL.

Variable Name	***Details***	***How does this relate to Notification Framework setup***
URL	URL Valid for Notification Channels: WORK LIST	The generated URL which is used by the Notification for the corresponding notification email. This is an example of the value for a generated notification URL referring to Campus Community, Personal information (student), Add/Update Person, Biographical Details: http://<EnvironmentName>/EMPLOYEE/HRMS/c/CC_BIO_DEMO_DATA_STDNT.SCC_BIO_DEMO.GBL?Page=SCC_BIO_DEMO_PERS&ACAD_CAREER=UGRD&EMPLID=<StudentID>&Action=U

Testing the Notification Rule

In this example, the Email Notification Rule is used. Select the *Test Rule* Action as shown below:

Image: Rules Engine Tester Page for Rules Engine Notification Example

This example illustrates the fields and controls for the Rules Engine Tester Page for Rules Engine Notification Example

Variable	***Instructions***
List of To Emplids	Provide an Emplid or multiple Emplids. The Emplid selected should have the correct setup in place to receive this notification. For Example, in order to send an email to this EmplID, a valid EmplID must have been entered as Preferred Email address (for example via Campus community, Personal information (student), Biographical (student), addresses/phones, Electronic addresses).
Consumer ID	Use provided prompt to select the Consumer ID for Rules Engine.
Template ID	Use the provided prompt to select a Template ID which has been created for Consumer ID Rules Engine.

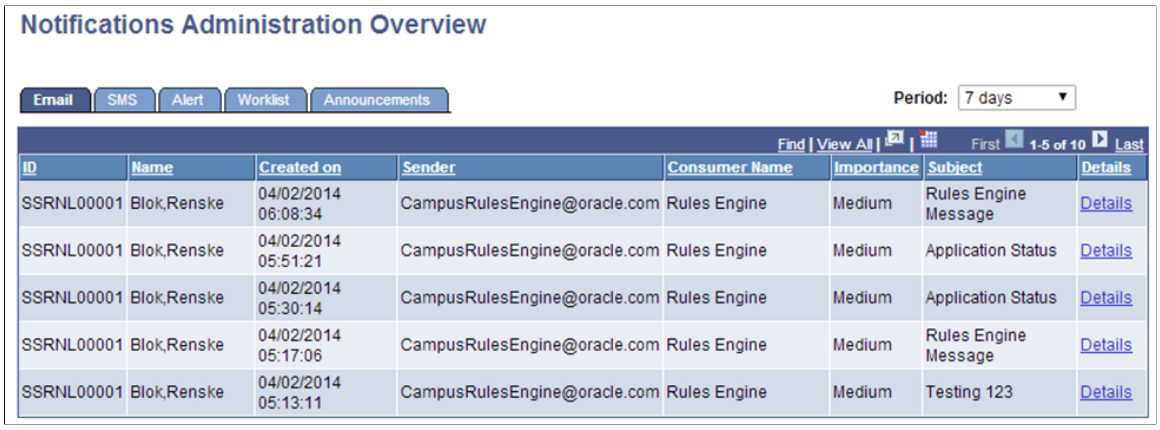
Variable	*Instructions*
List of NFK Template Variables	Use the provided prompt to provide a list of Template Variables. **Note:** When using the Tester you must provide Text values. When calling the Notification Rule from another Rule, you can provide Variables for all or some list Values.

Click Execute Test to test the Notification Rule. The Outputs grid displays information from the Notification Framework indicating the successful handling of the Notification; such as the Notification status Code, the Notification Status, and any relevant Error Messages.

After the Rule completes Generated notifications can be viewed via the Notification Admin component Open the Notification Admin component (Campus Community, Notifications, Admin Notifications). Select the appropriate Notification to view:

Image: Notifications Administration Overview Page for Rules Engine Notification Example

This example illustrates fields and controls for the Notifications Administration Overview Page for Rules Engine Notification Example.



ID	Name	Created on	Sender	Consumer Name	Importance	Subject	Details
SSRNL00001	Blok,Renske	04/02/2014 06:08:34	CampusRulesEngine@oracle.com	Rules Engine	Medium	Rules Engine Message	Details
SSRNL00001	Blok,Renske	04/02/2014 05:51:21	CampusRulesEngine@oracle.com	Rules Engine	Medium	Application Status	Details
SSRNL00001	Blok,Renske	04/02/2014 05:30:14	CampusRulesEngine@oracle.com	Rules Engine	Medium	Application Status	Details
SSRNL00001	Blok,Renske	04/02/2014 05:17:06	CampusRulesEngine@oracle.com	Rules Engine	Medium	Rules Engine Message	Details
SSRNL00001	Blok,Renske	04/02/2014 05:13:11	CampusRulesEngine@oracle.com	Rules Engine	Medium	Testing 123	Details

Calling the Notification Rule

The Notification Rule can be called from another Rule. This allows you to generate Notifications in a notification channel for multiple students which are selected based on the logic you have created in the calling Rule. The following is an example of a Rule calling the Notification Rule. This simple Rule sends a notification to active applicants in a selected career: The call statement has been added simply

by selecting the Notification Rule and providing required Input values. The prompt functionality for Consumer ID and Template ID also works on the call statement.

Image: Define Rule Page for Notification Rule Call Example

This example illustrates the fields and controls for the Define Rule Page for Notification Rule Call Example.