

# Oracle® Policy Modeling User's Guide

User Assistance and How-To Guide

Release 10.4.5

E54499-01

May 2014

Browse Policy Modeling User's Guide	
What's new	Quick links <b>NEW!</b>
Introducing Oracle Policy Modeling	Getting assistance
Projects and files	Writing rules
Designing and maintaining rule documents	Languages
Variables and constant values	Data model
Rules using entity instances	Temporal reasoning
Interviews and flows	Decision reports
Compiling and building	Finding and reporting
Analysis	Test cases
Debugging	Deployment
Custom functions and programming	Collaborating
Integrating	Accessibility
Reference	

V10.4.5

Copyright © 2009, 2014

Oracle Support

# What's new in Oracle Policy Modeling V10

## Version 10.4.0

### Modules

Oracle Policy Modeling now enables you to combine policy models from multiple teams or projects, and easily update them as needed. This reduces the risk of large policy automation projects and means that common policies across projects/divisions/systems can be readily shared. To do this you build your rulebase as a module, and then link to this module from other projects.

#### See also:

- [Create rules that can be shared with another project](#)
- [Include rules defined in a separate project](#)

### Entity and relationship creation changes

The user interface for defining data models in Policy Modeling projects has been simplified. Creating an entity in a properties file is now a quick one-step process that automatically sets up the containment relationship, as well as the identifying attribute for the entity. (The default containment relationship, and the default identifying attribute, can later be edited.) The user interface for editing containment and reference relationships is now consistent.

#### See also:

- [Define an entity](#)
- [Define a relationship](#)

### Inferred entity instances

In Oracle Policy Modeling you can now write policy models that determine which entity instances must exist. This means that there is no need to pre-create each entity instance that might be needed. Decision reports show why entity instances have been created.

#### See also:

- [Write rules that infer relationships and entities](#)

### Batch processor

The batch processor is a replacement for the Data Source Connector and allows a large number of 'cases' to be processed in batch. It has made it possible to easily analyze a policy model to see the results it will yield (using What-If Analysis documents). The batch processor can also be used to generate test scripts from existing Excel data.

#### See also:

- [Conduct what-if analysis using an Excel workbook](#)
- [Create test scripts from existing data](#)

## Testing coverage

There is a new report, Test Script Coverage, that enables you to measure the coverage of the test cases in a project. This helps to identify how you could improve the overall coverage of a test suite.

### See also:

- [Measure the coverage of a test suite](#)

## Other changes

- The version of BI Publisher used by Oracle Policy Automation has been upgraded from v.11.1.1.3 to v.11.1.1.7 (as of release 10.4.5)
- Displaying unformatted number values - you can now specify in the Attribute Editor that a number variable is to be displayed unformatted (eg in decision reports in the debugger and Web Determinations). See [Formatting of attribute values](#) for more information.
- Checkboxes in interviews - boolean values can now be collected using checkboxes. See [Customize interview user input options](#) for more information.
- Relationship filtering in interviews - you can now filter the list of entities that are available when collecting relationships, so that the user avoids seeing invalid options. See [Filter the list of available target entities](#) for more information. You can also now specify a visibility attribute for reference relationship controls (this works in the same way as [visibility options](#) for other control types).
- Sortable Project Explorer - folders and files in the Project Explorer can now be sorted alphabetically. This is an option called **Sort project explorer** under **Project Properties**. This option is on by default for new projects, and off when a project is upgraded from a previous version. (When turned off, folders and files will appear in the order that they were defined in the project file.)
- Warning when leaving a screen in Web Determinations - when you try to navigate away from an interview screen without submitting data, you are now shown a warning and get a chance to cancel, so that you don't unexpectedly lose the data you have entered.
- The accessibility of the Oracle Web Determinations (OWD) user interface has been improved (as of release 10.4.5). For more information, see [Accessibility features in Oracle Web Determinations](#).
- The Social Services Screening Example Project has been updated to reflect recent changes in the underlying legislative and policy rules, with new screens added to collect additional information now required as a result of the rule changes. New test cases have been added and the visualizations have also been updated. To see the updated Social Services Screening project, follow the instructions in [Open an example rulebase](#).
- Documentation of the following has been improved:
  - the process for doing a bulk import of expected test results - for more information, see [Specify expected results](#).
  - the different acceptable ranges of values for date variables in Oracle Policy Modeling and Oracle Web Determinations - for more information, see [Use constant values in rules](#).
  - using Microsoft Team Foundation Server for source control of Oracle Policy Modeling projects. For more information, see [Track rulebase changes on multi-developer projects](#), in particular [Install Microsoft Team Foundation Server](#).
  - confining the use of regular expressions to text variable attributes only - for more information, see: [Use regular expressions](#).

## Version 10.3.0

### BI Publisher integration

In Oracle Policy Modeling you can now use BI Publisher to author document templates in Word which can be used with Oracle Web Determinations to generate interview documents.

#### See also:

- [Overview: The process of creating an interview document](#)
- [Develop a template for an interview document](#)
- [BI Publisher code for Oracle Policy Modeling](#)

### Language support

Syntactic parsers are new for the following languages: Italian, Japanese, Portuguese (Brazilian), Portuguese (European), Russian. From the Help menu in Oracle Policy Modeling you can now access a list of available languages. This lists each language parser with its version number and the type of parser (ie [Syntactic](#) or [RLS](#)).

Translations, which are included in an Excel translation document, can now be marked as not requiring translation using an **Ignore Translation** button on the Oracle Policy Modeling toolbar. This is useful where the translation of an item is intended to be the same as in the original rule language (ie it is language-independent).

There are special considerations that need to be taken into account when writing rules in particular non-English languages. Documentation has been provided which explains what is supported (ie sentence structures and verb forms) and any limitations that you need to be aware of when writing rules in these languages.

#### See also:

- [Write rules in Arabic](#)
- [Write rules in Finnish](#)
- [Write rules in French](#)
- [Write rules in Italian](#)
- [Write rules in Japanese](#)
- [Write rules in Portuguese](#)
- [Write rules in Russian](#)
- [Write rules in Spanish](#)
- [Write rules in Turkish](#)

## Version 10.2.0

### Entity-level summary screen goals and other screen authoring changes

Goals, screen flows and labels which operate at entity level may now be added to a summary screen, by creating a summary screen folder associated with the relevant entity.

Entity-level attributes may also be used for visibility, dynamic default, mandatory and read-only settings.

Public names may now be assigned to screen flows.

**See also:**

- [Add entity-level items to the summary screen](#)
- [Customize interview user input options](#)
- [Hide, display and disable an interview screen element](#)
- [Define interview screen flow](#)

## Language support

Translations can now be added to a rulebase via an Excel translation document. The document is created by Oracle Policy Modeling and populated with all rulebase strings needing translation. The translations can then be filled out in the Excel document and compiled to produce a translated rulebase that may be run in Oracle Web Determinations.

The concept of the project locale has now been divided into a separate rule language and region, to better handle deployments involving one language used across multiple regions and vice versa. The data entry formats in Oracle Policy Modeling, Word/Excel rules, the debugger and Oracle Web Determinations are now restricted in line with this (essentially for Oracle Policy Modeling and the debugger to require basic format, and Web Determinations according to the rulebase region setting).

**See also:**

- [Create a new language translation for a rulebase](#)
- [Write rules in other languages](#)
- [Formatting of variable values](#)
- [Use constant values in rules](#)
- [Localize interview help \(commentary\)](#)

## Containment

Containment relationships are now an integral part of the data model for a rulebase. All entities must be defined within the context of a containment relationship, such that the network of containment relationships in the rulebase represents the main data structure of the rulebase. Additional relationships between entities are defined as reference relationships as required. Singleton entities have now been fully deprecated.

Projects created in versions of Oracle Policy Modeling prior to 10.2 are upgraded automatically when opened. Containment relationships are defined for the upgraded rulebase based on a number of [principles](#) applied to the old relationship structure of the rulebase pre-upgrade.

**See also:**

- [Upgrade a project](#)
- [Understand containment relationships and entity completion](#)
- [Define an entity](#)
- [Define a relationship](#)
- [Set up entities and containment relationships in the debugger](#)

## Updated Excel functionality

The rules generated from Excel decision tables have been optimized to produce smaller and more efficient rules. In particular, the way merged conclusion cells are interpreted has been revised, so that any condition row proving a conclusion in a merged cell can evaluate in any order. The usual "top-down" evaluation order applies to rows if their conclusion cells are not merged.

It is now also possible to use entity attributes in Excel decision tables, and to use most entity functions.

### See also:

- [Prove the same set of conclusions using multiple conditions](#)
- [Allow rule conditions to evaluate in any order and handle missing values](#)
- [Use entity attributes in an Excel rule table](#)

## Rule looping

Rule loops are now permitted as a valid part of a rulebase. A normal rule may be defined as a rule loop by using a Configuration element for the rule. Loops may also now be created between attributes proved in shortcut rules.

### See also:

- [Model loops in rule logic](#)
- [Capture implicit logic in rules](#)

## New functions

The following new text functions are available:

- [Contains](#): checks if a text string contains a particular substring
- [StartsWith](#): checks if a text string contains a particular substring at the start of the string
- [EndsWith](#): checks if a text string contains a particular substring at the end of the string
- [IsNumber](#): checks if a text string is a number
- [Length](#): finds the length of a text string

The following new temporal functions are available:

- [TemporalIsWeekday](#): determines whether each day in a specified range is a weekday
- [TemporalOncePerMonth](#): returns a temporal boolean value whose value is true only on a given day of the month

### See also:

- [Text function rule examples](#)
- [Temporal reasoning function rule examples](#)
- [Localized function references \(all languages\)](#)

## Preview screen

A Preview option is now available during the development of question screens which quickly and easily displays the question screen as it will appear in Oracle Web Determinations, without needing to complete an interview. If a debug session already exists, any data from the session is used to display the screen preview.

**See also:**

- [Preview a question screen](#)

## Custom function definition

Custom functions may now be called in the same way as any of the built-in functions in Oracle Policy Modeling, with input and return values defined entirely by the custom function implementation.

**See also:**

- [Write a rule that uses a custom function](#)

## 'Currently known' operator

A new operator 'currently known' is now available, to test whether or not an attribute has a value, without causing it to be investigated in the question search.

**See also:**

- [Certain and known operator rule examples](#)
- [Localized function references \(all languages\)](#)

## Native Subversion support

Subversion is now integrated directly into Oracle Policy Modeling. This provides access to rule file history and version comparisons.

**See also:**

- [Track rulebase changes on multi-developer projects](#)
- [Track versions of rulebase documents](#)
- [Retrieve a specific document version](#)

## Persist temporal visualization view

The temporal visualization view in the debugger will now persist when the debugger is restarted, and when the project or Oracle Policy Modeling is closed and reopened.

**See also:**

- [Visualize temporal data](#)

## Configure attribute validation messages

The error message shown for maximum/minimum/regular expression validations on an attribute can now be specified in the Attribute Editor.

**See also:**

- [Validate user input using errors and warnings](#)

## Configure add/remove entity instance buttons

The text used for the Add Instance and Remove Instance buttons on entity collect screens in Oracle Web Determinations can now be specified in the Screen Editor.

### See also:

- [Define a screen for collecting entity instances](#)

## Locate in Explorer option

A "Locate in Explorer" option is now available on rulebase files in the Project Explorer, to open the selected file's folder in Windows Explorer.

### See also:

- [Locate a rulebase file in Windows Explorer](#)

## Version 10.1

### Build and continue in Debugger

When you restart a debugger session now, you have the option to retain current session data. The rulebase will be built and then the debugger will restart and attempt to reload the old session data into the new debugger session. Data for an attribute, entity or relationship will only be lost if it has changed text and public name.

### See also:

- [Change a rule while debugging](#)

### Access to localized function references

The Function Reference list which is available from the Help menu in Oracle Policy Modeling now includes the description of the function in the native language.

The Function Reference list is also now available and searchable in the Oracle Policy Modeling User's Guide.

The rule authoring experience in Word and Excel is now fully localized for every syntactic and non-syntactic parser language.

A complete function reference is also available for every language from the Help menu.

### See also:

- [Localized function references](#)

### Command-line support for regression tester

Command-line support has been added for the regression tester. The C# project "RegressionTester.CmdLine.exe" within the regression tester solution provides an executable that allows a rulebase project's test scripts to be executed from the command line.

### See also:

- [Use the regression tester from the command line](#)

## InstanceValueIf function

*InstanceValueIf* is a new function for Oracle Policy Modeling. The rule author can now get a value from a unique entity instance, identified from the target entity instances of a relationship by a condition.

If the condition identifies a single target entity instance, then the value is the value calculated against that entity instance.

If more than one target instance meets the condition, then Uncertain is returned.

If no target instances meet the condition and the relationship is known, then the value is Uncertain.

### See also:

- [Entity and relationship function rule examples](#)

## Auto-include additional files at build time into rulebase .zip

There are files that are useful to include in the rulebase zip, such as configuration for custom functions and commentary for Web Determinations. These files can now be placed inside a folder called "include" in the project directory and they will be automatically added to the rulebase zip at build time.

This was previously possible by copying those same files into the "output" folder but this meant the output folder included a mixture of generated files and source files. Now the output folder can be safely excluded from source control, and deleted to ensure old output files are not left lying around.

### See also:

- [Include extra files in the build](#)

## Version 10.0

### Inferred relationships

Enhancements have been made to enable rule authors to:

1. Reason about multiple entities in the same rule (cross entity reasoning), and
2. Infer relationships through rules.

In previous versions, it was only possible to write rules while referring to one entity at a time. With an entity function, such as *Exists* or *ForAll*, the rule author could reason across a single relationship to the target entities, but these functions had a very limited application.

In Oracle Policy Modeling 10.0, rule authors can now reason across several different entities within the "scope" of a single rule. This is done using extended forms of the *For*, *Exists* and *ForAll* functions. Each function works like its older equivalent, but the boolean proof for the function is pushed to a subsidiary rule level.

The other new feature is the ability to conclude relationships. Previously, relationships were statically defined for a set of session data. A rule author can now create a new type of relationship, an inferred relationship, and can then infer the source and targets of those relationships using rules.

NOTE: Any V10.0 rulebases that use inferred relationships will need to be recompiled for V10.1.

### See also:

- [Reason about the relationship between two entities](#)
- [Entity and relationship functions](#)

- [Investigate an inferred relationship](#)

## Reasoning with partial knowledge

Reasoning with partial knowledge improves the reasoning done by Oracle Policy Modeling in situations where attributes or relationships used in a rule are unknown. In some of these situations, where not all, but enough information is known it is now possible to receive valid conclusions from a rulebase query. Previously the result of such queries would simply have been unknown.

See also:

- [Make a decision when only some data is known](#)

## Time of day and data and time data types

In order to provide more fine-grained operations on dates, a date-time attribute type, and a time of day attribute type have been added to Oracle Policy Modeling.

- Time of day - this is a string formatted as hh:mm:ss. For example,

**the specified start time for the employee = 07:47:31**

- Date time - this is a string formatted as yyyy-MM-dd hh:mm:ss. For example,

**the submission date time = 2009-08-12 17:30:00**

**See also:**

- [Get a date and time](#)
- [Get a time, second, minute or hour](#)
- [Count periods between two dates or times](#)
- [Time of day functions](#)
- [Date and time functions](#)

## Screen flow functionality

The use of Microsoft Visio for creating screen flows has been replaced with in-built screen flow functionality. Screen flows are now created and authored entirely within Oracle Policy Modeling.

**See also:**

- [Define interview screen flow](#)

## Updated commentary generation

Oracle Web Determinations 10.0 is able to serve commentary files without extra configuration when the commentary files are included in the deployed rulebase archive. Commentary for screens is now supported in addition to the attribute commentary which was previously supported. Oracle Policy Modeling now creates placeholder commentary files in the rulebase include directory, ready for modification, which will be archived together with the rules and screens each time the rulebase is built. Running Web Determinations in the debugger now shows the commentary as it will appear by default in a production environment.

**See also:**

- [Create, update or delete interview help \(commentary\)](#)

## Other changes:

- Build and run with Oracle Determinations Server - there is now the option to [run with Determinations Server](#) when building a rulebase.
- Debugging Oracle Web Determinations for .NET - Oracle Policy Modeling now supports debugging Oracle Web Determinations for .NET. This is available through the Debug Options dialog as part of the capability to attach to an existing Oracle Web Determinations Website. (This capability is not specific to .NET and applies equally well for connecting to an existing instance of Oracle Web Determinations for Java.) See [Use Oracle Web Determinations in the debugger](#) for more information.
- New rulebase list provider - the rulebase list provider is a new out-of-the-box alternative to the static selectable list options for input controls given through Oracle Policy Modeling. It provides the ability to package list files, in a strict XML format organized by locale, along with the rulebase archive. See [Source list contents from an external file](#) for more information.
- Single file rulebase deployment - building a project in Oracle Policy Modeling will now automatically build a <project>.zip file in the output folder. This package of all of the individual output components of a rulebase is the preferred method of deploying rulebases rather than as individual files. (NOTE: Any other files placed into the output folder will also automatically be included as part of this zip file, so unless the documentation explicitly directs you to, you should **not** put anything into the output folder.)
- Rebranding - the product formerly known as Haley Office Rules 2008 has been rebranded to become Oracle Policy Modeling 10.0. The documentation has been updated accordingly.
- Handling of many-to-many relationships in the debugger - in previous versions, many-to-many relationships were handled differently in the debugger to one-to-many, many-to-one and one-to-one relationships. If a many-to-many relationship was modified, the reverse relationship was not updated, as was the case with the other three relationship forms. This was in part due to a lack of support for partial knowledge. In Oracle Policy Modeling 10.0 many-to-many relationships are handled exactly the same as other relationships in the debugger. When a many-to-many relationship is modified, the reverse relationship will also be updated.
- Support for rule authoring in any language - an Oracle Policy Modeling project can now be created using a language parser developed using the Rapid Language Support Tool. For more information, see [Create a new project](#).

## Quick links

- [Rule syntax reference](#)
- [Screen flow syntax](#)
- [BI Publisher syntax](#)
- [Keyboard shortcuts](#)
- [Choose attribute text](#)
- [Write rules in Word](#)
- [Write rules in Excel](#)
- [Create an entity](#)
- [Create a relationship](#)
- [Use an entity or relationship in a rule](#)
- [Temporal reasoning](#)
- [Model the structure of legislation](#)
- [Design an interview](#)
- [Design a decision report](#)
- [Test a rule](#)
- [Polish a rulebase](#)
- [Deploy an interview to Web Determinations](#)
- [Using source control](#)
- [Example rulebases](#)
- [Upgrade a project](#)
- [Rule principles for OPM](#)
- [Modify the look and feel of Oracle Policy Modeling](#)

## Getting assistance

Topics in Getting Assistance:

- [How to use Oracle Policy Modeling User's Guide](#)
- [Create and deploy a rulebase](#)
- [Example rulebases](#)
- [Get trained in Oracle Policy Modeling](#)
- [Access further resources on Oracle Policy Automation](#)

See also:

- [Modify the appearance or layout of Oracle Policy Modeling](#)
- [Keyboard shortcuts for Oracle Policy Modeling](#)

### How to use Oracle Policy Modeling User's Guide

The Oracle Policy Modeling User's Guide is a comprehensive source of information relating to the use of Oracle Policy Modeling.

#### What do you want to do?

[Find information using the Contents](#)

[Find information using the Search](#)

[Find information using the Glossary](#)

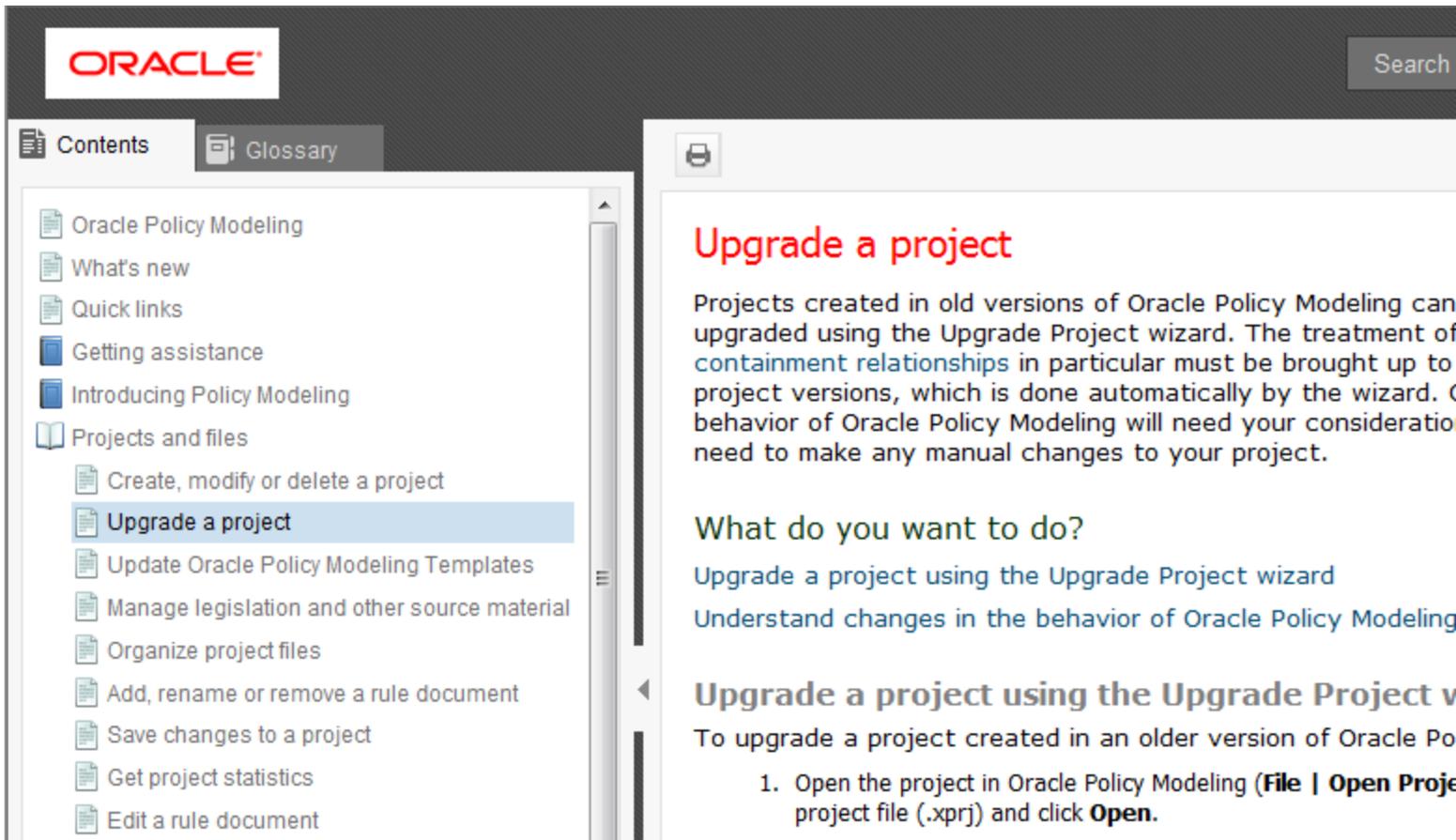
[Access the Oracle Policy Modeling User's Guide in another language](#)

#### Find information using the Contents

The Oracle Policy Modeling User's Guide is organized into sections corresponding to the main rule development tasks in Oracle Policy Modeling.

You can see these sections at any time by clicking on the **Contents** tab located in the left hand pane.

The topic you are viewing at the time will be highlighted in the Table of Contents:



### Find information using the Search

You can use the **Search** function to search for information that you are interested in. Click in the **Search** field at the top right hand side of the window and enter a keyword related to the information you are looking for.

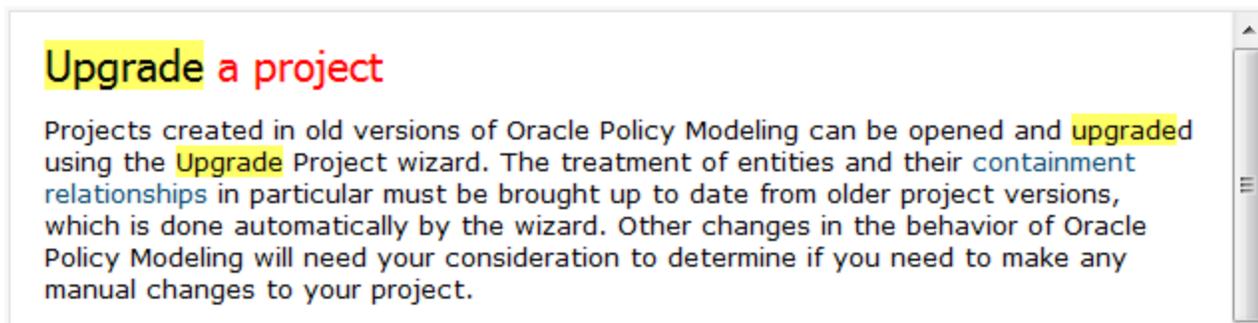


Then click the magnifying glass button, or click **Enter** to search.

Tip: To search for a specific string of text, enclose the text in double quotes in the Search field.

The search results will be displayed as a list of topic titles with summaries of their contents under each.

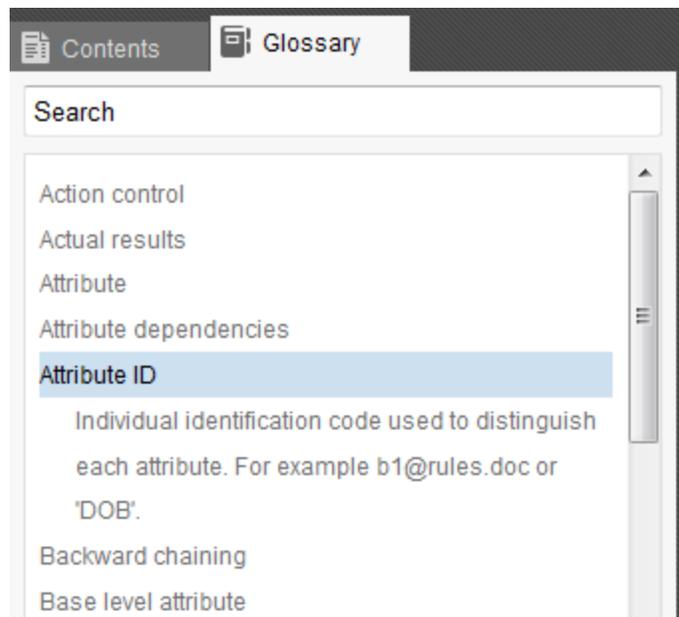
When you click on a topic title in the search results, that topic will open. The term you searched for is highlighted in yellow on the page.



## Find information using the Glossary

The Glossary is a list of key terms used in Oracle Policy Modeling. Click on the **Glossary** tab at the top of the left hand pane to open it.

When you click on a term in the Glossary, the definition will be shown below it.



## Access the Oracle Policy Modeling User's Guide in another language

To access the Oracle Policy Modeling User's Guide in another language, go to **Start Menu | Programs | Oracle Policy Modeling | Oracle Policy Modeling User's Guide** and select the language.

Note that the list of languages for the help will be those that were selected during the installation of Oracle Policy Automation. See the *Oracle Policy Modeling Installation Guide* in your Oracle Policy Modeling installation folder for further details.

## Function references

Function references are provided for all of the languages Oracle Policy Modeling supports. Note that only the US English function reference appears in search results.

See also:

- [Get trained in Oracle Policy Modeling](#)
- [Access further resources](#)

## Create and deploy a rulebase

Oracle Policy Modeling is a set of tools and methodologies which support the creation and deployment of rule-based knowledge models.

Rules can be authored using Microsoft Word or Microsoft Excel. In both Word and Excel you write rules in natural language and format these rules using Oracle Policy Modeling styles. You then compile your rule documents which creates generated rule format files in Oracle Policy Modeling. These files are then used to build rulebase files for use with the Oracle Determinations Engine.

Below are the steps involved in creating a rulebase using Oracle Policy Modeling. Click on any of the links for more information on that step.

1. [Create a new project.](#)
2. [Add a new rule document.](#)
3. [Write rules in Word or in Excel.](#)
4. [Compile the rules.](#)
5. [Debug the rulebase.](#)
6. [Deploy the rulebase.](#)

## Example rulebases

Several example rulebase projects are installed with Oracle Policy Modeling:

- Simple Benefits rulebase
  - A simple rulebase that assesses the claimant's eligibility for teenage child allowance and low income allowance. The rulebase has one entity 'the child', a test script file and no screens.
- Parents And Children rulebase
  - A rulebase that has a simple many-to-many relationship between two entities 'the parent' and 'the child'. It exemplifies the use of entity functions and the collection of entities and relationships on screens. It also has a test script.
- Interview Service Test rulebase
  - A rulebase with one nested rule that determines a person's eligibility for education expenses assistance based on the child's age and school. Screens collect the children, the schools and the children's schools.
- Healthy Eating rulebase
  - A comprehensive rulebase that looks at the diet of the customer's children and rates the overall family's health. There are source and system rules written in Word, as well as rules setting multiple conclusions from the same logic written in Excel. This primary purpose of this rulebase is to demonstrate the use of BI Publisher in Word to create interview documents such as a decision letter and an interview summary document.
- Social Services Screening rulebase
  - A comprehensive rulebase that investigates the household member's eligibility for a range of social services. There are rules written in Word and Excel, and in addition to the source rules there are many system rules (interpretative, procedural, validation and visibility). The rulebase also contains rule visualizations and test scripts. Social Services Screening is a very good example of a customized version of Oracle Web Determinations, and of a claim form document generated from the answers provided during an interview.
- Inferred Entity Instances rulebases
  - Inferred Brand Discount rulebase
    - A rulebase that uses inferred entity instances to group order items by brand and then apply a brand discount for purchases over \$100 for any given brand.
  - Inferred Benefits rulebase
    - A rulebase that infers the existence of benefits and tallies the number of people eligible for each benefit. It also demonstrates inferred instances using rule tables.

- Inferred Tax Years rulebase
  - A rulebase that infers the existence of tax year entity instances so that further rules related to those tax years could be applied.
- Inferred Service Delta rulebase
  - A rulebase that infers the existence of service entity instances in order to identify which services should be started, stopped or retained when a customer changes phone plans. It also demonstrates inferred instances from global values.
- Insurance Fraud Score rulebase
  - A rulebase that calculates the fraud score for an insurance claim. The rulebase has one entity 'the previous claim'. The fraud score is calculated based on the current claim and an average of fraud scores accumulated for previous claims. It demonstrates the following features in an Excel rulebase: creating rule tables with merged condition and conclusion cells, using 'Apply Sheet' to reason about attributes that change over time (fraud score points for cover and value), using entity level attributes, functions and calculations based on entity instances. The rulebase also contains rule visualizations and test scripts.
- Income Support Benefit
  - A rulebase containing a module file, using a fictitious example of rulebase which assesses eligibility and rate of unemployment benefit. The Rates and Thresholds module is separated from the main Income Support Benefit rulebase so the rates can be updated independently of the main rulebase and to allow those rates to be re-used in other rulebases.
- Aged Care Approval
  - A rulebase that investigates the validity of an Aged Care Approval. It has one Word rule document and one imported test case. The rules demonstrate several temporal functions operating together.

### Open an example rulebase

1. Go to \Program Files\Oracle\Policy Modeling\examples.
2. Select the folder for the rulebase you would like to view.
3. Copy the folder and paste it into C:\projects.
4. Open the folder and unzip the zip file for the project into that folder.
5. Open Oracle Policy Modeling and select **File | Open Project...**
6. Browse to C:\projects\\Development and select the <project name>.xprj file. Click **Open**.

### Get trained in Oracle Policy Modeling

Further training in Oracle Policy Modeling is available from Oracle University:

[Oracle Policy Modeling Courses](#)

Access further resources on Oracle Policy Automation

### Oracle Policy Automation Developer's Guide

Technical information on Oracle Policy Automation is in the Oracle Policy Automation Developer's Guide which is available from:

- [Start Menu | Programs | Oracle Policy Modeling | Oracle Policy Modeling Tools | Oracle Policy Automation Developer's Guide](#)

## **Oracle Policy Automation Discussion Forum**

To search for details of any questions you may have, or to ask questions directly if they have not already been discussed on the forum, go to:

- [Oracle Policy Automation Discussion Forum](#)

## **Oracle Policy Automation Knowledge Base**

The Oracle Policy Automation Knowledge Base contains various articles on Oracle Policy Automation, including technical 'how to' instructions, known issues and their workarounds, and product announcements. (NOTE: You will need Oracle customer details to view it.) It can be accessed from:

- [support.oracle.com](https://support.oracle.com)  
In the Browse Knowledge area, type "Oracle Policy Modeling" or "Oracle Policy Automation" into the Find a Product by Name field.

## Introducing Policy Modeling

Oracle Policy Modeling is an integrated development environment for developing rules and rule-based applications. It is also used to compile rulebases and screens for use by the Oracle Determinations Engine and Web Determinations.

Oracle Policy Modeling projects are comprised of files and settings contained in a project file. To get started in Oracle Policy Modeling, see [Create and deploy a rulebase](#).

To understand more about the way that Oracle Policy Modeling rulebases work, see [Oracle Determinations Engine and the Inference Cycle](#).

There are several sample rulebases installed with Oracle Policy Modeling. For more information, see [Example rulebases](#).

### Oracle Determinations Engine and the Inference Cycle

The Oracle Determinations Engine is an **inferencing engine** which works with Oracle Policy Modeling rules to conduct queries and make decisions based on those rules. In short, it is the 'brain' that does the thinking based on the rules you have defined. For example, if you set the value of "the person is a pensioner" to true, the Determinations Engine may infer that "the person is eligible for a discount at the university bookstore" is also true.

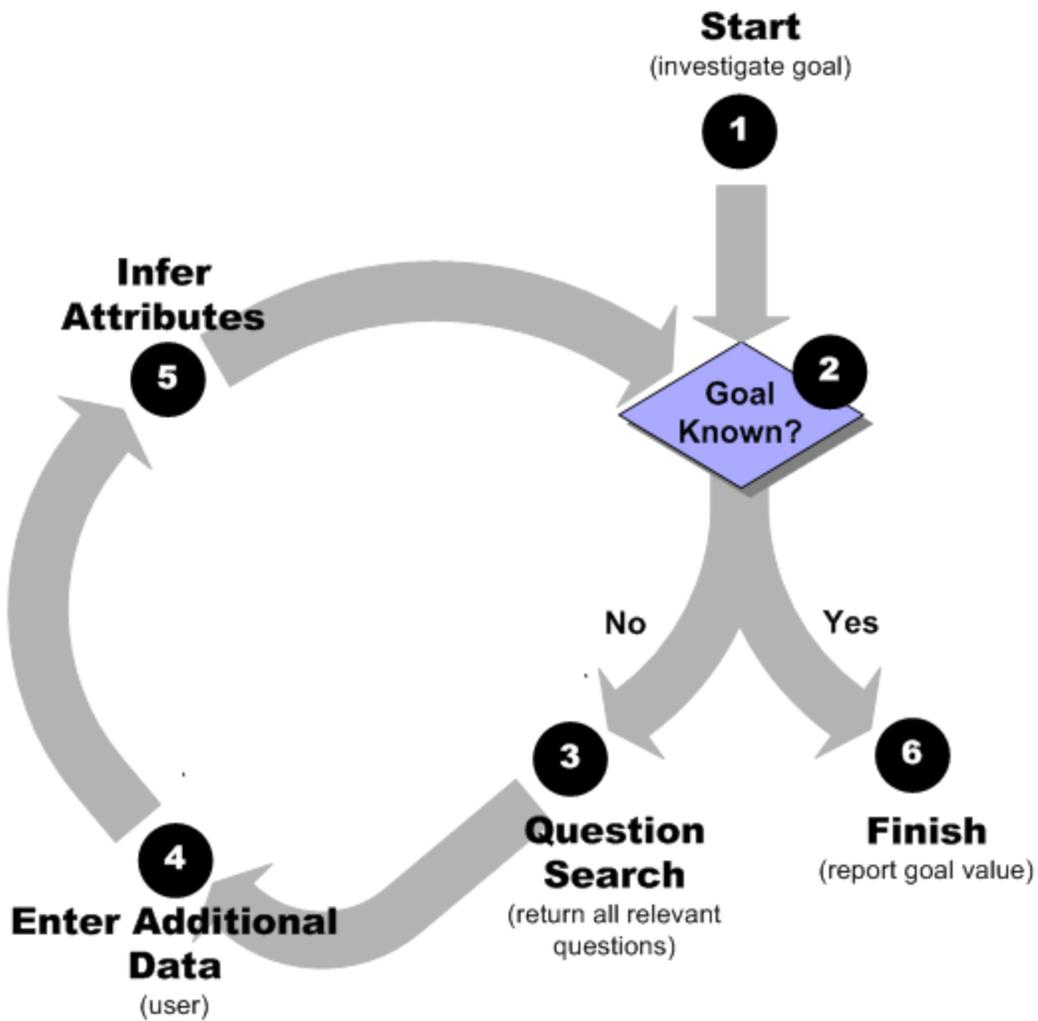
Each time you conduct a new assessment against a rulebase using the Determinations Engine (for example, creating a new interview in Web Determinations) you are creating what is called a **rulebase session**. The Determinations Engine does not maintain the state of its rulebase sessions, so each time you end a rulebase session your data will be forgotten by the Determinations Engine unless you explicitly save it.

The process of querying and inferencing during a rulebase session is known as the **Inference Cycle**.

### The Inference Cycle

The Inference Cycle is the cycle of question and answer which operates on rules to replicate the decision making process.

The following diagram illustrates the Inference Cycle:



The diagram above shows the following steps:

1. **Start (investigate goal):** An attribute is specified as the goal attribute to be investigated.
2. **Goal Known?:** The Determinations Engine determines whether or not the goal attribute has a value.
3. **Question Search:** The Determinations Engine finds all known (or unknown) attributes that influence the goal based on the rules in the rulebase (an inferencing process known as backward chaining), then reports any influencing attributes that are unknown. Another way of thinking about this is that the Determinations Engine is asking, "What do I need to find out to prove this attribute?".
4. **Enter Additional Data:** The Determinations Engine waits for a value(s) to be input for the attribute(s) raised by the Question Search.
5. **Infer Attributes:** The rule decision tree is scanned by the Determinations Engine in the reverse direction, drawing conclusions based on attributes that are now known. This inferencing process is known as forward chaining. Another way of thinking about this is that the Determinations Engine is asking, "What can I conclude based on the collection of what I know?".

6. **Finish:** Once the goal attribute is known, the Determinations Engine reports the value and how it reached that decision (if requested). The Determinations Engine generates a Decision Report (if required) using backward chaining, as described above.

The Inference Cycle repeats steps 2 to 5 until the goal attribute is known.

See also:

- [What is a rule?](#)
- [Interviews and flows](#)
- [Decision reports](#)
- [Deploy an interview to Web Determinations](#)

## Projects and files

Topics in "Projects and files"

- [Create, modify or delete a project](#)
- [Upgrade a project](#)
- [Update Oracle Policy Modeling Templates](#)
- [Manage legislation and other source material](#)
- [Organize project files](#)
- [Add, rename or remove a rule document](#)
- [Save changes to a project](#)
- [Get project statistics](#)
- [Edit a rule document](#)

See also:

- [Share rule documents across projects](#)

### Create, modify or delete a project

An Oracle Policy Modeling project is created to manage the rule documents and other files that make up a rulebase.

### What do you want to do?

[Create a new project](#)

[Open an existing project](#)

[Add existing files to a project](#)

[Add new files to a project](#)

[Delete a project](#)

Create a new project

To create a new rulebase project:

1. In Oracle Policy Modeling, select **File | New Project**.

**New Project**

Project Name:

Rule Language:  
 English (American) ▼

Region:  
 United States ▼  
 This will determine the numeric, date and currency formatting used in rule documents, and in Web Determinations.

Select a folder that will contain the project files.

Project Folder:

Create default folder structure

2. In the New Project dialog box, enter a name for the project in the **Project Name** field.  
 NOTE: Do not use a trailing "." (dot) in the Project Name, as this can cause a deployment error in IIS.
3. In the **Rule Language** drop-down list, select the language in which you will write the rules. The rule language determines what language documents are parsed in.  
 If the language that you want to create your project in is not in this list, you can use the Oracle Policy Modeling Rapid Language Support Tool to create a language parser for a different language. (The Oracle Policy Modeling Language Support Tool is available from Start | All Programs | Oracle Policy Modeling | Oracle Policy Modeling Tools | Oracle Policy Modeling Rapid Language Support Tool, and help on using that tool is available from the Help menu in the tool itself.)  
 Once you have created a new language parser using this tool, when you reopen Oracle Policy Modeling and select File | New Project, the parser you created will appear in the Rule Language drop-down list.  
 Note that you may also write a rulebase in one language, and then [add one or more translations to the rulebase](#) to allow it to be run in other languages.
4. In the **Region** drop-down list, select the appropriate region for the rulebase. This setting determines the formatting of numbers, dates and currency values. This applies by default to both the deployment of the rulebase in Oracle Web Determinations or other application (eg how values entered into your rulebase by a user are interpreted), and some aspects of the rule documents in your rulebase (eg how some constant values referenced in your rules, such as income limits, are interpreted). Note that you may customize the deployment settings so they are not based on this project setting - please see the [Oracle Policy Automation Developer's Guide](#) for details.
5. In the **Project Folder** field, specify the location for the project. The recommended location for Oracle Policy Modeling projects is c:\Projects or d:\Projects (whatever the main local drive is).  
 TIP: You can create a new folder by simply typing the directory followed by the new folder name into this field.

6. If you want to create the default folder structure, select the option to **Create default folder structure**.  
TIP: It is recommended that the default folder structure is created as it will help organize your project. For more information, see [Organize project files](#).
7. Click **Create** to create your project.  
Your new project will open in Oracle Policy Modeling. If you look in the project folder on your computer (eg in Windows Explorer) you will notice that a new folder **Development** has been created. This folder contains your project file (.xprj) and the default folders (if the options to create these was selected). The xprj file is the master project file which records the file and folder structure of the project.

## Open an existing project

To open an existing project:

1. In Oracle Policy Modeling, select **File | Open Project**.
2. In the **Open Project** dialog box, browse to your existing Oracle Policy Modeling project file (.xprj). Then click **Open**.

Alternatively, you may double-click on an existing project file in Windows Explorer to launch the project in Oracle Policy Modeling.

NOTE: If the project was created in an older version of Oracle Policy Modeling, it will need to be upgraded before it can be opened. See [Upgrade a project](#) for more information.

## Add existing files to a project

To add an existing file to a project:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the file to be placed in, then select **File | Add | Add Existing File...**
2. In the **Add Existing File** dialog box, browse to the file that you want to add. Then click **Open**. NOTE: If the file was created in an older version of Oracle Policy Modeling, you will be prompted to upgrade the file at this point.

The file will now appear in the Project Explorer in Oracle Policy Modeling and can be opened by double-clicking it.

## Add new files to a project

To add a new file to a project:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the file to be placed in, then right-click and select the type of file that you would like to add.  
The options are **Add New Word Document**, **Add New Excel Document**, **Add New Translation Document**, **Add New Screens File**, **Add New Properties File**, **Add New Visual Browser File**, or **Add New Test Script File**.
2. Type a name for the new document, then press **Enter**.

The file will now appear in the Project Explorer in Oracle Policy Modeling and can be opened by double-clicking it.

## Delete a project

To delete a project from your file system:

1. In Windows Explorer, browse to select the project folder that contains the project that you want to remove.
2. Right-click the folder and select **Delete**.
3. In the **Confirm Folder Delete** dialog box, click **Yes**.

## Upgrade a project

Projects created in old versions of Oracle Policy Modeling can be opened and upgraded using the Upgrade Project wizard. The treatment of entities and their [containment relationships](#) in particular must be brought up to date from older project versions, which is done automatically by the wizard. Other changes in the behavior of Oracle Policy Modeling will need your consideration to determine if you need to make any manual changes to your project.

### What do you want to do?

[Upgrade a project using the Upgrade Project wizard](#)

[Understand changes in the behavior of Oracle Policy Modeling](#)

### Upgrade a project using the Upgrade Project wizard

To upgrade a project created in an older version of Oracle Policy Modeling:

1. Open the project in Oracle Policy Modeling (**File | Open Project**), select the project file (.xprj) and click **Open**.
2. The Upgrade Project window is shown, showing the older version of Oracle Policy Modeling from which the project will be upgraded. Note that the project files will be copied to a backup location before the upgrade is performed, to ensure that you have the original version of the project to refer to if necessary. Release folders are not included in the upgrade process. Click **Continue**.
3. The project upgrade is performed, converting entities and relationships to the current version of Oracle Policy Modeling as required. Any test cases in the project are also upgraded. Any messages or warnings that are relevant to the upgrade are displayed in the **Error List** after the upgrade is performed.

The wizard will also upgrade an older properties file added to a new project in this way.

### Principles for the upgrading of entities and their containment relationships

The Upgrade Project wizard applies the following principles in upgrading entities and their containment relationships:

- "One-to-many" relationships between the global entity and other entities are upgraded to containment relationships where possible.
- Where the relationship structure provides no clear definition of containment relationships, the presence of entity collect screens for the relevant entities guides which relationships are defined as containment relationships.
- Where the relationship structure provides no clear definition of containment relationships, and no relevant entity collect screens are defined, new containment relationships are created from the global entity to the relevant entities, and the old relationships preserved as reference relationships.
- Singleton entities (deprecated) or one-to-one entities will not be treated as containing entities in the upgrade process.
- Relationships from projects created in Oracle Policy Modeling version 10.0 or earlier will be upgraded as reference relationships, and new containment relationships created from the global entity to other entities as appropriate.

### Understand changes in the behavior of Oracle Policy Modeling

#### Radio buttons for booleans

The Radio Buttons option for boolean inputs on screen controls has been removed. This means that if an existing project uses the Radio Buttons option with default values, you will need to delete the control and recreate it using the Default option (which creates radio buttons for booleans).

## Output folder

The 'output' folder is now a strict output folder and can no longer be used to include additional files in the compiled rulebase zip file. If you have any other files stored in the output folder of your project, you will need to move these to the 'include' folder to have them included in the compiled rulebase zip file.

## Time/date difference functions

The various time/date difference functions (MinuteDifference, HourDifference, DayDifference, WeekDifference, etc) no longer return 0 in the case where the first time/date parameter is after the second time/date parameter. This means that the order of the two parameters is no longer significant, for example, "the number of days between X and Y" will produce the same result as "the number of days between Y and X".

If your rules using these time/date difference functions are relying on a 0 result, or you want to ensure that you get exactly the same behavior as previously, you will need to build some extra logic into your rules to set the conclusion to 0 if the second date is before the first date.

## Missing values in Excel

Any condition row proving a conclusion in a merged cell can now evaluate in any order. This means that a rulebase outcome in this release may be known earlier than in previous versions. To have your rules evaluate in the top-down order of previous versions, unmerge your conclusion cells.

## Functions in Excel

If you want to use a text function in an Excel rule table you now need to put the function text in parentheses. Existing projects that use text functions will need to have parentheses added otherwise the function will be treated as a text value.

## Text values in Excel

Changes made to how Excel processes cell contents have affected the way quoted text is interpreted. This means that double quotes, instead of single quotes, should now be used.

## Unknown relationship reasoning

There are two significant consequences of the changes to how unknown relationship reasoning now operates.

The first is that backward chaining knows more about what information might possibly be required in chasing down a goal.

For example, say your rulebase has household members, and each household member refers to some global property such as the number of bedrooms in the residence. Previously you actually needed to create a household member before the engine knew that the bedroom-count could be required. Now it can actually reason about a 'hypothetical' household member and from there work out the bedroom-count is a question that may need be to be asked.

The second consequence is that the engine can also sometimes draw conclusions in cases where it previously did not think it could.

Say, for example, that you have the following rule:

**the parent does not require disability carer's assistance if**

Exists(the parent's children, the child has a disability)

And you have a bunch of a children and a bunch of parents, but you haven't yet said who is the parent of who (ie both parent and child are global-level entities). If none of the children have a disability, the engine will now infer that none of the parents require disability carer's assistance. It knows this because even without knowing who a person's children are, it knows that none of the hypothetical candidates could fulfill the criteria, therefore the conclusion is definitively false.

## Warning shown when the Oracle Web Determinations template version does not match the current version of Oracle Policy Modeling

When you Build and Debug with Screens, or Build and Run with Web Determinations, if the 'Replace deployed version' option is turned off, and the Web Determinations template version is not the same as the current version of Oracle Policy Modeling, the following warning will be displayed:

"The currently deployed version of Web Determinations is not the version expected by Oracle Policy Modeling. This might cause problems during runtime. Do you want to continue?"

To prevent this warning from being shown, select the option to replace your currently deployed version of Web Determinations in the Debug Options or Build and Run dialog. (This is not done automatically in case the user has customized Web Determinations.)

## Document controls

A document will be created based on the document control information in a project created in a version of Oracle Policy Modeling prior to 10.3.0. The resulting document will not have an RTF template associated with it (since previous versions used XSLT) so this will generate a build warning that will need to be manually addressed in the project. Also, any previously specified decision reports that do not have public names will also cause build warnings and will need to be updated.

## Unformatted text in translation documents

When an existing translation document is opened, a new column "Unformatted Text" will have been added to the Statements (3rd Person) and Variables (3rd Person) worksheets. It will contain non-translated fields which will need to be translated with the basic form of the attribute. See [Update a translation file](#) for how to do this.

See also:

- [Understand containment relationships and entity completion](#)

## Update Oracle Policy Modeling Templates

Project files created in previous versions of Oracle Policy Modeling are typically [upgraded](#) when the project is loaded in the new version of the application, or when added as existing files to a project. Occasionally you may need to update the template of an Oracle Policy Modeling document manually. To do this you use the **Template Update Wizard**.

1. Go to **Tools | Update Oracle Policy Modeling Templates...**
2. Specify the folder containing the documents you wish to update. By default this will be the **Development** folder in your project folder.
3. Select the **Include sub-folders** checkbox if you want the wizard to look in all sub-folders for documents to update.
4. Select the **Update document styles from template** if you want to update the Oracle Policy Modeling document styles (if these have changed from the previous version).
5. Select the **Remove embedded statements and variables** option if you want to strip the metadata from the documents (ie if documents were last compiled against entities and relationships that no longer exist or have been relocated).
6. Click **Next**. The Wizard will then scan the specified folder/s and list the documents that use the Oracle Policy Modeling template. Use the checkboxes next to the documents to select which documents you want to update the templates of. (By default all documents will be selected.)
7. Click **Next**. The results of the template update will be shown on the next screen.
8. Click **Finish** to close the Template Update Wizard.

## Manage legislation and other source material

Legislation and other source material can be contained within the Oracle Policy Modeling project to make it easy to access and refer to these documents while working on a project. These documents should be kept in their original unchanged format, and the in-scope content from them copied and pasted into separate rule document files for processing into Oracle Policy Modeling rules.

### Add a source document to a project

Source documents should be contained in a separate folder in the project, ideally in the **Documents/Source** folder. To add a document to this folder:

1. In the Project Explorer in Oracle Policy Modeling, select the Documents/Source folder, then right-click and select **Add Existing File...**
2. In the **Add Existing File** dialog box, browse to the file that you want to add. Then click **Open**.  
NOTE: You can only add Word, Excel or PDF files to your project.

The file will now appear in the Project Explorer in Oracle Policy Modeling and can be opened in its own application by double-clicking it.

### Exclude a source file from the build

Source documents should be excluded from the build. To do this:

1. In the Project Explorer in Oracle Policy Modeling, select the source file.
2. Right-click and select **Properties...**
3. In the **Properties** dialog box, clear the **Include document in build** checkbox. Then click **OK**.

The document icon will now be shown with a red line in the bottom right hand corner in the Project Explorer to indicate that the document is not included in the build.

## Organize project files

Folders are used in Oracle Policy Modeling to organize project files. When you create a new project you have the option to create a default folder structure which is the standard way of organizing your project files.

### What do you want to do?

[Decide whether or not to use the default project folder structure](#)

[Create a new project folder](#)

[Add an existing folder](#)

[Rename a project folder](#)

[Remove a project folder](#)

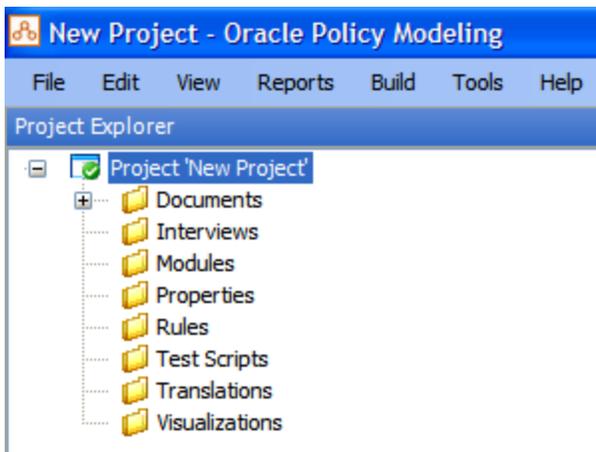
[Move files between folders](#)

[Sort folders and files](#)

[Locate a rulebase file in Windows Explorer](#)

### Decide whether or not to use the default project folder structure

The default folder structure, created when you [set up a new Oracle Policy Modeling project](#), is:



This folder structure is also physically created in the same location as your Oracle Policy Modeling project. The folders are for assistance only. Documents may be contained in any folder.

If this folder structure is not suitable for your individual project, unselect the option to **Create default folder structure** in the **New Project** dialog. You will then need to manually create project folders (see below).

### Create a new project folder

To create a folder in your Oracle Policy Modeling project:

1. Select the folder in the Project Explorer where you would like to create the folder. (If you want the folder created at the top level in your project, select the project name.)
2. Right-click and select **Add New Folder**.
3. Type a name for the folder, then press **Enter**.

### Add an existing folder

To add an existing folder to a project:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the existing folder to be placed in.
2. Right-click and select **Add Existing Folder...**
3. In the **Add Existing Folder** dialog box, select the folder/s that you want to add. NOTE: This dialog box will only display the folders that already exist in the project folder.
4. Use the check box to indicate whether you want to include all files and sub folders, then click **OK**.

NOTE: When adding sub-folders, hidden files and directories will be ignored. Hidden files/folders can still be added manually using the respective **Add Existing [File/Folder]** options.

### Rename a project folder

To rename a folder in your project:

1. In the Project Explorer in Oracle Policy Modeling, right-click the folder that you want to rename and select **Rename**.
2. Type a new name for the file, then press **Enter**.

### Remove a project folder

To remove a folder from a project:

1. In the Project Explorer in Oracle Policy Modeling, right-click the folder that you want to remove and select **Remove from Project**.

### Move files between folders

To move a file to a different folder:

1. In the Project Explorer in Oracle Policy Modeling, select the file that you want to move.
2. Drag the file to the folder where you want to move it to (the folder will become highlighted) and release your mouse button.
3. You may be advised that moving the file may cause existing attribute links to break because the document is currently using an automatically generated Scope ID. Click **Yes** to persist the current Scope ID so as to avoid these broken links.

### Sort folders and files

By default, folders, and files in folders will be sorted alphabetically. To turn this feature off (so that folders and files appear in the order that you added them):

1. Go to **File | Project Properties | Common Properties | General**.
2. Unselect the **Sort project explorer** checkbox.
3. Click **OK**.

### Locate a rulebase file in Windows Explorer

You can locate any of your rulebase files in Windows Explorer from within Oracle Policy Modeling.

1. In the Project Explorer, right-click on the file you wish to open in Windows Explorer.
2. Select the **Locate in Explorer** option in the menu. A new Windows Explorer window will be opened showing the folder containing the rulebase file.

### Add, rename or remove a rule document

Oracle Policy Modeling rules are written in Microsoft Word or Microsoft Excel. After a rule document has been added to a project, it can later be renamed and/or removed.

## What do you want to do?

[Add a new rule document](#)

[Rename a rule document](#)

[Remove a rule document](#)

### Add a new rule document

To add a new rule document to a project:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the file to be placed in.
2. Right-click and select either **Add New Word Document** or **Add New Excel Document**.
3. Type a name for the new rule document, then press **Enter**.

The file will now appear in the Project Explorer in Oracle Policy Modeling and can be opened by double-clicking it.

### Rename a rule document

To rename a rule document:

1. In the Project Explorer in Oracle Policy Modeling, right-click the file that you want to rename and select **Rename**.
2. Type a new name for the file, then press **Enter**.

## Remove a rule document

To remove a rule document from a project:

1. In the Project Explorer in Oracle Policy Modeling, right-click the file that you want to remove and select **Remove from Project**.

NOTE: The file remains in your file system but has been removed from your Oracle Policy Modeling project. To permanently delete a file from both your file system and from your project, right-click it in Oracle Policy Modeling and select **Delete**.

See also

- [Add existing files to a project](#)

## Save changes to a project

If there are changes in your project that need to be saved, an asterisk will be displayed next to the project name in the Project Explorer in Oracle Policy Modeling. You can save changes to individual files, or save all changes to the project.

NOTE: Changes to Microsoft Word and Excel documents need to be saved from within these applications. This happens automatically when you compile.

## Save changes to an individual file

To save changes to an individual file:

1. In the Project Explorer, select the file.
2. Select **File | Save <file name>**.

## Save all changes to the project

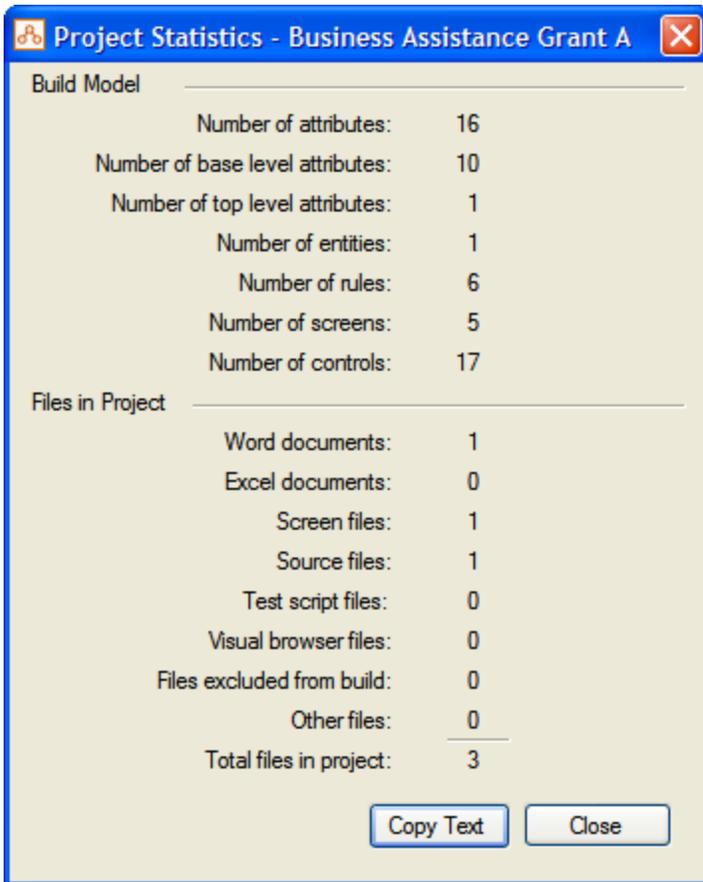
To save all changes to the project:

1. Select **File | Save All**, OR
2. Press Ctrl+Shift+S.

## Get project statistics

The **Project Statistics** list is a summary of the current status of the project, including the number of files in the project, and the number of attributes, entities, rules and screens.

To view the project statistics, select **File | Project Statistics**.



To copy the text of this dialog box, use the **Copy Text** button.

## Edit a rule document

To edit a rule document you need to open the document in Word or Excel.

In the Project Explorer in Oracle Policy Modeling, either:

1. Double-click the rule file, OR
2. Right-click the rule file and select **Open with Microsoft Word** or **Open with Microsoft Excel**.

The file will then open in its own application.

Make the necessary changes to the document and then [compile](#) it.

## Writing rules

### Topics in "Writing rules"

- What is a rule?
- Decide whether to write rules in Word or Excel
- Write rules in Word
- Define rule tables in Word documents
- Define decision tables in Excel workbooks
- Make your Excel rules easier to understand
- Capture implicit logic in rules
- Write rules in the negative
- Prove an attribute using multiple rules
- Model loops in rule logic
- Include an existing attribute in a rule
- Choose a function to include in a rule
- Add rule metadata
- Validate user input using errors and warnings
- Use rules to trigger external software applications

### See also:

- Create and deploy a rulebase
- Split and link rules
- Choose a name for an entity, relationship or attribute
- Use an attribute in a rule
- Use an entity or relationship in a rule

## What is a rule?

### **What do you want to learn about?**

What is a rule?

What is a rulebase?

Conclusions and conditions

What is an attribute?

Attribute levels

Connecting conditions using and/or

Grouping conditions using both/all and either/any

Alternative conclusions

## Rule types

### What is a rule?

A **rule** is an assertion that a conclusion can be drawn from a particular state of affairs. For example:

If you leave the ice cream in the sun, then the ice cream will melt.

It is a good idea to take an umbrella if it is raining outside.

Full-time students and pensioners are eligible for a discount at the university bookstore.

Your plane can take-off from the airport if it has permission from the control tower and has completed a safety check.

The movie ticket will cost \$10 if the ticket is for a child.

The claimant is not eligible for an aged pension if the claimant is not a citizen

Rules operate on data and can incorporate operations such as [comparisons](#) and [mathematical functions](#).

### What is a rulebase?

A **rulebase** is simply a collection of one or more connected rules. For example:

Rule 1:

**the person is eligible for a discount at the university bookstore if**

the person is a full-time student or

the person is a pensioner

Rule 2:

**the person is a full-time student if**

the person is studying a full-time load and

the person does not have a full-time job

## Conclusions and conditions

Each rule must have a **conclusion** (the state of affairs that can be determined) and usually has at least one **condition** (the conditions upon which that determination may be made). A conclusion is the "Then" part of an "If... Then..." statement. A condition is the "If" part of an "If... Then..." statement.

CONCLUSION: the ice-cream will melt if

CONDITION: the ice-cream has been left in the sun

CONCLUSION: it is a good idea to take an umbrella if

CONDITION: it is raining outside

CONCLUSION: the person is eligible for a discount at the university bookstore if

CONDITION: the person is a full-time student

CONDITION: the person is a pensioner

CONCLUSION: your plane can take-off from the airport if

CONDITION: it has permission from the control tower

CONDITION: it has completed a safety check

CONCLUSION: the cost of the movie ticket = \$10 if

CONDITION: the ticket is for a child

CONCLUSION: the claimant is not eligible for an aged pension if

CONDITION: the claimant is not a citizen

NOTE: The value of the condition may be different to the value of the attribute as used in the condition.

The table below demonstrates the range of values which a condition may have:

Condition	Actual Citizenship	Value
The claimant is an Australian citizen	Australian	True
The claimant is an Australian citizen	American	False
The claimant is not an Australian citizen	Australian	False
The claimant is not an Australian citizen	American	True

What is an attribute?

An attribute is a single unit of data or fact. For example:

- the person is a full-time student
- the cost of the movie ticket

An attribute is of a particular data type: boolean, text, number, currency, date, time of day, or date and time. Boolean attributes can either have a true or false value, and variable attributes take a text, number, currency, date, time of day, or date and time value depending on the type of variable.

The following are some examples of attributes and types:

- the person is hungry (boolean attribute)
- the person's name (variable attribute – text)
- the person's date of birth (variable attribute – date)
- the number of cookies the person wants to eat (variable attribute – number)
- the cost of the person's meal (variable attribute – currency)

An attribute always belongs to a particular entity even if it is the global entity. Attributes form the building blocks of rules.

## Attribute levels

Attributes will have different purposes depending on their place in the rule hierarchy. For example, consider the hierarchy of attributes in the following rules:

Rule 1

**the person is eligible for a discount at the university bookstore if**

the person is a full-time student or

the person is a pensioner

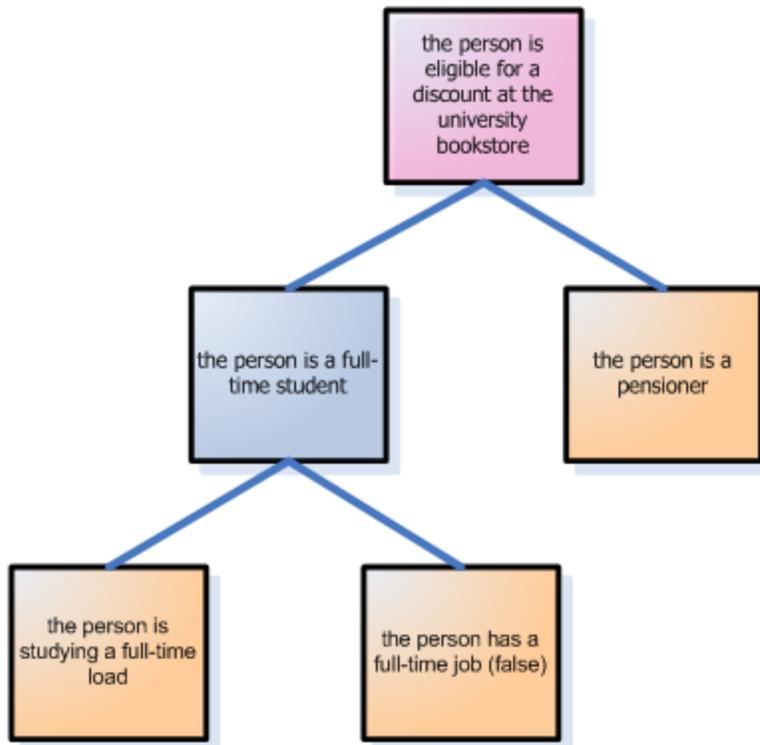
Rule 2

**the person is a full-time student if**

the person is studying a full-time load and

the person does not have a full-time job

Diagrammatic form:



In this rulebase, the attribute "the person is eligible for a discount at the university bookstore" is the **top-level attribute**. That is, the attribute is at the top of the rule hierarchy, it is not used as a condition in any other rule. Top-level attributes usually represent the main outcome or **primary goal** of the rulebase (that is, the question the rulebase seeks to answer).

The attribute in the middle, "the person is a full-time student" is called an **intermediate attribute** as it is used as a condition in at least one rule and a conclusion in another. Intermediate attributes can also be called goals where they calculate an outcome which may be of interest to a user.

The attributes:

- the person is a pensioner
- the person is studying a full-time load and
- the person has a full-time job

are all **base-level attributes** in the rule hierarchy. That is, there are no rules explaining how these attributes are to be determined. The value of base-level attributes must be provided by the user.

### Connecting conditions using and/or

Where a rule contains multiple conditions, the conditions must be separated by an **and** or an **or** to indicate whether one or all conditions are required to satisfy the conclusion.

For instance,

Example 1	Example 2
the person is eligible for a pension if:	the person is eligible for a pension if:
the person is over 65.	the person is over 65.
AND	OR
the person is a citizen.	the person is unable to work.

In Example 1, both conditions must be true to be able to draw a positive outcome for the person's eligibility. If either condition is false, then only a negative outcome can be drawn.

In Example 2, either the first or second condition, or both, must be true to be able to draw a positive outcome. If both the conditions are proved false, then a negative outcome is drawn.

For more information on the possible outcomes when using **and** or **or**, see [Truth tables](#).

There is no restriction on the number of **ands** and **ors** that can be used in a rule. For instance,

Example 1	Example 2
the person is eligible for a pension if	the person is eligible for a pension if
the person is over 65	the person is over 65
AND	OR
the person is a citizen	the person is a citizen
AND	OR
the person is unable to work	the person is unable to work

Both **ands** and **ors** can be used within the same rule in order to closely model source material. It is not possible, however, to mix these two operators on a single level without creating an ambiguity in the logic.

### Explain this further

For example:

A if B or C and D

could be interpreted as:

A if B or (C and D)

in which case B is sufficient to prove A. Or it could be interpreted as:

A if (B or C) and D

in which case, D would always be required.

The rule author must distinguish between the two interpretations when writing the rules.

### Grouping conditions using both/all and either/any

The **all** operator is used to group conditions separated by **and**. In the example "A if B or (C and D)" the brackets are around the conditions joined by an **and** so you must use the **all** operator in your rule:

#### A is true if

B is true

or

all

C is true

and

D is true

The **any** operator is used to group conditions separated by **or**. In the example "A if (B or C) and D" the brackets are around the conditions joined by an **or** so you must use the **any** operator in your rule:

#### A is true if

any

B is true

or

C is true

and

D is true

NOTE: You may also use the word **both** in place of **all** and **either** in place of **any**. Using these words has the same effect but may make the text more readable where only two conditions are grouped.

The grouping operators sit above the conditions they are grouping. The conditions being grouped sit beneath the grouping operator and should therefore take the style of the next level down. For example, if the word "any" is in **Level 1** style, the conditions it is grouping should be in **Level 2** style.

The following example demonstrates this placement:

```
the claimant is eligible for a pension if
  the claimant is poor
  or
  all
    the claimant is sick and
    the claimant has been sick for more than 6 months and
    the claimant does not another form of income
```

Where your rule continues (as in the example below) at the higher level, the appropriate operator (**and** or **or**) should be added as a separate line at the same level as the subsequent condition. For example:

```
the claimant is eligible for a pension if
  the claimant is poor or
  all
    the claimant is sick and
    the claimant has been sick for more than 6 months and
    the claimant does not another form of income
  or
  the claimant has been entitled to a pension previously
```

### Alternative conclusions

By default, Oracle Policy Modeling assumes all rules contain an **alternative conclusion**. That is, if the conditions are not satisfied, you can infer the opposite of the conclusion. For example, given the rule:

```
CONCLUSION: it is a good idea to take an umbrella if
CONDITION: it is raining outside
```

If it is not raining outside, you may conclude that it is not a good idea to take an umbrella.

The alternative conclusion need not be stated, it is assumed in all rules unless otherwise indicated.

### Rule types

Oracle Policy Modeling supports the following rule types:

- **Global rules** - use global attributes
- **Entity-level rules** - use entity-level attributes and operate on sets of data simultaneously.
- **Shortcut rules** – allow the value of one base attribute to be inferred from the value of another base attribute. These are the only rules which do not require an alternative conclusion.

- **Warning and error event rules** – fire a warning or error in the Oracle Determinations Engine. These are commonly used to control screen inputs (such as warning the user they have entered conflicting data).
- **Custom event rules** – allow the rulebase to call custom code where the functions in the rulebase are simply not sufficient or data is stored outside of the rulebase (for example, to call an external database of dates rather than capturing the dates in rules).

## Decide whether to write rules in Word or Excel

Microsoft Excel should be used to capture the rule if:

- the source material is a decision table or
- the rule logic is appropriate to convert into a decision table (see below)

In addition only rules of the following type should be written in Excel:

- where multiple conclusions can be set from the same logic (Example A)
- where multiple conclusions can be set from different values of one attribute (Example B)

Otherwise, all rules should be written in Microsoft Word.

## Is the rule logic appropriate to convert to a decision table?

The rule logic is appropriate to convert into a decision table if the rule logic is not more than one level deep. If the rule logic is more than one level deep it can still be converted to a decision table providing:

- intermediate logic is not required in the decision report\* and
- the rule is relatively simple to translate into a decision table while having confidence that all combinations of attribute values are captured in the decision table.

\*Excel decision reports just show the values and the outcome, without detailed reasoning.

## Example A (multiple conclusions set from the same logic)

**the person must be sent an approval letter if**

the person is eligible

**the person must be sent an information pack if**

the person is eligible

## Example B (multiple conclusions set from different values of one attribute)

**the pet is a dog if**

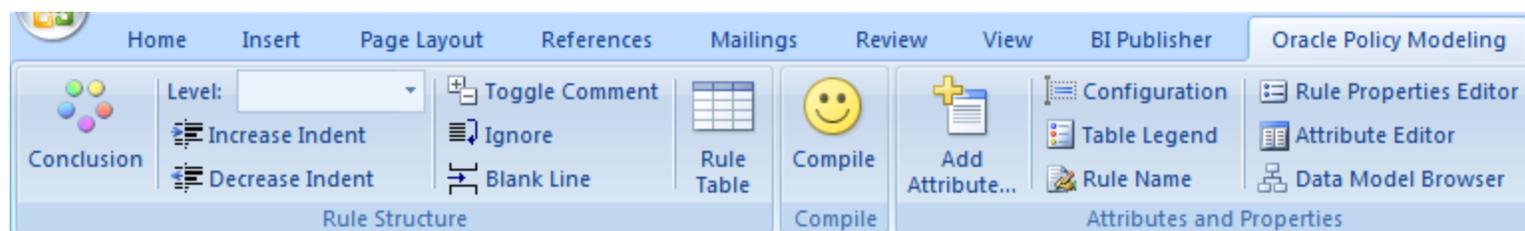
the pet's species = "dog"

**the pet is a cat if**

the pet's species = "cat"

## Write rules in Word

Using Microsoft Word you can write your rules in plain English. You then format these rules with the styles provided on the Oracle Policy Modeling tab to enable them to be compiled into a format that can be used by the Oracle Determinations Engine.



Before you start writing rules, you need to change some of the default settings in Word.

### What do you want to do?

[Prepare Word for writing rules](#)

[Understand Oracle Policy Modeling format and structure](#)

[Write a rule in Word](#)

Prepare Word for writing rules

Some normal settings in Microsoft Word will interfere with rule creation by Oracle Policy Modeling, so you will need to make the following changes to Word settings:

#### AutoCorrect

In Tools | AutoCorrect Options | AutoCorrect tab (in Word 2003), or Word Options | Proofing | AutoCorrect Options | AutoCorrect tab (in Word 2007 and later):

- Uncheck Capitalize first letter of sentences
- Uncheck Capitalize first letter of table cells
- Uncheck Replace text as you type

#### AutoFormat As You Type

In Tools | AutoCorrect Options | AutoFormat As You Type tab (in Word 2003), or Word Options | Proofing | AutoCorrect Options | AutoFormat As You Type tab (in Word 2007 and later):

- Uncheck "Straight quotes" with "smart quotes"
- Uncheck Automatic bulleted lists
- Uncheck Automatic numbered lists
- Uncheck Format beginning of list item like the one before it
- Uncheck Set left- and first-indent with tabs and backspaces
- Uncheck Define styles based on your formatting

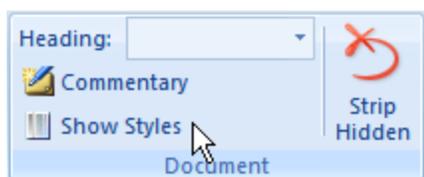
## Measurement Units and Style Area

Set the units of measurement to Centimeters and the Style Area Width to about 3cm – this will help you to see what is happening with the Oracle Policy Modeling styles.

For Word 2003:

- In Tools | Options | General tab, change Measurement units to centimeters.
- In Tools | Options | View tab, set the Style area width to 3cm.

For Word 2007 and later, the Show Styles button in the Document group of the Oracle Policy Modeling tab provides a shortcut to display the style area.



The settings to do this manually in Word 2007 and later can be found in Word Options | Advanced | Display:

- Change Show measurement in units of: to Centimeters.
- Set the Style area pane width in Draft and Outline views: to 3cm. Note that you will need to select the Draft or Outline Document Views while you are using Word in order to see this.

TIPS:

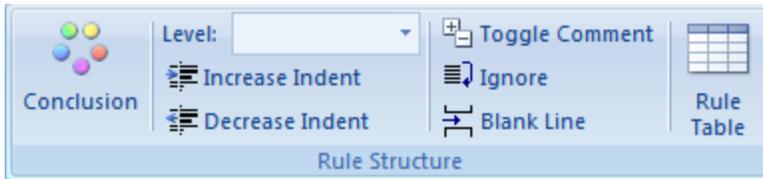
- i. Make sure that the rule language and the dictionary language in Microsoft Word are synchronized (eg if the rule language is English (American), the dictionary language in Word should be English (U.S.)).
- ii. For extremely complex projects containing either very large rule documents (70+ pages) or large numbers of rule documents, you should also turn off auto-saving, backup and background repagination to improve the performance of Microsoft Word with Oracle Policy Modeling.

## Understand Oracle Policy Modeling format and structure

Oracle Policy Modeling format is quite strict in order to maintain consistency and completeness of rules and to avoid logical ambiguity. In particular, styles and indentation play an important role in recognizing the meaning of rules. Indentation and styles are used to separate the conditions from the conclusion, and conditions of different levels from each other. Distinct conditions are separated onto different lines, and the placement of **and** and **or** between conditions has special significance.

Rules need to be marked up in Word using Oracle Policy Modeling styles in order to be recognized by the Oracle Policy Modeling compiler. The styles appear in the Oracle Policy Modeling toolbar and in the document templates which are attached to all Word documents created through Oracle Policy Modeling. Oracle Policy Modeling looks for these styles when parsing your rules to determine the various rule components. Each style has a unique style name and coloring to make it easy to identify. Text which is not in the Oracle Policy Modeling styles is ignored by the Oracle Policy Modeling compiler.

The rule below shows an example of the OPM styles that would be applied in Word using the Conclusion and Level styles on the Oracle Policy Modeling tab:



**the claimant is eligible for living allowances if OPM - conclusion**

the claimant is living alone and OPM - level 1  
the claimant satisfies the age criteria OPM - level 1  
    the claimant satisfies the male age criteria OPM - level 2  
        the claimant is aged over 65 and OPM - level 3  
        the claimant is a man OPM - level 3  
    or OPM - level 2  
    the claimant satisfies the female age criteria OPM - level 2  
        the claimant is aged over 60 and OPM - level 3  
        the claimant is a woman OPM - level 3

### Write a rule in Word

To write a rule in Word:

1. **Create and open a Word document** in your project.  
In Word you will notice the Oracle Policy Modeling toolbar. (If the toolbar is not visible in Word 2003, go to **View | Toolbars | Oracle Policy Modeling** to open it.) This toolbar is what you will use to format your rules in Oracle Policy Modeling styles.
2. Put the cursor on a new blank line in the Word document.
3. Type "the person is happy if". This will form your rule conclusion.
4. Place the cursor somewhere in this text and click the **Conclusion** button on the Oracle Policy Modeling toolbar.
5. Place the cursor at the end of the line (after the "if") and press the **Enter** key to start a new paragraph.  
The Level 1 style will automatically be applied to the new paragraph. You will notice that the Level 1 style is indented slightly from the Conclusion style to highlight the difference in rule levels.
6. Type "the sun is shining".  
That's it! You have just created a rule in Word.

### Define rule tables in Word documents

When using multiple rules to prove an attribute, you must be extremely careful to ensure that you have closed the logic with your rules. If all of the rules proving your conclusion (goal) attribute do not provide full logical coverage, your rules will not cover every possible situation.

Imagine that you wanted to add the following rules to your model:

**the passenger's favorite color = "blue" if**

the passenger selected the blue seat

**the passenger's favorite color = "orange" if**

the passenger selected the orange seat

**the passenger's favorite color = "purple" if**

the passenger selected the purple seat

If there were a fourth seat color (eg "olive"), then the rules would not cope with that situation.

Instead of using multiple rules to prove the goal, you use rule tables to cover this situation. Rule tables provide an invisible layer of truth management by enforcing the effective creation of additional conditions and enforcing question order to avoid goal exhaustion when your rules are built.

The following diagram shows how this table must be structured:

<b>conclusion</b>	
<b>value</b>	condition
<b>value</b>	condition
<b>value</b>	<b>otherwise</b>

The first row of the table defines which attribute will be used as the conclusion attribute for the rule.

The left hand column is used to specify values (includes mathematical expressions) which will set the value of the conclusion attribute if the condition in the right hand column of the same row is satisfied.

The final row provides an alternative conclusion, to which the conclusion will be set if none of the conditional rows are satisfied.

## Add a rule table in Word

To add a rule table in Word:

1. Place the cursor on a new blank line in your Word rules document and click the **Rule Table** button on the Oracle Policy Modeling toolbar.  
A pre-formatted table will be inserted.

	<b>otherwise</b>

2. Enter your conclusion in the first row of the table.

the passenger's favorite color	
	<b>otherwise</b>

3. In each subsequent row of the table enter a value in the left hand column and the condition that sets it in the right hand column.

the passenger's favorite color	
"blue"	the passenger selected the blue seat
"orange"	the passenger selected the orange seat
"purple"	the passenger selected the purple seat
	<b>otherwise</b>

4. In the final row, enter a value for the alternative conclusion in the left hand column.

the passenger's favorite color	
"blue"	the passenger selected the blue seat
"orange"	the passenger selected the orange seat
"purple"	the passenger selected the purple seat
<b>uncertain</b>	<b>otherwise</b>

## Define decision tables in Excel workbooks

To author rules in Excel, you simply write rules in tables, and use Oracle Policy Modeling styles to identify the type of information in the cells so that they can be compiled for use with the Oracle Determinations Engine. You can have as many worksheets for rules in your document as you need.

### What do you want to do?

Understand the styles used for rule tables

Create a rule table in Excel

Prove multiple attributes for the same set of conditions

Prove the same set of conclusions using multiple conditions

Allow rule conditions to evaluate in any order and handle missing values

Write a comparison type rule where a decision applies to a range of numbers or dates

Split rule tables according to the date they apply from

Use entity attributes in an Excel rule table

Prove a text attribute in an Excel rule

Understand the styles used for rule tables

Excel rules which are intended for compiling in Oracle Policy Modeling need to be marked up using the styles supplied with the Oracle Policy Modeling Excel document template. The following styles are used for writing rules:

Style Name	Description
<b>Conclusion Heading</b>	Used to mark up a conclusion column in a rule block. The text is either "conclusion" or an attribute ID.
Conclusion	Used to mark up an attribute that will be concluded by a rule
<b>Condition Heading</b>	Used to mark up a condition column in a rule block. The text is either "condition" or an attribute ID.
Condition	Used to mark up a condition for a part of a rule. If the condition header is "condition", the condition must be a complete expression or a valid boolean attribute. If the condition header is an attribute ID, the condition must be either a constant or a comparison of the same type as the attribute.
<i>Else</i>	Used to mark up the else condition
Commentary	Used to mark up descriptive text in a rule block. The text is ignored when generating the rule.

The heading cells are optional. Similarly, the order of cells is irrelevant since each style is unique - as long as the necessary styles are used with valid cell contents.

NOTES:

- i. Regardless of the order of declaration on a worksheet, the order of processing is "global entity", "entity" and then any attributes. This ensures that attributes appear in the correct entity.
- ii. To format a cell as a currency value, do not use the  button on the Excel formatting toolbar - instead go to **Format | Cells** and select **Currency** on the **Number** tab.
- iii. When working with numbers, currencies, dates and time in Microsoft Excel, the regional setting of the computer should accord with the [rulebase project's region](#). This is because Microsoft Excel formats the data types using the templates in the regional setting.
- iv. If you use a text attribute you can either put the value of that text attribute in quotes or not in quotes and it will be treated the same way.
- v. If you want to use a text function in a rule table, you need to put the function text in parentheses.

Create a rule table in Excel

When you [add an Excel document](#) to your project, it will contain a rule template on the **Rule Table** worksheet that looks like this:

condition	condition	conclusion	conclusion
commentary			
	<i>else</i>		

To write a simple rule in Excel which contains a single condition and a single conclusion, follow the steps below. In this example we will be concluding the nationality of the individual based on their country of citizenship. NOTE: Variable attributes should be **declared** in a properties file before use in Excel. (There is no need to declare boolean attributes before using them in rules.) In this example, the text variables "the country of citizenship" and "the nationality of the individual" have already been declared in the properties files in the project.

1. Replace the text **condition** in the second column with "the country of citizenship". This cell is already in the correct **Condition Heading** style. As we will only be having one set of conditions you can delete the first **condition** column.
2. Replace the text **conclusion** with "the nationality of the individual". This cell is already in the correct **Conclusion Heading** style. As we will only be having one set of conclusions you can delete the other **conclusion** column.
3. Type "USA" in the cell below the "the country of citizenship" cell. Tab across to the next cell (the cell below the "the nationality of the individual" cell) and type "American". These cells are already in the correct styles: **Condition** and **Conclusion** respectively. Delete the next two rows, as they won't be used.
4. In the row below, enter another condition "Scotland" with the associated conclusion "Scottish". Follow this on the next row with another condition "Japan" and conclusion "Japanese".
5. Type "uncertain" in the cell next to the **else** condition. This applies an alternative conclusion of "uncertain".

Your rule table should look like this:

the country of citizenship	the nationality of the individual
USA	American
Scotland	Scottish
Japan	Japanese
<i>else</i>	uncertain

Decision tables written in Excel are converted into internally generated rule tables by Oracle Policy Modeling when the rules are compiled. The table above will create the following rule (xgen) in Oracle Policy Modeling. (This can be viewed in OPM by right-clicking on the rule document in the Project Explorer and selecting **Open Rule Browser**.)

the nationality of the individual	
<b>Rule Tables.xgen</b>	
a1: the nationality of the individual	
"American"	a2: the country of citizenship = "USA"
"Scottish"	a2: the country of citizenship = "Scotland"
"Japanese"	a2: the country of citizenship = "Japan"
<b>"uncertain"</b>	<b>otherwise</b>

### Prove multiple attributes for the same set of conditions

Using just one table in Excel you can prove multiple attributes for the same set of conditions (unlike in Word which would require multiple rule tables).

Assuming you have the following variables already declared, the text variables "the country of citizenship", "the nationality of the individual" and "the currency of the country", you could have the following rule table:

the country of citizenship	the nationality of the individual	the currency of the country
USA	American	Dollar
Scotland	Scottish	Pound
Japan	Japanese	Yen
<i>else</i>	uncertain	uncertain

Prove the same set of conclusions using multiple conditions

You can specify multiple conditions for a particular conclusion in Excel, merging the conclusion cells if appropriate to influence the way the rule is evaluated.

For example, you may wish to determine the appropriate ticket type for different combinations of adults and children. If you have the following variables:

Attribute Type	Attribute Text	Legend Key
Number	the number of adults in the group	Adults
Number	the number of children in the group	Children
Text	the ticket type	Ticket

you may have the following rule table:

Adults	Children	Ticket
1	0	Single
1	1	Double
2	0	Double
2	1	Family
2	2	Family
2	3	Family
3	0	Family
<i>else</i>		Combo

The rule generated for this table in Oracle Policy Modeling will look like the following:

the ticket type	
Multiple conclusions unmerged.xgen	
ticket_type: the ticket type	
"Single"	<b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 0
"Double"	<b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 1
"Double"	<b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 0
"Family"	<b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 1

the ticket type	
Multiple conclusions unmerged.xgen	
"Family"	<b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 2
"Family"	<b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 3
"Family"	<b>all</b> number_adults: the number of adults in the group = 3 <b>and</b> number_children: the number of children in the group = 0
"Combo"	<b>otherwise</b>

We can leave a condition cell empty if we do not wish to test the value of the attribute for that conclusion cell. In our example, we may decide that two adults can enter under a Family ticket if they have any children with them, and three adults can be covered by a Family ticket regardless of whether there are children with them.

Adults	Children	Ticket
1	0	Single
1	1	Double
2	0	Double
2		Family
3		Family
	<i>else</i>	Combo

This will simplify the logic, and the rule generated:

the ticket type	
Multiple conclusions simplified.xgen	
ticket_type: the ticket type	
"Single"	<b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 0
"Double"	<b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 1
"Double"	<b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 0
"Family"	number_adults: the number of adults in the group = 2
"Family"	number_adults: the number of adults in the group = 3
"Combo"	<b>otherwise</b>

We can also [merge the cells](#) for the conclusion values, if there are multiple condition rows that prove the same conclusion.

Adults	Children	Ticket
1	0	Single
1	1	Double
2	0	
2		Family
3		
<i>else</i>		Combo

This will simplify the appearance of the Excel rule table and emphasize that the value inferred for Ticket will be the same in more than one possible scenario. However, it will also change the way Oracle Policy Modeling interprets the logic of the rule. The internal rule table generated from an Excel rule table includes a row for each Excel conclusion cell. This means that instead of having two rows in the generated rule table proving the same conclusion value (which will be evaluated in order from the top down), we now have a single row proving the conclusion value, with multiple options that may be evaluated in any order. This can be useful if our rules need to [allow for some condition values being unknown](#).

the ticket type	
Multiple conclusions merged.xgen	
ticket_type: the ticket type	
"Single"	<b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 0
"Double"	<b>either</b> <b>all</b> number_adults: the number of adults in the group = 1 <b>and</b> number_children: the number of children in the group = 1 <b>or</b> <b>all</b> number_adults: the number of adults in the group = 2 <b>and</b> number_children: the number of children in the group = 0
"Family"	<b>either</b> number_adults: the number of adults in the group = 2 <b>or</b> number_adults: the number of adults in the group = 3
"Combo"	<b>otherwise</b>

TIP: To see an example of a complete rulebase with merged condition and conclusion cells, open and run the [Insurance Fraud Score example rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### Allow rule conditions to evaluate in any order and handle missing values

The internal rule tables that are generated by Oracle Policy Modeling from decision tables in Excel are evaluated row by row from the top down. If the first row of a table cannot be evaluated (ie if some of the condition values are unknown), then the evaluation of the rule table as a whole will not progress beyond that row, even if a later row in the same table can be evaluated because all of its condition values are fully known.

In some cases, this may not be the most useful way for the rule to evaluate. If a single conclusion is proved in multiple ways, you can merge a single conclusion cell across all of the different condition rows. Oracle Policy Modeling will then allow any of those condition rows to prove the conclusion value, in any order.

For example, in the following rule cells we would like either of the two rows to be able to prove the conclusion.

Occupation	Age	Entitlement
Student		TRUE
	16	TRUE

With the current rule table layout, the rule generated by Oracle Policy Modeling will have separate rows for each of the rows in our Excel rule. Because a rule table evaluates from the top down, this will mean that even if we know that a person is 16 and hence is entitled to Youth Benefit, the rule table would be unable to conclude a result until we know the person's occupation and can evaluate the first row.

the applicant is entitled to the benefit	
Handle missing data unmerged.xgen	
applicant_entitlement: the applicant is entitled to the benefit	
<b>true</b>	applicant_occupation: the applicant's occupation = "Student"
<b>true</b>	applicant_age: the applicant's age = 16
<b>uncertain</b>	<b>otherwise</b>

However, if we [merge the cells](#) containing the conclusions that apply to these two rows, the internal rule generated by Oracle Policy Modeling combines these rows with an "or" condition in a single rule table row, rather than the two separate rule table rows generated above.

Occupation	Age	Entitlement
Student		TRUE
	16	

This new structure allows the conditions proving the conclusion to be evaluated in any order, so the second row will now allow the rule to be evaluated even if the first row values are unknown.

the applicant is entitled to the benefit	
Handle missing data merged.xgen	
applicant_entitlement: the applicant is entitled to the benefit	
<b>true</b>	<b>either</b> applicant_occupation: the applicant's occupation = "Student" <b>or</b> applicant_age: the applicant's age = 16
<b>uncertain</b>	<b>otherwise</b>

Write a comparison type rule where a decision applies to a range of numbers or dates

For non-text conditions, it is likely that the decision will apply to a range of numbers or dates rather than to a specific number or date. A simple example is the mapping of taxable income to tax rates for a particular date range:

Attribute Type	Attribute Text	Legend Key
Date	the assessment date	Assessment Date
Currency	the client's taxable income	Taxable Income
Number	the client's tax rate	Tax Rate

Assessment Date		Taxable Income		Tax Rate
≥2006-07-01	<2007-07-01	>=0	<12000	0
		>=12000	<24000	0.22
		>=24000	<36000	0.27
		>=36000	<48000	0.36
		>=48000		0.48
≥2005-07-01	<2006-07-01	>=0	<12000	0
		>=12000	<24000	0.22
		>=24000	<36000	0.27
		>=36000	<48000	0.35
		>=48000		0.47
≥2004-07-01	<2005-07-01	>=0	<12000	0
		>=12000	<24000	0.21
		>=24000	<36000	0.26
		>=36000	<48000	0.34
		>=48000		0.46
<b>else</b>				0.5

It is also possible that you may want to have multiple comparisons for one attribute as exemplified below:

Attribute Type	Attribute Text	Legend Key
Number	the current temperature	Temp
Text	the person's gender	Gender
Text	the state the person is likely to be in	State

Temp	Temp	Gender	State
	<=0		Freezing
>0	<12	male	Cold
>0	<16	female	Cold
>=20	<24		Comfortable
>30			Hot
<b>else</b>			Uncertain

### Split rule tables according to the date they apply from

Tables can be split over several sheets in the same file to allow for regular table updates that apply from a particular date. This is managed by the insertion of a master table that prioritizes the sheets. The prioritization is done by reference to sheet name, which is specified in the tab for the sheet. For example, you could have:

Attribute Type	Attribute Text	Legend Key
Text	the type of ticket	Ticket
Currency	the ticket price	Price
Date	the date of purchase	Date

Date	Apply Sheet
>= 2006-07-01	2006-2007
>= 2005-07-01	2005-2006
<i>else</i>	pre 2005-2006

The logic of these tables is consolidated on compile, and therefore does not result in multiply proven attributes. Master tables use the standard rule condition and conclusion styles but have a single conclusion column headed "Apply Sheet" in the **Conclusion Heading** style. Note that the text "Apply Sheet" therefore cannot be used as a column heading in a standard rule table.

In this example, you would have three other worksheets which contain the rule tables below. Note that the worksheets must be titled (case-sensitive) according to the names given in the Apply Sheet column.

pre 2005-2006

Ticket	Price
Adult	14
Concession	10
Child	6
<i>else</i>	14

2005-2006

Ticket	Price
Adult	16
Concession	12
Child	8
<i>else</i>	16

2006-2007

Ticket	Price
Adult	20
Concession	15
Child	10
<i>else</i>	20

This will create the following rule in Oracle Policy Modeling:

the ticket price	
Split tables.xgen	
price_ticket: the ticket price	
20	<b>all</b> ticket_type: the type of ticket = "Adult" <b>and</b> purchase_date: the date of purchase >= 07/01/2006
15	<b>all</b> ticket_type: the type of ticket = "Concession" <b>and</b> purchase_date: the date of purchase >= 07/01/2006
10	<b>all</b>

the ticket price	
Split tables.xgen	
	ticket_type: the type of ticket = "Child" <b>and</b> purchase_date: the date of purchase >= 07/01/2006
20	purchase_date: the date of purchase >= 07/01/2006
16	<b>all</b> ticket_type: the type of ticket = "Adult" <b>and</b> purchase_date: the date of purchase >= 07/01/2005
12	<b>all</b> ticket_type: the type of ticket = "Concession" <b>and</b> purchase_date: the date of purchase >= 07/01/2005
8	<b>all</b> ticket_type: the type of ticket = "Child" <b>and</b> purchase_date: the date of purchase >= 07/01/2005
16	purchase_date: the date of purchase >= 07/01/2005
14	ticket_type: the type of ticket = "Adult"
10	ticket_type: the type of ticket = "Concession"
6	ticket_type: the type of ticket = "Child"
14	<b>true</b>
<b>uncertain</b>	<b>otherwise</b>

TIP: To see an example of a complete rulebase using 'Apply Sheet' to reason about attributes that change over time, open and run the [Insurance Fraud Score example rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### Use entity attributes in an Excel rule table

You can prove entity-level attributes in Excel rule tables, however, all conclusion attributes in the table must be in the same entity. The condition attributes in the rule table may be in the same entity as the conclusion, or they may reference any entities in the containment relationships of the conclusion entity.

For example, the following rule table infers conclusion attributes in "the pet" entity, using condition attributes in the entity "the child" and the global entity, which are both in its containment relationship as shown:



the grocery shopping has been done	the child is on school holidays	the pet is happy	the pet is well fed
TRUE	TRUE	TRUE	TRUE
FALSE		FALSE	FALSE
	FALSE	FALSE	
	<b>else</b>	uncertain	uncertain

Entity level attributes can also be used in condition cells with most [entity functions](#). For example, the following rule uses the InstanceCount function to set the child's pocket money depending on how many pets she owns.

condition	the amount of pocket money the child gets
the number of the child's pets = 0	\$5.00
the number of the child's pets = 1	\$8.00
the number of the child's pets = 2	\$10.00
<i>else</i>	\$15.00

NOTE: The entity functions that cannot be used in this way in Excel are those which deal with multiple entities: ForScope, ForAllScope, ExistsScope, IsMemberOf, IsNotMemberOf, InstanceEquals, InstanceNotEquals.

TIP: To see an example of a complete rulebase using entity level attributes, functions and calculations based on entity instances, open and run the [Insurance Fraud Score example rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### Prove a text attribute in an Excel rule

When proving a text attribute in an Excel rule, you need to enclose the attribute text in parentheses so that the compiler recognizes it as an attribute.

For example, if you had the following declarations:

Attribute Type	Attribute Text
Text	the location of the overall winner
Text	the winner of the overall award
Text	the winner of the award in Australia
Text	the winner of the award in Japan
Text	the winner of the award in the UK
Text	the winner of the award in the US

you would need to put the text attribute's text in parentheses when it is being concluded in a rule table. For example:

the location of the overall winner	the winner of the overall award
Australia	(the winner of the award in Australia)
Japan	(the winner of the award in Japan)
United Kingdom	(the winner of the award in the UK)
United States	(the winner of the award in the US)
<i>else</i>	uncertain

If you had not put the parentheses around these text attributes in the rule, these attributes would not be recognized and the resulting rule would conclude the literal strings.

When concluding a specific value for a text attribute it is not necessary to enclose it in parentheses (note that you can either put the value of that text attribute in quotes or not in quotes and it will be treated the same way).

See also:

- [Make your Excel rules easier to understand](#)

## Make your Excel rules easier to understand

There are several ways in which you can make your Excel rules easier to understand.

### What do you want to do?

Shorten attribute names in Excel workbooks

Simplify rule table layout by merging cells

Change rule table orientation

### Shorten attribute names in Excel workbooks

You can create an abbreviated way of referring to an attribute in Excel using a Legend Key. Specification of this abbreviated form is optional.

To specify a legend key:

1. In Excel, open your **Declarations** worksheet.
2. Next to your **Attribute Type** and **Attribute Text** columns, add the title "Legend Key". Click on the **Legend Key Heading** button on the Oracle Policy Modeling toolbar to set the style of this cell. NOTE: This column is already there in the default Excel worksheet so you will only need to do this step if you have manually deleted the Legend Key column at some stage.
3. Next to each attribute (in the Legend Key column) specify the abbreviated attribute name. Use the **Legend Key** button on the Oracle Policy Modeling toolbar to set the style of these cells.
4. Open you **Rule Table** worksheet. You can now use the legend key text as **Condition Headings** and **Conclusion Headings**.

For example, if you have the following declaration:

Attribute Type	Attribute Text	Legend Key
Text	the country of citizenship	Country
Text	the nationality of the individual	Nationality

you could have the following rule table:

Country	Nationality
USA	American
Scotland	Scottish
Japan	Japanese
<i>else</i>	uncertain

You can also use legends with tables which use boolean attributes.

For example, if you have the following declaration:

Attribute Type	Attribute Text	Legend Key
Number	the individual's age	Age
Boolean	the individual is disabled	Disabled
Boolean	the individual is entitled to compensation	Compensation

you could have the following rule table:

Age	Disabled	Compensation
<18	TRUE	TRUE
>65	TRUE	TRUE
	FALSE	FALSE
	<i>else</i>	FALSE

Simplify rule table layout by merging cells

Looking at the multiple condition example below, we note that the values for the Adults condition cells consist of only three unique values 1, 2 and 3.

Attribute Type	Attribute Text	Legend Key
Number	the number of adults in the group	Adults
Number	the number of children in the group	Children
Text	the ticket type	Ticket

Adults	Children	Ticket
1	0	Single
1	1	Double
2	0	Double
2	1	Family
2	2	Family
2	3	Family
3	0	Family
	<i>else</i>	Combo

We can choose to merge the cells in this column that share the same value. To merge cells in Excel, select the cells that you want to merge and then click the  **Merge & Center** button on the Excel formatting toolbar. You may get a warning that advises that merging will keep the upper-left most data only. Click **OK**.

Adults	Children	Ticket
1	0	Single
	1	Double
2	0	Double
		Family
		Family
3		Family
	<i>else</i>	Combo

This table is equivalent (in function) to the original table, but allows us to emphasize that only three distinct values are used for Adults, and the rows that they cover.

You can also merge conclusion cells, however note that this will change the structure of the rule logic slightly. See [Prove the same set of conclusions using multiple conditions](#) and [Allow rule conditions to evaluate in any order and handle missing values](#) for further details.

## Change rule table orientation

Typically a rule table will be specified with the conclusion and conditions listed left-to-right in separate columns, and each set of conditions and conditions listed in separate rows, as shown below. (NOTE: In this example "can be trusted" represents the boolean attribute "the user is of a trustworthy nature".)

condition	taxable income		the user can be trusted	the risk level of the user
the user is applying for some money	<=100	> 0	FALSE	high
	<=2000	> 100		high
	<=50000	> 2000	TRUE	medium
		>50000		low

It is possible to rotate a rule table such that the rows and columns are swapped. This effectively means that we represent a rule table in the Y-X orientation rather than the X-Y orientation. For this example, the rotated rule table would be:

condition	the user is applying for some money			
taxable income	<=100	<=2000	<=50000	
	> 0	> 100	> 2000	>50000
the user can be trusted	FALSE		TRUE	
the risk level of the user	high	high	medium	low

Both rule tables will generate the exact same rules when compiled.

## Capture implicit logic in rules

Shortcut rules are a type of application rule used to capture implicit logic which does not automatically flow from source rules. Shortcut rules only participate in [inferencing](#) and do not participate in the question search, and are therefore useful in streamlining interviews.

## What do you want to do?

[Understand how shortcut rules work](#)

[Write a shortcut rule](#)

## Understand how shortcut rules work

To understand how a shortcut rule works, consider the following two statements:

The claimant has lived in America for more than 50 years

The claimant has lived in America for more than 20 years

The second statement must be true if the first is true. On the other hand, the reverse is not the case, although it might be true (ie it is not necessarily false).

This type of situation needs to be captured where rules are used in a software application, to avoid situations where the application asks redundant questions (just imagine answering that you had lived in America for 50 years, then being asked if you'd lived there for 20 years!).

If we try to model this logic using our default rule format, we would have the following rule:

**the claimant has lived in America for more than 20 years if**

the claimant has lived in America for more than 50 years

When compiled, this rule would automatically be given an alternative conclusion. An alternative conclusion for this rule would not be correct, however, as it is not necessarily the case that the claimant has not lived in America for 20 years just because they have not lived there for 50 years.

Assume the rule did **not** have an alternative conclusion. While investigating the conclusion, the question search would traverse this rule and try to prove the condition. Assuming that no other rule proved the conclusion, if the condition returned false, there would be no alternative conditions to investigate, resulting in the goal being exhausted.

Instead, we can define this rule as a shortcut rule and it will then provide the required logic. As a shortcut rule, if the condition is set to a value of true following the operation of the question search on other rules, inferencing operates to set the value of the conclusion attribute. If the value returned is not true, this rule will not fire, and the conclusion attribute will not be set. Alternative conclusions are not set for shortcut rules.

It is possible for attributes proved in shortcut rules to be interrelated in a logical loop, because no alternative conclusions are set and because they are not traversed by the question search. So in addition to the shortcut rule above, it is also possible to have the following shortcut rule:

**the claimant has not lived in America for more than 50 years if**

the claimant has not lived in America for more than 20 years

NOTE: Shortcut rules should only be used when you can prove a base level attribute before it is asked.

### Write a shortcut rule

To write a shortcut rule in Microsoft Word, click the **Shortcut Rule** button on the Oracle Policy Modeling toolbar (or press F7) to add a shortcut rule template.

Essentially, the only difference between the format of a shortcut rule and a standard rule is that the rule has an additional paragraph above it which uses the **Rule Type** style and reads "shortcut rule". Lacking this heading, the rule will be given an alternative conclusion, and will participate in the question search, causing goal exhaustion. The rule template also leaves a line for you to provide a rule name.

For example,

*shortcut rule*

**the claimant has lived in America for more than 20 years if**

the claimant has lived in America for more than 50 years

It is also possible to use rule tables for writing shortcut rules. For example,

*shortcut rule*

<b>the claimant lives in Australia</b>	
<b>true</b>	the claimant lives in Sydney
<b>true</b>	the claimant lives in Canberra

<b>the claimant lives in Australia</b>	
<b>false</b>	the claimant lives in London

NOTE: You must not include an alternative conclusion in your rule table for a shortcut rule.

## Write rules in the negative

Attributes may be expressed in either the positive form ("the person is happy") or negative form ("the person is not happy") in both conclusions and conditions. For example, you may write "the person is not happy" and the rule engine will recognize this as the negative form of "the person is happy". When compiled, the attribute will be marked up like this:

**[not b10] the person is not happy**

## Avoid multiple conclusions when writing negative rules

Repeating a conclusion can result in two conflicting criteria for the conclusion to be satisfied. For example:

**[b11] the person is considered an employee if**

**[b3]** the person works set hours

and

**[not b11] the person is not considered an employee if**

**[b4]** the person owns the equipment required to do the job

In this example, if a person works set hours and owns the equipment, is the person an employee or not? The logic is unclear. The logic needs to be grouped or prioritized to make that decision.

When using both the positive and negative forms of an attribute within a rulebase, take care that the attribute is only concluded once.

To avoid repeating the conclusion, an exclusion clause can be linked in to the conclusion rather than simply restating the same conclusion in the negative. For example:

**[b11] the person is considered an employee if**

**[b3]** the person works set hours and

**[not b4]** the person does not own the equipment required to do the job

## Prove an attribute using multiple rules

A multiply proven attribute is an attribute that appears as the conclusion attribute of more than one rule (including shortcut rules).

An attribute which is proven by multiple rules like this will not function correctly in the Engine because of the operation of the automatic alternate conclusion in every rule. The closed logic of alternative conclusions will prevent multiple rules being traversed - the first rule traversed will close off the possibility of the other forms operating.

## What do you want to do?

[Intentionally prove an attribute using multiple rules](#)

[Check my rules for multiply proven attributes](#)

## Intentionally prove an attribute using multiple rules

The nature of the business rule domain and methodological factors may mean you need to have attributes with multiple, or distributed (non-adjacent), proofs in your rulebase. For such rules it is necessary to designate the rule as a 'rule fragment'. A 'priority' needs to also be specified for these rules to guarantee predictable question searches and inferencing. Lower numbers indicate higher priority (ie priority 1 takes precedence over priority 2 rule fragments).

To create rule fragments:

1. In Microsoft Word, write your rules.
2. Place the cursor somewhere in the first rule and click the **Rule Properties Editor** button on the Oracle Policy Modeling toolbar.
3. Select the **Rule Fragment** check box.
4. Specify a **priority**.
5. Click **OK**. Repeat steps 2-5 for each rule.

You will notice that your rules in Word are now preceded by a configuration line which indicate that the rule is a rule fragment and shows the priority.

At rulebase build time, rules marked as rule fragments will be automatically combined into a single rule using an inclusive *or* operator, and will use the author-assigned priorities to determine the question order.

A rule without a proof becomes the default alternative conclusion for the collection of rules if it has a lower priority than the other rules. So, for example, if you wanted to specify an otherwise clause for these rule fragments:

```
rule_property[fragment:1]
```

```
the claimant's queue position = 1 if
```

```
    the claimant has been queuing longer than anyone else
```

```
rule_property[fragment:2]
```

```
the claimant's queue position = 1 if
```

```
    the claimant has jumped to the start of the queue
```

You would need to also have a rule without conditions which specifies the alternative conclusion, ie:

```
rule_property[fragment:3]
```

```
the claimant's queue position = 2
```

Then if the claimant has not been queuing longer than anyone else and has not jumped to the start of the queue, then the result would be that the claimant's queue position is 2.

If this alternative conclusion rule is not provided, the result will be uncertain when the other rules are disproven. That is, if the claimant has not been queuing longer than anyone else and has not jumped to the start of the queue then the result would be uncertain.

## Check my rules for multiply proven attributes

Oracle Policy Modeling automatically runs a check for Multiply Proven Attributes when building a rulebase. If any multiple proven attributes are detected, an error will be logged in the Error List and the build will be canceled. To see which attributes are proven by more than one rule, you can generate a Multiply Proven Attributes Report. To do this, select **Reports | Multiply Proven Attributes** from the main menu in Oracle Policy Modeling.

Attributes that are concluded in rules marked as rule fragments will not fail the Multiply Proven Attributes check at build time, and will not appear in the Multiply Proven Attribute report (unless they are multiply-proven by some other "normal" rule).

The [identifying attribute of an entity that is inferred via different relationships](#) will appear in a Multiply Proven Attributes report. Here the identifying attribute is technically multiply-proven because there are multiple rules that cause it to have a value. But the attributes don't actually conflict and aren't problematic, because by definition, if two different values are inferred, then two instances are created and the two values can peacefully co-exist. (However, if another "normal" rule independently inferred the identifying attribute, then there would be a conflict there.)

## Model loops in rule logic

Generally, having loops in your rule logic should be avoided, as they can result in rules which can never be proven, and in unintended behavior in your rulebase, if the logic is not carefully checked. To prevent this situation occurring accidentally, Oracle Policy Modeling will validate your rulebase for rule loops when you build the project.

However, in some situations, in particular when working with rules using entity instances, it may be desirable for a controlled logic looping situation to be created within rules.

Consider an example where a person entity is used to model the person's citizenship. The person's citizenship status may be inferred from their place of birth, or it may be inferred from the citizenship status of one of their parents. If the person's parents are also instances of the person entity, represented by the self-referential relationship "the person's parents", then we may wish to create a rule to model the logic as follows:

### **the person is a citizen if**

the person was born in the country or  
at least one of the person's parents is a citizen

This rule contains a logical loop, in that "the person is a citizen" is both proved and used (via "the person's parents" relationship) in the same rule. However the logic of the scenario we wish to model is sound. We can allow this logical looping to be a valid part of the rulebase by defining the above rule as a rule loop.

To define a rule as a rule loop:

1. Define the line above your rule as a Configuration line (use the Configuration button in the Oracle Policy Modeling toolbar, or use the keyboard shortcut Alt+F).
2. Enter the text "rule\_loop" in the Configuration line.

rule\_loop

### **the person is a citizen if**

the person was born in the country or  
at least one of the person's parents is a citizen

If the logical loop encompasses multiple rules, each rule must be defined as a rule loop.

TIP: It is important to ensure that the logic of the rule allows an alternative way to prove the conclusion without using the rule loop logic, to avoid having the rule loop endlessly. In the example above, the rule premise "the person was born in the country" provides this.

NOTE: When introducing logic loops into your rules in this way, it is very important that the rules be tested thoroughly to ensure no unintended behavior results in the rulebase.

See also:

- [Fix a build error](#)
- [Capture implicit logic in rules](#)

## Include an existing attribute in a rule

At the project level, Oracle Policy Modeling automatically links all attributes with the same text together and treats them as one attribute. This means you can write your rules in any document, or in a number of documents, in any order within those documents, and Oracle Policy Modeling will link them all together for you, provided the same attribute text (including capitalization) is used.

So, once a variable attribute has been added to a properties file, it can be used in any rule in any rules document (Word or Excel). Similarly, once you have written a rule using a boolean attribute, that boolean attribute can be used in any rule. A condition of one rule will be automatically linked to the conclusion of another rule if the attribute text is exactly the same.

Once an attribute is linked with another they are logically collapsed within the Oracle Policy Modeling model and displayed as a single item in the Data Model and Build Model views.

To ensure you are using an existing attribute in a rule (not inadvertently creating a new one with very similar text):

1. In your Word rule document, click on the **Data Model Browser** button on the Oracle Policy Modeling toolbar.
2. On the Attributes tab, right-click on the text of the attribute and select **Copy Text to Clipboard**.
3. In the appropriate place in your rule, press **Ctrl+V** to paste the attribute text.

## Choose a function to include in a rule

Functions are used to extend the capabilities of expressions. These are useful for performing a number of common calculations which frequently appear in rules.

There are many different types of functions that you can use in your rules:

Function Type	Use
<a href="#">Numerical functions</a>	Used with number and currency variables to perform basic and complex arithmetic calculations, trigonometric calculations and maximum/minimum calculations
<a href="#">Date functions</a>	Used with date variables to express the current date (based on the system date at the start of the session), to calculate a relative date, to find a date in a year, to get particular dates/days/months/years, to count periods between two dates, and to get an earliest/latest date
<a href="#">Time of day functions</a>	Used with time of day variables to express the current time of day, to set the time of day, to calculate the difference in seconds/minutes/hours between two times of day, to extract the

Function Type	Use
	second/minute/hour from a time of day, and to get an earliest/latest time of day
Date and time functions	Used with date and time variables to express the current date and time (based on the system date/time at the start of the session), to set the date and time, to calculate the difference in units between two dates, to extract a unit from a date and time, to extract a time of day, and to get an earliest/latest date and time
Text functions	Used with text variables to combine text strings and to extract parts of text strings
Entity and relationship functions	Used to perform operations on entity-specific data to produce global results, such as counting the number of instances of an entity, obtaining the highest/most recent or lowest/least recent value of an entity-level variable, and adding up numerical values gathered from each instance of the entity
Temporal reasoning functions	Used in rules to compute results for, and express relationships that involve, attributes over multiple periods

NOTE: If you have a project which uses a RLS (Rapid Language Support) parser, the syntax for the functions are defined in the configuration for that particular RLS parser. For more information on using an RLS parser, and changing the templates for the functions in such a project, see the Help available in the Rapid Language Support Tool.

### Nested functions

Functions can be nested within other functions to form complex expressions.

### Examples

To retrieve a minimum value from a given set of numbers (ie more than two), you can nest the [Minimum function](#) multiple times to accommodate your set of numbers. For example:

- `Minimum(x,Minimum(y,z))`

NOTE: If your set of numbers are instances in an entity, then use `InstanceMinimum` or `InstanceMinimumIf` [entity functions](#).

To write a rule where a money value is always rounded up the nearest dollar, you would use nested functions as follows:

**the total benefit paid in whole dollars = (((the total benefit paid truncated to 2 decimal places)+ 0.99) truncated to 0 decimal places)**

The rounding up in this rule is achieved by:

- truncating the value of {the total benefit paid} to 2 decimal places: **trunc (the total benefit paid, 2);**
- adding 0.99 to the result: **<result1> + 0.99;** and
- truncating the new figure to zero decimal places: **trunc (<result2>,0)**

If the total benefit paid is 180.7569 (as a result of previous calculations in the rulebase), then the first step truncates this 180.75. The second step adds 0.99 to 180.75 giving a figure of 181.74. The third step truncates this to 181.

See also:

- [Function syntax references for US English](#)
- [Function syntax references for other languages](#)

## Add rule metadata

The following rule metadata can be added to a rule:

- Rule name - specifies the name that will be used for the rule in the list of generated rules in Oracle Policy Modeling
- Rule source - specifies the origin of the rule, such as the legislative provision, policy document reference or instruction manual reference
- Rule definition - specifies the purpose, meaning or behaviour of the rule
- Rule start date - specifies the date that the rule applies from
- Rule end date - specifies the date that the rule ceases to apply
- Rule fragment indicator and priority - specifies that the rule is [one of several which prove a single conclusion attribute](#), and the priority of the rule within this group of rule fragments.
- Rule loop indicator - specifies that the rule is part of an [intended rule loop](#)

NOTE: Only the last two rule properties affect how the rule operates, the others are for documentation purposes only.

To add or amend rule metadata:

1. In your Word rules document, place your cursor anywhere in the relevant rule and click the **Rule Properties Editor** button on the Oracle Policy Modeling toolbar.

2. In the **Rule Properties** dialog box, enter your rule metadata in the appropriate fields.

The image shows a 'Rule Properties' dialog box with the following fields and options:

- Rule Name: [Text Input]
- First Class Properties:
  - Synchronization ID: [Text Input]
  - Rule Source: [Text Input]
  - Rule Definition: [Text Area]
  - Rule Start Date: [Text Input]
  - Rule End Date: [Text Input]
- Rule Fragment:  [Text Input] (priority)
- Rule Loop:
- Custom Properties: [Text Area]

Buttons: OK, Cancel

See also:

- [Prove an attribute using multiple rules](#)
- [Model loops in rule logic](#)

## Validate user input using errors and warnings

Validation of a rulebase is done in two ways: using error and warning events, and by specifying validations on user input. Validation will warn or prevent the user from entering values which do not meet certain criteria when running the rulebase.

Error and warning events are types of rules that specify an action to be taken in a process outside of the rulebase. They operate in a similar way to normal rules, except that instead of inferring an attribute, they execute a command (ie firing the command specified in the conclusion line of the event rule). Event rules participate in [inferencing](#) only – not in the question search.

Oracle Policy Modeling also allows you to specify validations on the user input at runtime. These validations are set using minimum and maximum values and regular expressions on variable attributes. These input validations are triggered at the point which the value is submitted to the Engine and not during inferencing.

## What do you want to do?

[Write an error event rule](#)

[Write a warning event rule](#)

[Specify minimum and maximum values](#)

[Use regular expressions](#)

### Write an error event rule

An error event is used to pass a message to the user, and prevent them from continuing an investigation until the condition which triggered that error no longer applies.

To write an error event rule use the following syntax for the conclusion line of the rule:

- `error("<error message text>") if`

For example,

```
Error("You can only be married to one person.") if  
the applicant's number of spouses > 1
```

### Write a warning event rule

A warning event is used to pass a message to the user, but permits them to continue despite the condition which triggered that warning.

To write a warning event rule use the following syntax for the conclusion line of the rule:

- `warning("<warning message text>") if`

For example,

```
warning("The date of birth you have entered is in the future.") if  
the person's date of birth > the current date
```

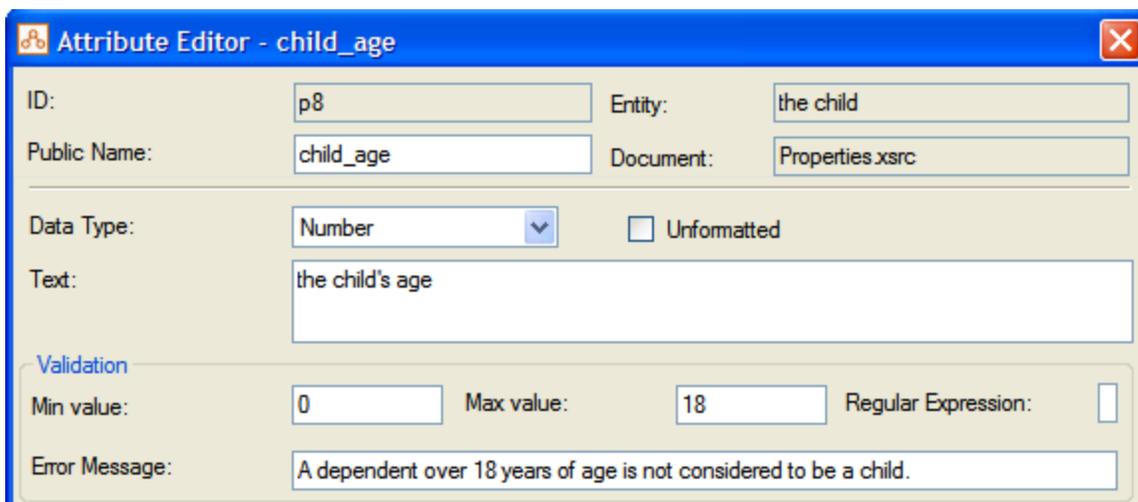
### Specify minimum and maximum values

Minimum and maximum values can be specified for number, currency, date, time of day, and date/time variables to ensure that data entered by the user falls within a certain range. Values must be specified in the correct format.

- For numbers and currency, this is the lowest and highest number you wish to allow users to enter.
- For dates, this is the earliest and latest date you wish to allow users to enter. Dates must be in the format yyyy-MM-dd.
- For time of day variables, this is the earliest and latest time you wish to allow users to enter. Times must be in the format hh:mm:ss.
- For variables of type date and time, this is the earliest and latest date and time you wish to allow users to enter. The date/-time values must be in the format yyyy-MM-dd hh:mm:ss.

To specify minimum and maximum values for an attribute:

1. Open the properties file for your project and double-click on the attribute in the Attribute view to open it in the **Attribute Editor**.
2. Enter the minimum value in the **Min value** text field.
3. Enter the maximum value in the **Max value** text field.
4. Enter a message in the **Error Message** text field, to be displayed to the user when the validation is triggered, or leave the default error message (ie "Invalid Value").



5. Click **OK**.

### Use regular expressions

Regular expressions can be used in Oracle Policy Modeling to ensure that the data entered by the user matches an expected format. You can also specify a custom error message to display when the user enters data that does not conform to the format. An example of where you might use a regular expression would be to check that a driver's license number contains the correct number of digits and / or alphabetical characters in the correct relative positions.

Note: Regular expressions are only recommended for use with text variable attributes. They are *not* recommended for use with number, currency, time of day, or date and time variable attributes. It is currently possible to specify a regular expression for a non-text variable attribute in Oracle Policy Modeling, but this functionality will be removed in a future release. In any case, regular expressions that include spaces and brackets can only be used on text variables (spaces and brackets are not valid input characters for other variable types).

Common regular expressions are given below:

Use	Regular Expression
To check basic types of email addresses	<code>^[w-]+(?:\.[w-]+)*@(?:[w-]+\.)+[a-zA-Z]{2,7}\$</code>
To check the data entered follows a certain format: 2 letters 6 numbers 1 letter (eg AB123456C)	<code>^[A-Za-z]{2}[0-9]{6}[A-Za-z]\$</code>
To check the data is a phone number in the format: (NN) NNNN NNNN where N = number. This example will match with or without the 2	<code>^\(\(0[0-9]\)\) ?[1-9][0-9]{3} ?[0-9]{4}\$</code>

Use	Regular Expression
spaces separating the sections.  To validate that the number of words in a text input does not exceed a certain number. In this example, the value N is equal to the number of permitted words minus one, so if the permissible number of words is 8 then, N = 7.	$\wedge\s*(\s+(\s+\s+){0,N}\s*)?\$$
To check a National Insurance Number (NINO)	$\wedge[A-CEGHJ-PR-TWZ][A-CEGHJ-NPR-TWZ]?\d{2}?\d{2}?\d{2}?[ABCD]$

TIP: There are a number of internet resources available on crafting regular expressions(eg <http://en.wikipedia.org/wiki/Regex>).

To specify a regular expression for an attribute:

1. Open the properties file for your project and double-click on the attribute in the Attribute view to open it in the **Attribute Editor**.
2. Enter the regular expression in the **Regular expression:** text field.
3. Enter a message in the **Error message:** text field, to be displayed to the user when the validation is triggered, or leave the default error message ("Invalid Value").
4. Click **OK**.

## Use rules to trigger external software applications

A custom event rule is used to allow rulebase events to trigger an external software application. This feature is designed to increase the flexibility of the Oracle Policy Modeling product set by enabling integration with inference events.

To write a custom event rule use the following syntax for the conclusion line of the rule:

- raiseevent <custom event>( <attribute id>, <attribute id>... ) if

For example, if you want to display some events to the user, and log other events at the backend, then a custom event can be useful to provide this additional categorisation. In this instance you could create custom events called DisplayError and LogError.

To test a custom event rule, you can use the debugger to check if the conditions for the rule were met. Other problems with custom events (eg problems with the event handler which mean that the rule doesn't do what you expect it to) will need to be diagnosed by a technical person.

## Designing and maintaining rule documents

Topics in "Designing and maintaining rule documents"

- Identify what rules are needed
- Model the structure of legislation
- Split a rule across documents
- Improve the wording of a rule
- Split and link rules
- Model discretion within rules

### Identify what rules are needed

Conceptually, rules perform three types of discrete roles:

1. **Source rules** model core source material. Typically, these are closest in structure and wording to the underlying rules which are being represented. In the case of legislation-based rulebases, these are the core legislative rules.
2. **Business rules** model policy and interpretative implementations of the underlying source material.
3. **System rules** provide an integration function, linking source rules and user input. For example, when developing models for use with Oracle Web Determinations, rules are needed to support screen flows and document generation, as well as to tie together discrete source elements and to work with screen controls.

These distinctions are simply conceptual – in fact there is no difference between the rules in syntax or operation. They are a convenient way of thinking about different types of rule operation, and help you to organize your rules for easiest maintenance.

TIP: The physical structure of the rulebase, that is the location of the rules in the rulebase, should reflect the actual structure of the underlying source rules, while accommodating the conceptual division described above.

### Identifying source rules

Once the scope of the rulebase has been determined and documented, you need to create rules documents for the sections of the material that are in scope. One of the primary activities in interpreting source material is to identify the logical foundation blocks in that material – the logical operators and the conditions. Identifying conditions first helps you to break down your rules and further identify logical operators between them.

These source rules need to be transformed into Oracle Policy Modeling format. During this process, the rule text may need to be changed in order to explicitly model relationships between the rules and to adequately handle entities. However, as a general rule, the structure and semantics of the source material should be retained as fully as possible.

### Show me an example

The following example shows how conditions can be extracted from source material.

Source material:

#### **Australian Government Assistance for Areas Affected by Cyclones Monica and Larry Fact Sheet**

##### **Business Assistance Fund**

Who can get it?

- Registered businesses, including farmers, in areas affected by Cyclones Monica and Larry.

To qualify, businesses must:

- have a registered Australian Business Number/s (ABN) on or before 20 March 2006
- be located in the areas declared affected by the combined impacts of Cyclones Monica and Larry as defined in Annexure 1
- have not already claimed the Business Assistance Fund
- receive more than 50 per cent of their income from the registered business, if a sole trader or partnership (unless they are a primary producer with long lead times to production)
- have been adversely affected by the combined impacts of Cyclones Monica and Larry
- be prepared to show evidence supporting claimed losses
- make a claim by 31 August 2006
- have been solvent immediately before Cyclone Larry, and
- have been owned by the applicant/beneficiary immediately before Cyclone Monica.

#### Resulting conditions:

the registered business has an ABN on or before 20 March 2006

the registered business is located in the areas declared affected by the combined impacts of Cyclones Monica and Larry as defined in Annexure 1

the registered business has already claimed the Business Assistance Fund

the registered business receives more than 50 per cent of their income from the registered business

the registered business is a sole trader

the registered business is a partnership

the registered business is a primary producer with long lead times to production

the registered business has been adversely affected by the combined impacts of Cyclones Monica and Larry

the registered business is prepared to show evidence supporting claimed losses

the registered business has made a claim by 31 August 2006

the registered business was solvent immediately before Cyclone Larry

the registered business was owned by the applicant/beneficiary immediately before Cyclone Monica

#### Identifying business rules

Once the source rules are created, the language may still be too complex for an ordinary user. Hence it is often essential to use plain English language to express the same concepts and reuse information wherever possible. This interpretative step is captured as a separate layer to ensure that the source rules are as pure as possible and that the interpretative step is explicitly recorded in the rules.

#### Identifying system rules

A typical rulebase modeled solely on source and business rules is not suitable for use as a user-oriented software application. System rules, also known as application rules, are used to provide an additional layer between source rules and user input where the application requires it.

System rules may take a number of forms including:

- Validating user input (eg date of birth not in the future)
- Proving visibility attributes (eg the generate claim form link should be displayed if the person is eligible for the benefit)

- Proving one piece of data with another (eg the person is not pregnant if the person is male)
- Providing a level of data mapping between your rules and the data being fed into the rulebase (eg setting "the person's rank is Captain" (Boolean) from "the person's rank" (text attribute))
- Streamlining question flow (eg the customer's basic information has been collected if the customer's first name, surname, address and credit rating are known)

## Compare the rules document with the source material

You can compare the language, structure and logic of your Word rule document with the source material quickly and efficiently using the **View Side by Side** feature in Word (assuming the source material is also written in Word). For more information, see the Microsoft Word help.

## Model the structure of legislation

Modeling legislation involves firstly determining what parts of the legislation to model, and then using a combination of indentation and structural elements to model the in scope structure of the legislation. Modeling legislation is typically done using Word rule documents.

### What do you want to do?

[Use the Ignore and Commentary styles to identify parts of the legislation that won't be modeled](#)

[Use structural elements to model legislative structure](#)

[Use keywords to customize automatic structural attributes](#)

[Model conditions without structural rule elements](#)

[Use Heading styles to organize rules](#)

### Use the Ignore and Commentary styles to identify parts of the legislation that won't be modeled

The scoping phase of rulebase construction requires deciding which areas of the rules are relevant to the application and the level of detail to which each area should be modeled.

The outcome of this analysis is captured in a Scoping Document. The Scoping Document consists of a copy of the source material marked with comments and coloring to indicate areas which will and will not be included in the source rules. The following styles should be used in this phase:

- **Ignore**  
The Ignore style is used to indicate any parts of the source material that should not be modeled (ie are out of scope) for the rulebase. To format text using this style, click on the **Ignore** button on the Oracle Policy Modeling toolbar.
- **Commentary**  
The Commentary style is used to indicate any information that should be covered in commentary text. (NOTE: marking text in this style does not automatically turn the text into the actual commentary that will be linked to text.) To format text using this style, click on the **Commentary** button on the Oracle Policy Modeling toolbar.
- **Normal**  
The **Normal** style in Word is used for all parts of the source material which are to be modeled (ie are in scope) in the rulebase.

Comments and footnotes can also be added to justify each scoping decision. This assists review and rulebase maintenance.

## Use structural elements to model legislative structure

The structural elements in legislation (section, paragraph, subparagraph etc) or policy (guidance, chapter, criterion) can be captured in rules.

During compiling, Oracle Policy Modeling will automatically generate structural attributes based on the numbering system used in your rules. The default form of these automatic attributes is "section x is satisfied".

A single tab character ( → ) is used before any conclusion or condition to define structural rule elements. You cannot use the tab character anywhere in your rules except for this purpose.

### **4 → the claimant is eligible for living allowances if**

(a) → both

→ the claimant is living alone and

→ either

(i) → both

→ the claimant is aged over 65 and

→ the claimant is a man

→ or

(ii) → both

→ the claimant is aged over 65 and

→ the claimant is a woman

Compiling this rule will result in the following structural attributes being automatically generated:

section 4 is satisfied

section 4(a) is satisfied

section 4(a)(i) is satisfied

section 4(a)(ii) is satisfied

In any transformation to Oracle Policy Modeling format, the representation of structural elements should isomorphically model the structural elements from the source material. Altering the numbering conventions will make it impossible to cross-reference your rules against the original material.

It is possible to customize these automatic attributes to more accurately reflect the source material you are modeling (see below).

## Use keywords to customize automatic structural attributes

Attributes of the form "section x is satisfied" do not always provide a satisfactory reference to the source material you are modeling, so Oracle Policy Modeling provides a number of ways to customize these automatic attributes.

### **Default Structural Element**

You can specify a default structural element, such as "regulation", "ruling", or "provision" in your rules to override "section", which is the default element used if none is specified.

This needs to be written in your Word document using the following syntax:

```
Default_structural_element[Regulation ]
```

This results in automatic structural elements in the following form:

```
Regulation 1 is satisfied
```

```
Regulation 1(a) is satisfied
```

To add this, you must use the **Configuration** style above the rules which are to use the new element (click on the **Configuration** button on the Oracle Policy Modeling toolbar to set this style).

You can add multiple configuration lines in your document to customize sections of your document.

You may also wish to add the name of the instrument, or source document for greater clarity in your source rules, for example:

```
Default_structural_element[Tax Regulations 1996 regulation ]
```

NOTE: This configuration setting is "space-sensitive". If you don't add a space after the element, one will not be added.

### Default Structural Global Proof

You can specify a default structural global proof, such as "applies" or "has been met" to replace the default "is satisfied" for global structural attributes using the following syntax:

```
Default_structural_globalproof[^x applies]
```

This results in automatic structural elements in the following form:

```
Section 2 applies
```

```
Section 23(a) applies
```

### Default Structural Entity Proof

You can specify a default structural entity proof, such as "applies to" or "has been met" to replace the default "is satisfied" for entity-level structural attributes using the following syntax:

```
Default_structural_entityproof[^x applies to ^entity]
```

This results in automatic structural entities in the following form:

```
Section 2 applies to the claimant
```

```
Section 23(a) applies to the claimant
```

NOTE: This will apply to all entities in the text following the definition.

### Ignore

Your rules can be configured to ignore specific word combinations.

The following syntax will ignore propositions in the form "this paragraph is satisfied ":

```
Ignore[this paragraph is satisfied]
```

This is useful for providing placeholder text in rules, so that they make sense when read from within the Oracle Policy Modeling document, but extra attributes and rule layers are not created.

### Replace

Your rules can be configured so that certain word combinations will be replaced with automatic structural terms. This is used in conjunction with the substitution token "^x". The syntax is:

Replace[ <text to be replaced> , <replacement text including structural element ^x> ]

## Replace Entity

The Replace syntax can also be applied to entity-level attributes. This is used in conjunction with the substitution tokens "^x" and "^entity". The syntax is:

Replace[ <text to be replaced> , <replacement text including structural elements ^x and ^entity> ]

## Model conditions without structural rule elements

We often model conditions in a rule that do not reflect structural elements within the rule section that we are modeling. The following are some examples of where this occurs:

- a subsection contains three conditions in a single provision;
- a subsection contains an application condition in its preamble, followed by several qualifying paragraphs;
- a section sets out some of the criteria for satisfaction of its goal, but other sections contain additional criteria (exceptions or extensions);
- the source material does not use numbering.

Oracle Policy Modeling format deals with this by using structural elements where they are explicit, and otherwise by representing additional conditions without structural elements. The following example illustrates this:

### **38 → the company is an eligible company if**

- (a) → the company is registered in Australia and
- (b) → the company's annual turnover is less than five million dollars and
- (c) → the company is a private company and
- the company is not disqualified under section 39

This example shows a situation in which one section has three paragraphs, but another section forms an implicit additional premise, that must be added to the rule. In this case, a,b,c and the additional premise need to be true in order to prove that 38 is true.

## Use Heading styles to organize rules

Headings should be used to break your rules into discrete, manageable sections. There are three heading styles in the Oracle Policy Modeling template in Word. These can be applied by using the **Heading 1**, **Heading 2** and **Heading 3** styles in the Oracle Policy Modeling toolbar.

When Oracle Policy Modeling compiles your rules, it automatically places rules within folders and sub-folders based on your headings and their corresponding levels.

## Split a rule across documents

There are times when you might have multiple rule developers who all want to add conditions to the same rule but they don't want to (or can't) share access to the same rule document.

For example, you might have a rule which says "the advice to contact the customer helpline should be displayed if". On a large project you may have multiple rule developers all working on different topic areas but several topics advise the customer to contact the

customer helpline. If you wanted to have all rule developers editing the one rule document, each rule developer would need to wait their turn to work on it.

**Rule fragments** allow each rule developer to have the rule proven in their own rule document to work on at their leisure. The rule fragments are combined into a single rule (separated by "or"s) when the rule documents are built. NOTE: Rule fragments only work in Word, not in Excel.

For more information on how to write rule fragments, see [Prove an attribute using multiple rules](#).

## Improve the wording of a rule

The wording of rules can be improved in two ways:

1. By using a variable comparison to infer a number of separate boolean attributes, one for each possible value.
2. By replacing grouping operators (any/either, all/both) with a new attribute.

### Using variable comparisons to infer boolean attributes

Sometimes a boolean attribute will be correct but very awkward to read and answer. In this sort of situation, it is often advisable to use an interpretative rule to make the attribute more coherent. This involves creating another rule which "wraps" the lower level rule.

The following example shows a rule which uses a string comparison with a text variable "the type of pet":

**the pet is a lizard if**

the type of pet = "lizard"

This type of rule structure is commonly used to transform a variable comparison to a boolean attribute that can be reused throughout the rulebase in the source rules. The variable can be used to infer a number of separate boolean attributes, one for each possible value (for instance, the pet is a dog, the pet is a cat etc). A drop-down list can then be used in the interview to collect the value of the variable from the user.

### Replacing grouping operators with new attributes

Intermediate attributes in Oracle Policy Modeling format can be added instead of using grouping operators, as demonstrated in the following example.

Before (using grouping operators):

**the claimant is eligible for living allowances if**

the claimant is living alone and

any

all

the claimant is aged over 65 and

the claimant is a man

or

all

the claimant is aged over 60 and

the claimant is a woman

After (grouping operators replaced with new attributes):

**the claimant is eligible for living allowances if**

- the claimant is living alone and
- the claimant satisfies the age criteria
  - the claimant satisfies the male age criteria
    - the claimant is aged over 65 and
    - the claimant is a man
  - or
  - the claimant satisfies the female age criteria
    - the claimant is aged over 60 and
    - the claimant is a woman

Addition of these intermediate attributes is highly recommended to improve understanding of decision reports and to assist in debugging.

### Split and link rules

Each rule specifies logical relationships between conditions. Logical relationships can be modeled into rule networks. Large networks of rules can be built in this form. This is known as nesting or chaining rules. Lower-level rules are nested within higher-level rules.

### What do you want to do?

[Understand how rules link together](#)

[Link rules together](#)

[Split large rules into smaller rules](#)

### Understand how rules link together

In a rule network, you can have many thousands of rules working together in an interconnected way. Take for example, the following two rules:

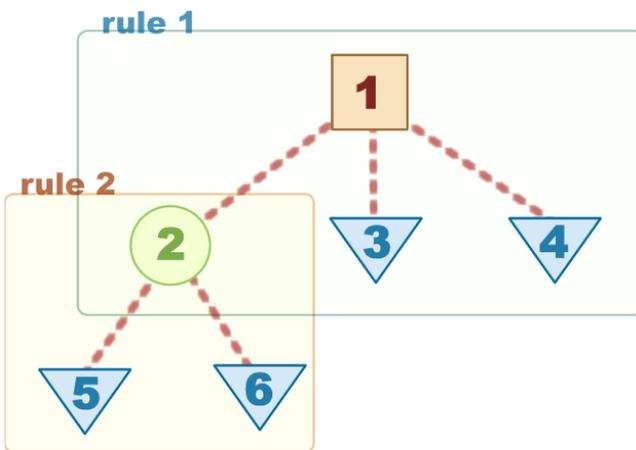
#### Rule 1

Rule Element	Comment
1: The person is eligible for a licence if:	This is the goal proposition or conclusion of the rule.
2: The person is qualified to drive a car	This is one of the rule conditions but is also the conclusion for Rule 2 (below).
and 3: The person lives in Victoria	This is one of the rule conditions.
and 4: The person is over the required age limit	This is one of the rule conditions.

## Rule 2

Rule Element	Comment
2: The person is qualified to drive a car if:	This is the goal proposition or conclusion of the rule. Rule 1 also uses this as a condition.
5: The person has passed the driving examination	This is one of the rule conditions.
and 6: The person has passed the medical test	This is one of the rule conditions.

In the example above, Rule 2 is nested within Rule 1:



Condition 2 is one of the conditions which proves the conclusion of Rule 1. However, Condition 2 is also the conclusion for Rule 2, which is further proved by Conditions 5 and 6. So, part of Rule 1 is proved by Rule 2.

### Link rules together

The Oracle Policy Modeling compiler will recognize that a particular attribute has been used multiple times, provided the same text is used each time. You can therefore link rules within or between documents in a project simply by using the text of a condition in one rule as the conclusion of another rule.

A basic example of linked rules is:

**[b1] the person is eligible for a home loan if**

[b2] the person is employed

**[b2] the person is employed if**

[p1>0] the person's weekly income > 0

In this example, the Oracle Policy Modeling compiler recognizes that the text "the person is employed" is exactly the same in both rules, and therefore labels the attribute the same accordingly.

Linking can also occur between rules where either the positive or the negative form of an attribute is used. To do so, you should use the exact text of the negation (you can check this in the **Attribute Editor** in Word or Excel by clicking on the **Edit** button for the attribute). For example, the text "the person is not employed" will be recognized as the negative form of attribute "the person is employed":

**[b3] the person is eligible for rent assistance if**  
    **[not b2]** the person is not employed

The easiest way to ensure you are using the same text of an attribute is to use the copy-paste function in Word or Excel, or to 'drag and drop' the text of the attribute from the **Data Model Browser** which is accessed from the Oracle Policy Modeling toolbar in Word.

### Split large rules into smaller rules

It is theoretically possible, using Oracle Policy Modeling format, to model extremely complex logic, with further and further indentations correctly separating groupings of conditions. That said, you should not go lower than 5 levels in a single rule because highly complex rule structures will become difficult to maintain or understand.

You can avoid large, highly nested rules by breaking the rule into smaller rules (ie by moving some of the structure of the rule to a new top level rule).

For example, the following rule:

**the claimant is eligible for the pension if**  
    the claimant is a man and  
    the claimant satisfies the age criteria  
        the claimant is aged over 65 or  
        both  
            the claimant is blind and  
            the claimant is aged over 40

can be broken into two rules:

**the claimant is eligible for the pension if**  
    the claimant is a man and  
    the claimant satisfies the age criteria

**the claimant satisfies the age criteria if**  
    the claimant is aged over 65 or  
    both  
        the claimant is blind and  
        the claimant is aged over 40

### Model discretion within rules

A discretionary decision is one that relies on the wisdom and experience of the user. It can also be regarded as a question of informed opinion rather than fact.

There are three main approaches to handling discretions in the rulebase (NOTE: References to a "decision maker" should be read as a reference to the user):

### 1. Direct approach

The discretion to be exercised is asked as a base level attribute. The user exercises the discretion based on help text guidance. A free text reason box is added where appropriate to collect audit information. This approach is appropriate where questions of fact and value are inseparable in the exercise of discretion so that a decision maker attaches the value to a matter of fact even in choosing to have regard to that matter of fact.

In such cases, exercise of the discretion as a base level attribute presupposes appropriate data collection by the decision maker. The help text relating to the base level attribute will need to specify what entitlement data is more/most relevant to the exercise of the discretion, and suggest values to be attached to particular entitlement data.

### 2. Recommendation approach

The rulebase collects data related to the discretion and then presents a recommendation to the user. The user is asked to confirm or override the discretion and to fill in the free text reason box for audit purposes. This approach is appropriate where questions of fact and value are separable, but must be reconciled by a decision maker in the exercise of the discretion. For example, the form in which a discretionary provision appears may establish the relevant matters of fact, but the decision maker is required to attach a value to each matter of fact before exercising the discretion.

The rule structure must ensure that the recommendation is known before the discretionary decision is required.

For example:

**the child is a good child if**

it is known whether or not the child has a clean room and  
it is known whether or not the child has gone to bed on time and  
the decision maker is of the opinion that the child is a good child

**the system recommends that the child is a good child if**

the child has a clean room and  
the child has gone to bed on time

In such cases, exercise of the discretion as a base level attribute relates to the reconciliation of questions of fact and value by a decision maker. The help text relating to the base level attribute will need to suggest values to be attached to particular entitlement data in the exercise of the discretion.

### 3. Guided approach

This approach has two optional paths - the user can exercise the discretion immediately as a base attribute (ie approach 1), or can choose to be guided through the various considerations that must be made in exercising the discretion. A guided data collection process is used to ensure that the user has considered the appropriate factors for exercising the discretion, and that the factors can be reviewed for audit purposes. The user is then presented with a question as in approach 1 that requires the user to enter the discretion as a base level attribute, with a free text reason box. This approach is appropriate where questions of fact and value are entirely separate, in that the only question of value in the exercise of discretion is that it be exercised at all. For example, the form in which a discretionary provision appears sets out the matters of fact which must be considered if the discretion is to be exercised.

In such cases, exercise of the discretion as a base level attribute by a user would relate to whether it is appropriate for the discretion to be exercised following a consideration of the material facts. The help text relating to the base level attribute would need to specify the precise situations (if any) in which it would be inappropriate to exercise the discretion given the material facts.

The default position is to use the direct approach. The recommendation approach should be used in the limited situations where it is possible. The guided approach should only be used where neither of the first two approaches is appropriate. All discretions need significant [help text](#) support.

# Languages

## Topics in "Languages"

- [Write rules in other languages](#)
- [Create a new language translation for a rulebase](#)
- [Localize interview help](#)
- [Localize interview document templates](#)
- [Select the user interface for rule authoring](#)
- [Configure the list of recognized verbs](#)
- [Format a numeric constant for the correct region](#)
- [Language specific considerations](#)

## Write rules in other languages

Oracle Policy Modeling supports rule authoring in any language. The rule language and region are set for a rulebase, defining the language parser used to write rules, and the formatting used for date, number and currency values.

### What do you want to do?

[Specify the rule language](#)

[Specify the rulebase region](#)

[Change the rule language or region](#)

[View the function syntax for the rule language](#)

[See which version of a language parser a rulebase is using](#)

### Specify the rule language

The rule language determines what language documents are parsed in. It is also used to decide what language rule table text should be added in.

You specify the rule language for a project when you [create a new project](#). In the **New Project** dialog there is a drop-down list that contains a list of **Rule Languages** for you to select from. This list reflects the [language parsers](#) installed with Oracle Policy Modeling. The default rule language is English (American), or the last rule language you worked with in Oracle Policy Modeling previously. Once you have created your project and commenced rulebase development (ie once rules or attributes have been created), the rule language is locked and you cannot change it.

### Creating a new language parser

If the language that you want to create your project in is not listed in the Rule Language list, you can use the **Oracle Policy Modeling Rapid Language Support Tool** to create a language parser for that language. (The Oracle Policy Modeling Language Support Tool is available from **Start | All Programs | Oracle Policy Modeling | Oracle Policy Modeling Tools | Oracle Policy Modeling Rapid Language Support Tool**. Help on using that tool is available from the Help menu in the tool itself.)

Once you have created a new language parser using this tool, when you reopen Oracle Policy Modeling and create a new project, the parser you created will appear in the Rule Language drop-down list.

## Syntactic and non-syntactic parsers

Syntactic parsers are those that include a configurable list of recognized verbs. This means that attributes can be entered in rules using the positive, negative or uncertain form and the parser will generate the other forms correctly. Syntactic parsers are those in the Rule Languages list (in the New Project dialog) that do not have "(RLS)" after the language name.

Non-syntactic parsers do not have a built-in verb list and so the sentence parses are generated using a generic sentence form defined in the configuration for that particular RLS parser. These parsers are shown in the Rule Languages list with "(RLS)" after the language name , for example "Thai (RLS)".

## Specify the rulebase region

The rulebase region determines how numbers, dates and currency values are formatted. This is used to interpret any constant values used within your rules, eg income limits, dates of effect, etc.

You specify the region for a project when you [create a new project](#), by selecting from the **Region** drop-down list in the **New Project** dialog. The default region shown for this list is based on what Oracle Policy Modeling detects as your current system locale.

The region setting also controls the default formatting applied when your rulebase is deployed, eg whether a date value entered by a user is interpreted in dd-mm-yyyy or mm-dd-yyyy format. Note that you may customize the deployment settings so they are not based on this project setting - please see the [Oracle Policy Automation Developer's Guide](#) for details.

Once you have created your project and commenced rulebase development (ie once rules or attributes have been created), the rulebase region is locked and you cannot change it.

## Change the rule language or region

If no rules or attributes have been added to the project, you can change the rule language or region by:

1. Go to **File | Project Properties | Common Properties | General**.
2. Click the browse button next to **Rule Language** to open the **Language Selector**. Select a different rule language, then click **OK**.
3. Click the browse button next to **Region** to open the **Region Selector**. Select a different region, then click **OK**.

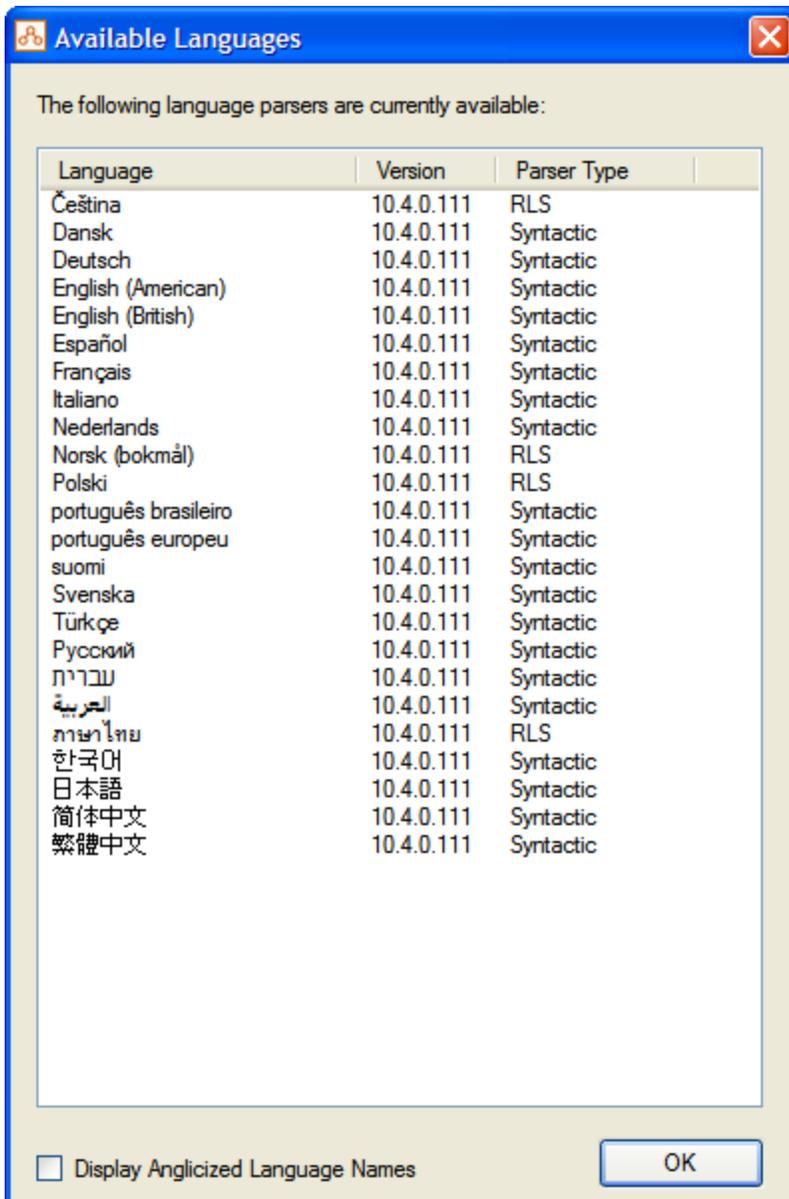
Once you have created your project and commenced rulebase development (ie once rules or attributes have been created), the rule language and region are locked and you will not be able to change these settings.

## View the function syntax for the rule language

For the languages that Oracle Policy Modeling has in-built parsers for, the syntax for the functions in the chosen rule language is available at **Help | Function Reference**. To view the localized versions of the function reference, go to the topic [Localized function references](#).

## See which version of a language parser a rulebase is using

The version numbers for each of the parsers is shown at **Help | Available Languages**.



The **Display Anglicized Language Names** checkbox is used to display the English names for each language rather than the localized name. (This setting is not saved, so it will always be un-checked when the dialog is first shown.)

See also:

- [Create a new language translation for a rulebase](#)
- [Select the user interface language for rule authoring](#)
- [Localized function references](#)
- [Language specific considerations](#)

## Create a new language translation for a rulebase

You can translate an existing rulebase into another language by adding translation documents to the rulebase. Translation documents are Excel files in which you specify your own translations of the relevant elements of the rulebase, in any language you choose. The translation document is created for you by Oracle Policy Modeling, including all rulebase elements which need a translation in order to be deployed (ie attributes, screens, rulebase messages, events, and general rulebase metadata). Note that any element can be flagged as not requiring translation.

Translation documents are useful when you do not have access to a parser for a language, and you do not wish to create one using the Oracle Policy Modeling Rapid Language Support (RLS) Tool. Because you only create translations for the specific phrases used in your rulebase, it can be quicker to deploy a rulebase using this method than by creating a new RLS parser.

Translation documents also allow you to deploy a single rulebase in multiple languages, while continuing to maintain your rulebase in its original language.

Note that creating and running a translation document allows you to provide a language translation of your rulebase only, it will not modify the formatting of data such as currency, date and number conventions etc. These formats are set based on the Region setting in Project Properties for your original rulebase.

### What do you want to do?

[Add a translation of an existing rulebase](#)

[Run a translation of a rulebase](#)

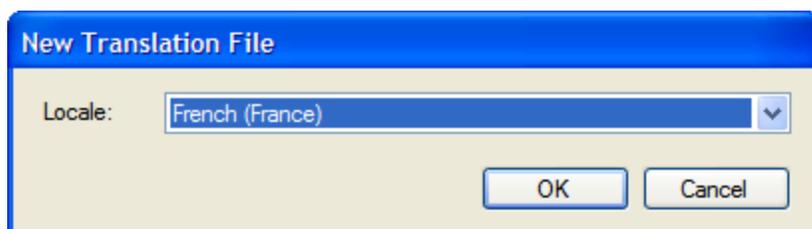
[Check for untranslated text in a rulebase](#)

[Update a translation file](#)

### Add a translation of an existing rulebase

You can easily add a rulebase translation to an existing rulebase project. It is advisable to add a translation of a rulebase only after the bulk of rulebase development is complete. This will minimize the amount of rework needed in the translation if the main rulebase changes after the translation is done.

1. Right-click on the **Translations** folder in the Project Explorer of your rulebase project, and select **Add New Translation Document**.
2. In the **New Translation File** dialog, select the appropriate **Locale** for your new rulebase translation, based on the relevant language and region, and click **OK**. Note that the rulebase cannot have a translation for the [rule language of the rulebase](#), or two translations for the same locale setting - a build error will result if these files are included in the rulebase build.

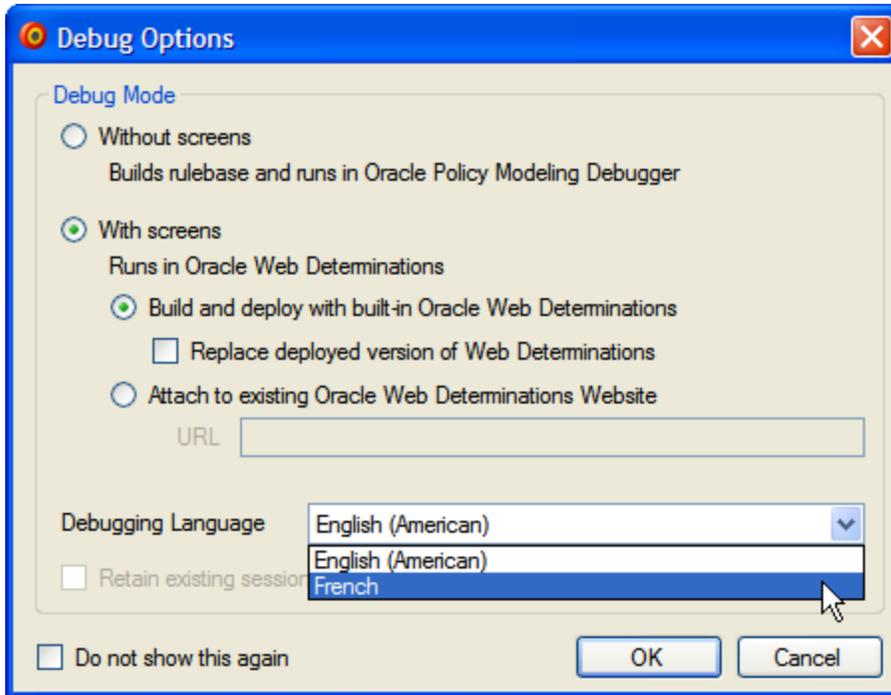


3. The new translation document is added to the Project Explorer. Enter an appropriate name for the translation document.
4. Open the new translation document in Excel. Oracle Policy Modeling has automatically inserted all the elements of the rulebase which require a translation to your chosen language, which may include:

- Statements (3rd person): unformatted text, positive, negative, question and uncertain forms are included
  - Statements (2nd person): positive, negative, question and uncertain forms are included
  - Variables (3rd person): unformatted text, translated text, positive, question, uncertain and unknown forms are included
  - Variables (2nd person): translated text, positive, question, uncertain and unknown forms are included
  - Screens: text for all relevant screen items is included (attribute free-form text, flow captions, goal captions, label captions, screen order data review title, screen name, add/remove entity instance text)
  - Metadata: commonly used rulebase text such as true/false/uncertain/unknown and gender labels are included
  - Events: text displayed to the user in Error and Warning rulebase events is included
  - Messages: text used in attribute validation messages is included
5. Fill out the translation document, completing the translations for each of the different items and their forms for your chosen language. The first column in each worksheet lists the rulebase item in the original rule language. This column is protected and must be left unaltered, to enable Oracle Policy Modeling to map the translations you supply to the original rulebase element.

NOTES:

1. If the translation of an item is intended to be the same as in the original rule language (ie it is language-independent), click on the **Ignore Translation** button on the Oracle Policy Modeling toolbar. This will flag the item as not requiring translation and it will therefore be excluded from the [Untranslated Text report](#).
  2. If your rulebase contains attributes using [second person substitution](#), additional tabs for statements and variables will be created containing the second person substitution text for these attributes. In addition to columns for the Statement and Variable elements listed above, the second column contains the second person substitution attribute in the original rule language, which must also be left unaltered.
  3. If your rulebase contains [variable substitution](#), use the substitution format "%<attribute public name>?%" (see [Substitute an attribute value into the text on screens](#) for other options using this syntax) to insert the desired attribute values into the translated text. The basic form of the 3rd person statement and variable entered in the Unformatted Text column will be used for substitution when the substituting attribute has an unknown value.
  4. If your attribute translations include [gender pronoun substitution](#), as shown in the example above, ensure that you provide the three values required by Oracle Policy Automation (corresponding to male, female and neutral genders), even if the same values are used.
6. When you have provided translations for all rulebase elements (or marked them as to be ignored for translation), compile the document using the Compile button in the Oracle Policy Modeling toolbar.  
TIP: You may also wish to translate any rulebase commentary you are using. See [Localize interview help](#) for details.
7. You can now [run the translation of the rulebase](#), or alternatively test it in the debugger by going to **Build | Build and Debug** in Oracle Policy Modeling, then select your chosen language as the **Debugging Language** in the **Debug Options** dialog. This drop-down list shows the main rule language, plus any other languages for which translations have been created.



NOTE: If you debug with screens and select a Debugging Language for which Oracle Policy Modeling does not provide built-in support (see [Localized function references](#) for supported languages), you will need to [set up the locale in Oracle Web Determinations](#).

8. The rulebase will now run in the debugger, using your translations for rulebase questions etc, and built-in translations for the screen text of Oracle Web Determinations.

## ORACLE Web Determinations

[Synthèse](#) | [Vérification des données](#) | [Enregistrer](#) | [Enregistrer sous](#) | [Charger](#) | [Redémarrer](#) | [Fermer](#)

Base de règles : SimpleBenefits Paramètres régionaux : fr-FR ID d'utilisateur : guest

### Revenu de revendicateur

Quel est le revenu annuel du revendicateur?

\*

TIP: You can also preview the translation for individual question screens in Oracle Web Determinations without running a full interview, using the [Preview](#) option and selecting your new translation in the debug options.

#### Run a translation of a rulebase

Once you have created one or more translations for a rulebase as detailed above, you can then run the rulebase in Oracle Web Determinations to access a fully-translated interview for your rules.

1. Open your rulebase with translations in Oracle Policy Modeling, and select **Build | Build and Run**.
2. If you select to run your rulebase in Oracle Web Determinations, you will see a browser screen showing you the locales of the available language options in which the rulebase can be run. The options available include the main rule language, plus any other languages for which translations of the rulebase have been created. Select the translation you wish to run. TIP: The text in the list of locales is configurable. For more information, see [Change the locale list in Oracle Web Determinations](#).
3. The rulebase interview commences, in your selected translation.

## ORACLE Web Determinations

[Vérification des données](#)

[Enregistrer](#)

[Enregistrer sous](#)

[Charger](#)

[Redémarrer](#)

[Fermer](#)

Base de règles : SimpleBenefits Paramètres régionaux : fr-FR ID d'utilisateur : guest

### écran sommaire

- [Est-ce que revendicateur admissible à l'allocation enfant adolescent?](#)
- [Quel est le paiement de rente de revenu bas du revendicateur \(par mois\)?](#)

### Set up a new locale in Oracle Web Determinations

If you run your rulebase in Oracle Web Determinations and select a translation locale for which Oracle Policy Modeling does not provide built-in support (see [Localized function references](#) for supported languages), you may see an error message if the locale properties have not yet been configured.

You will need to create a configuration file to run the translation you selected in Oracle Web Determinations. To do this:

1. In Windows Explorer, browse to the **Release** folder in which your Oracle Web Determinations is running, and go to **\web-determinations\WEB-INF\classes\configuration**
2. Locate the **messages.<locale>.properties** file which corresponds to the original rulebase language for your rulebase. For example, *messages.en.properties* for a rulebase written in US English.
3. Make a copy of this file, and rename it with the appropriate locale text for your translation. For example, *messages.lv-LV.properties* for a Latvian translation. (The required file name is also shown in the Oracle Web Determinations error message, as shown above).
4. You may optionally open the new file in a text editor and enter translations for the various configuration items under the heading **localised text for input controls**. See the [Oracle Policy Automation Developer's Guide](#) for more information on the settings in this file.

### Check for untranslated text in a rulebase

Once you have added translations to your rulebase, Oracle Policy Modeling will automatically detect whether any elements of the rulebase that require translation have not yet been translated. Warnings are generated at build time if this is the case, and you can also run a report that will list all untranslated text in the rulebase.

1. To view this report, go to **Reports | Untranslated Text**.
2. The **Untranslated Text** report is shown, listing all relevant rulebase elements for which a translation has not yet been supplied. This report will not show any items that have been marked as 'Ignore Translation' in the translation document. Items are grouped together, eg screen text, metadata, attributes with all forms that are missing translations.

If multiple translations have been added to the rulebase, items are grouped within the separate translation files.

**Untranslated Text**

Save Regenerate

Untranslated Text  
Generated 8/08/2011 1:36:15 PM

**New Translation Workbook (fr-FR).xls**

**French**

**Message:** A dependent over 18 years of age is not considered to be a child.

**Screen Text:** Question Screens  
**Screen Text:** Child Details  
**Screen Text:** The Children  
**Screen Text:** Add child  
**Screen Text:** Claimant Housing  
**Screen Text:** Data Review

**the claimant is eligible for low income allowance**

**Third Person:**

**Positive:** The claimant is eligible for low income allowance.  
**Negative:** The claimant is not eligible for low income allowance.  
**Question:** Is the claimant eligible for low income allowance?  
**Uncertain:** The claimant might be eligible for low income allowance.

**the claimant is eligible for the teenage child allowance**

**Third Person:**

**Positive:** The claimant is eligible for the teenage child allowance.  
**Negative:** The claimant is not eligible for the teenage child allowance.  
**Uncertain:** The claimant might be eligible for the teenage child allowance.

3. Use this report to complete any missing translations, then click **Regenerate** at the top of the report to verify that all translations have been completed.
4. The **Error List** will also show a warning if your rulebase has missing translations. Click on **View | Error List** to display this. Double-click on the warning message to open the Untranslated Text report.

### Update a translation file

It is advisable to add a translation of a rulebase only after the bulk of rulebase development is complete. This will minimize the amount of rework needed in the translation if the main rulebase changes after the translation is done. However, Oracle Policy Modeling will automatically detect if any new items have been added to the rulebase that aren't reflected in the translation, and prompt you to ensure these are handled effectively.

To update a translation file:

1. First [check for any missing translations](#) as detailed above, to determine whether updates to the translation file are required.
2. Open the translation document by double-clicking it in the Project Explorer. When the document opens, Oracle Policy Modeling will automatically insert any missing rulebase elements which require a translation.

	A	B	C	D	E	F
1	<b>Original Text</b>	<b>Translated Text</b>				
2	Question Screens	Question Screens				
3	Assessment Summary	Assessment Summary				
4	Claimant Income	Claimant Income				
5	Data Review	Data Review				
6						
7						

Navigation tabs: Screens, Statements (3rd Person), Variables (3rd Person), Metadata

3. [Enter the translations](#) for the newly inserted items (or mark the items as 'Ignore Translation'), using the process detailed above, and compile the translation document.
4. Re-check for any missing translations using the **Untranslated Text** report or **Error List**, and debug or [run your rule-base translation](#) to test your changes.

**NOTES:**

- i. Oracle Policy Modeling will not remove content from your translation file, so if you change or remove items from the rulebase, you will need to manually check your translation file to make any necessary updates. If an item in the rulebase is modified, Oracle Policy Modeling will insert the updated item into the translation file, however if you wish to remove or re-use the old translation instead, you must manually make these changes.
- ii. If substitution is enabled for any attributes after the rulebase has been translated, the existing translations will need to be manually updated. The quickest way to do this would be to add a new translation file and manually merge the changes (ie replace the statements for the affected attributes in the old file with the correct forms from the new file).

**See also:**

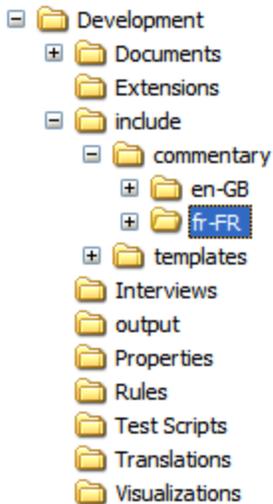
- [Write rules in other languages](#)
- [Localize a rulebase](#)

**Localize interview help**

When you [add a language translation to your rulebase](#), you may also wish to create a translated version of your rulebase commentary, which provides interview help. If no localized commentary is provided, commentary will not be displayed when you run your translated rulebase. To create localized commentary:

1. In Windows Explorer, browse to the *Development\include\commentary* folder for your rulebase. Within this location, a folder named for the original rulebase language (eg "en-GB") contains the default commentary for the rulebase. This folder is created when you first [create the commentary for your rulebase](#) (ie for the original rulebase language).

2. Create a copy of the default commentary folder, and rename it with the code for your rulebase translation language. For example, copy the "en-GB" folder and rename the copy "fr-FR" to create commentary if you have a French rulebase translation.



3. Modify each of the commentary HTML files in the new folder, to translate the commentary text as appropriate for your rulebase. Note that as for creating the original commentary files, some knowledge of html is useful, to help identify which text in the file is displayed to the user.
4. Build the rulebase, and debug or run in the translation language to view the new localized commentary files.

NOTE: Commentary should only be translated once it has been finalized for the original rulebase language. Subsequent changes to the rulebase or its commentary files/content will require manual changes to your localized commentary files.

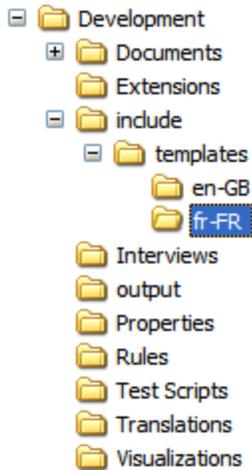
See also:

- [Create, update or delete interview help](#)
- [Create a new language translation for a rulebase](#)

## Localize interview document templates

When you [add a language translation to your rulebase](#), you should also create a translated version of any interview document templates. You need to do this for every language that your rulebase supports. To create localized document templates:

1. In Windows Explorer, browse to the *Development\include\templates* folder for your rulebase. Within this location, a folder named for the original rulebase language (eg "en-GB") contains the document templates for the rulebase. This folder is created when you first [create a new interview document](#) (ie for the original rulebase language).
2. Create a copy of the templates folder for the original rulebase language, and rename it with the code for your rulebase translation language. For example, copy the "en-GB" folder and rename the copy "fr-FR" to create the templates for a French rulebase translation.



3. Translate the text of each template in the new folder to that locale.
4. Build the rulebase, and run in Oracle Web Determinations. You will see that the set of templates used is based on the locale of the session.

NOTE: If no localized template file exists, then clicking the [document link in Web Determinations](#) will generate an error.

See also:

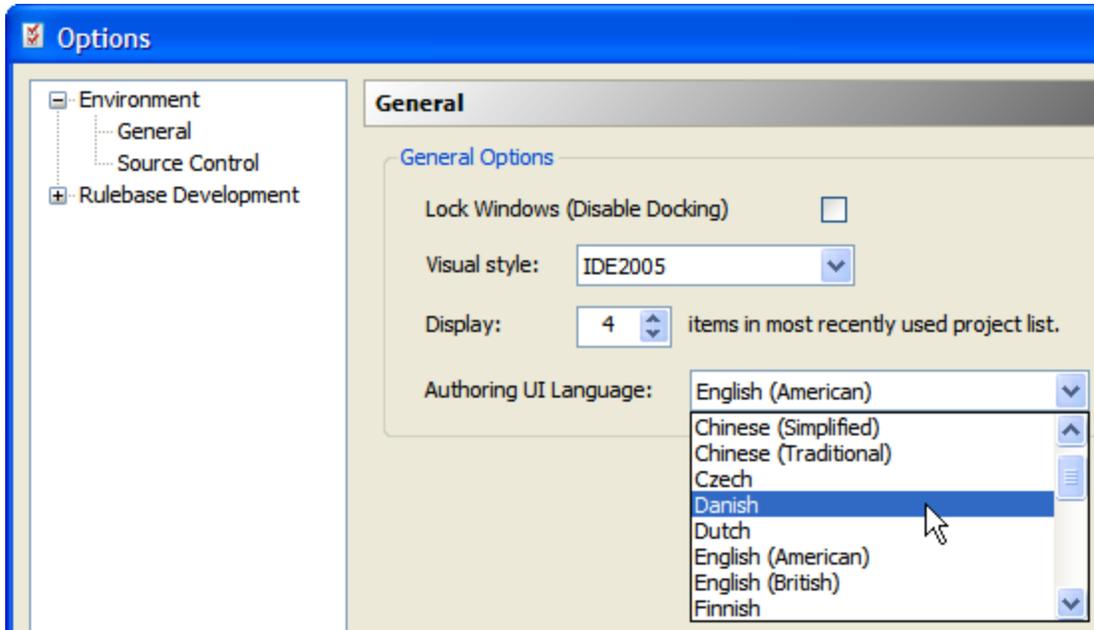
- [Create, update or delete an interview document](#)
- [Develop a template for an interview document](#)

## Select the user interface language for rule authoring

The user interface language setting controls what language is used for the user interface of all Oracle Policy Modeling components in Word and Excel, such as dialog boxes and messages. Oracle Policy Modeling supports the following user interface languages: Arabic, Chinese (Simplified), Chinese (Traditional), Czech, Danish, Dutch, English (American), English (British), Finnish, French, German, Hebrew, Italian, Japanese, Korean, Polish, Portuguese (Brazilian), Portuguese (European), Russian, Spanish, Swedish, Thai and Turkish. By default the authoring user interface language is set to English (American).

To change the authoring user interface language:

1. In Oracle Policy Modeling, go to **Tools | Options | Environment | General**.
2. Select a language from the **Authoring UI Language** drop-down list.



3. Click **OK**.

### Configure the list of recognized verbs

Oracle Policy Modeling comes with a library of commonly used verbs. These verbs have already been [conjugated](#)<sup>1</sup> and are ready to use in attributes.

Oracle Policy Modeling will be unable to initially parse an attribute if it contains an unknown verb (ie a verb that is not in the default verbs list). In this instance, it is necessary to create a custom verbs list to add the new verb to. From then on, Oracle Policy Modeling will recognize that verb whenever it is used in an attribute.

NOTE: Projects which use a Rapid Language Support language parser do not contain a list of verbs so the options below do not apply to these projects. For more information on how to edit the sentence forms for such projects, see the Help available in the Rapid Language Support Tool.

### What do you want to do?

[Create a new verb file](#)

[Add a new verb](#)

[Modify existing verb forms](#)

[Delete a verb](#)

[Delete a verbs file](#)

Create a new verb file

To create a verb file:

1. In Oracle Policy Modeling, select **File | Edit Verbs....**
2. You will be prompted to create a custom verbs file. Click **Yes**.

---

<sup>1</sup>The conjugations are the different forms the verbs take depending on what person and tense they are.

TIP: Custom verb files are created at the project level. If you already have a custom verb file from another project that you want to use in your project, copy it into the **Development** folder for your project. The file must be called verbs.xml.

### Add a new verb

After you have created a new verb file (see above), you can then add new verbs to it. In the **Verbs List** dialog box:

1. Check the verb is not in the existing **Verbs List** by entering the verb in the find field.
2. If the verb is not shown in the list, click **Add...** to add it.
3. In the **Verb Editor** dialog box, press **Tab** to automatically conjugate the verb.
4. Check that all of the verb forms are correct.
5. Click **OK** to add the verb to the verb list.

NOTE: Oracle Policy Modeling does not recognize the difference between regular and irregular verbs and initially conjugates all verbs as regular verbs. If the verb you are adding is an irregular verb, you must know what form the verb takes for each [person/tense](#)<sup>1</sup> so that you can update the verb forms in the Verb Editor appropriately.

### Modify existing verb forms

To modify an existing verb form:

1. In Oracle Policy Modeling, select **File | Edit Verbs....**
2. If you have not edited the verb list before, you will be prompted to create a custom verbs file. Click **Yes**.
3. In the **Verbs List** enter the verb you want to edit.
4. When the verb is highlighted, click **Edit....**
5. In the **Verb Editor** make the necessary changes and click **OK**. Then click **OK** in the **Verbs List**.

### Delete a verb

To delete a verb from the verbs list for the project:

1. In Oracle Policy Modeling, select **File | Edit Verbs....**
2. In the **Verbs List** enter the verb you want to delete.
3. When the verb is highlighted, click **Delete**. Then click **OK**.

### Delete a verbs file

To delete a custom verbs file that is no longer needed in a project:

1. In Oracle Policy Modeling, select the project name in the Project Explorer.
2. Right-click and select **Locate in Explorer**. A new Windows Explorer window will be opened showing the Development folder for the project.
3. Select the verbs.xml file and click **Delete**.

Note that if the project still has attributes in it that use the deleted custom verbs file, these will be detected as broken parses at build time and reported as errors.

---

<sup>1</sup>The simple present tense is used for habitual actions that are not just happening now. The simple past tense is used for actions that took place in the past and that are not taking place now. The present progressive tense is used to express current actions. The past perfect tense is used for actions that were completed before some other event.

See also:

- [Language specific considerations](#)

## Format a numeric constant for the correct region

Depending on the region settings for your rulebase, the meaning of the comma "," can be as a [decimal separator](#)<sup>1</sup> (rather than a full stop ".") or as a thousand separator. The space character " " may also be designated as a thousand separator for some regions. It is important to note that this can affect the way numbers are used in function definitions, since the parameter separator used in functions (ie to denote the different values being provided to the function) is a comma, and spaces are also used to separate the parameters visually.

Consider the following example:

**the man's first initial = substring(the man's name, 0, 1)**

When authoring for a region which uses a decimal point as the separator, this reads as 'the man's first initial is equal to a substring of the man's name from position 0 for 1 character'.

When authoring for a region which uses a comma as a decimal separator, the compiler has no way of knowing whether or not the "0," is a number which has not been entered correctly (ie missing the decimal part as in 0,15).

A similar situation can arise with the use of the comma or space character as a thousand separator, when entering larger numbers into functions.

With this in mind, when authoring function rules where ambiguity such as this could occur, numbers used in functions should be encased in further brackets. For example:

**the man's first initial = substring(the man's name, (0), (1))**

Or where the comma is used as a thousand separator:

**the product key = substring(the full product code, (2,050), (27))**

It can also help to avoid ambiguity if the space characters are used to separate each of the parameters in a function, although this is not a requirement enforced by Oracle Policy Modeling.

Note that this does not affect the use of number variables, only stated constants.

See also:

- [Use constant values in rules](#)
- [Use variables in rules](#)
- [Formatting of variable values](#)

## Language specific considerations

Topics in "Language specific considerations"

- [Write rules in Arabic](#)
- [Write rules in Finnish](#)
- [Write rules in French](#)

---

<sup>1</sup>The symbol marking the point between the whole and decimal parts of a number.

- Write rules in Hebrew
- Write rules in Italian
- Write rules in Japanese
- Write rules in Korean
- Write rules in Portuguese
- Write rules in Russian
- Write rules in Spanish
- Write rules in Turkish

## Write rules in Arabic

### Supported sentence structures

The Arabic parser supports equational (verbless) sentences of the form Subject – Object, and verbed sentences of one of the forms Verb – Subject – Object (VSO) or Subject – Verb – Object (SVO). Note that the parser only supports single-word objects in verbless and VSO sentences. If a multi-word object is required, you should rephrase the sentence as an SVO sentence.

### Supported verb forms

Arabic verbs conjugate for mood (Active/Passive), tense (Present Indicative / Present Jussive / Past), gender (Masculine/Feminine), person (1st, 2nd, 3rd) and number (Singular, Dual, Plural). Note that in many grammar books, Present is known as Imperfect and Past is known as Perfect.

For each mood, tense and gender, the verbs list contains the following person and number combinations: 2nd person singular, 3rd person singular and 3rd person plural.

When adding new verbs to the dictionary, the parser correctly auto-conjugates all regular tri-literal verbs, and irregular tri-literal Hamzated verbs. For all other kinds of verbs, you need to manually review and [edit the verb conjugations](#).

The Jussive form is used to form the negation of Past tense verbs. The Jussive form is also used to form uncertain sentences. Please note that in actual Arabic grammar, the form used for uncertain sentences would be Subjunctive. However, in Oracle Policy Modeling Jussive and Subjunctive are equivalent because these forms only differ in short vowels which Oracle Policy Modeling omits.

### Limitations

#### Vocalizations

The Arabic parser supports Modern Standard Arabic (MSA). An important feature of written MSA is that short vowels (that might otherwise be denoted using diacritical marks above/below consonant symbols) are omitted and the language users infer them from the context. In particular, this means that the default Arabic verbs list does not contain these short vowels, and the parser will only recognize verbs that are written without short vowel diacritical marks. If other versions of verbs are required, these need to be [added to the verbs list](#).

#### Hamza on Alef

There are some Arabic verbs which begin with the Alef character and may have an implicit Hamza above Alef, or below Alef (e.g. `أستخدم` and `أأكل`). The default verbs list has explicit Hamzated versions of these verbs with Hamza above Alef (e.g. `أستخدم` and `أأكل`), so the parser will only recognise these Hamzated versions in rule documents. If other implicit versions are required to be parsed, these need to be added to the verbs list.

### No demonstratives

In order for substitution to work, demonstratives should not be used before substituted attributes. For example, instead of the following sentence (literally "this the person which"):

مرت سنتان منذ أن تم طرد هذا الشخص من الخدمة المدنية

use "the person":

مرت سنتان منذ أن تم طرد الشخص من الخدمة المدنية

### Attached pronouns

The sentence generation and substitution mechanism currently does not support attached pronouns. Specifically, when substituting a pronoun in place of an attribute, the parser replaces the attribute with an unattached pronoun. This should be correct in most cases, ie when the pronoun occurs at the beginning of the sentence. However, it may cause suboptimal sentences when the pronoun occurs at the end of a sentence. In such cases, the user may override the generated sentences as described in [Customize sentence text](#).

## Write rules in Finnish

### Supported sentence structures

The Finnish parser supports two kinds of sentences: Subject – Verb – Object (SVO) and Subject – Verb – Complement (SVC).

1. In SVO sentences, the verb performs some action on the object, or has some effect on the object. For most SVO sentences, the parser produces two parse matches, and the first parse match has a case transformation for the object of a negative sentence. See [Object case transformation in SVO sentences](#) below.
2. In SVC sentences, the verb is often "to be" and denotes equality, or assignment. In these sentences, the object is usually in the nominative case.

### Supported verb forms

In Oracle Policy Modeling the verbs list contains four tenses for each verb: Present, Past, Perfect and Pluperfect. For each tense, it contains the positive and negative forms for 2nd person singular, 3rd person singular, 3rd person plural and passive.

### Limitations

#### Object case transformation in SVO sentences

In SVO sentences, the divisibility of the object determines its case. If the object is divisible, it is in partitive. Otherwise, the object is in genitive. Irrespective of the case of the object in a positive SVO sentence, the object of a negative SVO sentence is always in partitive. As a result, when transforming between positive and negative SVO sentences that have non-divisible objects, the Finnish parser changes the case of the object.

The following criteria determine when the parser changes the case of the object:

- a. Either the original sentence is negative and the object is partitive, or
- b. The original sentence is positive and the object is nominative/genitive.

In case (a), the object is transformed into genitive for the positive/question/uncertain sentences.

In case (b), the object is transformed into partitive for the negative sentence.

Note that in the case of a multiple-word object, only the case of the first word of the object is checked. Additionally, the case transformations are not guaranteed 100% correct, since in some cases the parser cannot correctly determine the partitive stem of a word.

Additionally, since in some cases this transformation is not required, for each SVO sentence the parser also supplies another parse match which keeps the object intact between positive/negative sentence forms. The transformed parse match is displayed first (default) if:

- a. Either the original sentence is negative and the object is partitive, or
- b. The original sentence is positive and the object is genitive.

Thus, if the original sentence is positive and the object is nominative, the default parse match is the untransformed one.

If neither of the supplied parse matches is appropriate, you need to override the generated sentence text as described in [Customize sentence text](#).

#### **Pronoun possessives**

In possessive phrases, there is a variation in the possessed object depending on the possessor. If the possessor is a regular noun or a person's name, then the possessed object takes a basic nominative form. For example:

**Henkilön lapsi on onnellinen** = The person's child is happy

**Jussin lapsi on onnellinen** = Jussi's child is happy

However, if the possessor is a personal pronoun (eg my, his), then the possessed object takes on an additional ending that depends on the plurality of the possessor. For example:

**Sinun lapsesi on onnellinen** = Your child is happy

The Finnish parser currently does not support this change in the possessed object. That is, second person substitution would (incorrectly) generate:

**Sinun lapsi on onnellinen** = Your child is happy

Similarly, third person substitution would (incorrectly) generate:

**Hänen lapsi on onnellinen** = His/her child is happy

In this case, the user needs to override the generated sentence text as described in [Customize sentence text](#).

## **Write rules in French**

### **Limitations**

#### **Contractions**

The French parser does not currently support creating contractions after substitution has taken place. For example, consider the following attributes that allow for substitution:

*la personne*

*la maladie*

and the following Boolean attribute, which conveys that "the person's family has a history of the disease":

*la famille de la personne a une histoire de la maladie*

Oracle Policy Modeling generates the following interrogative form for the Boolean attribute:

*Est-ce que la famille de %personne?% a une histoire de %maladie?%?*

If the person is identified as John and the disease is identified as arthritis, Oracle Policy Modeling would generate the following question:

*Est-ce que la famille de John a une histoire de arthrite?*

Unfortunately, the question should contract the "de" and "arthite" as follows:

*Est-ce que la famille de John a une histoire d'arthrite?*

Currently, Oracle Policy Modeling will not perform the contraction.

## Write rules in Hebrew

### Verbs list

The default Hebrew verbs list contains conjugations for 4142 verbs. Because the **Verbs List** dialog sorts the verbs by their shoresh (root) and because multiple verbs may share the same shoresh, you may have to look at more than one entry in the Verbs List to find the particular verb of interest to you.

In Oracle Policy Modeling, vowels (denoted by diacritical marks above/below consonant symbols) are omitted, and users infer them from the context. In particular, the default Hebrew verbs list does not contain vowel diacritical marks, and the parser will recognize only those verbs that are written without vowel diacritical marks.

### Negation

To indicate negation in present tense in a Boolean attribute, use the appropriate formal word ("אינה", "אינו", "אינך", "אינך", "אינך", "אינך", "אינך", "אינך") rather than the informal word "לא".

### Parsing

When attributes are [parsed](#) in Hebrew, Oracle Policy Modeling looks for words, such as personal pronouns, that act like verbs. This may result in multiple parses for an attribute (shown in the **Confirm New Attributes** dialog with a gray background). These will need to be checked to ensure the correct parse is chosen. For more information, see [Review the attribute parses in a rules document](#).

### Excel rule tables

In an Excel rule table, boolean conclusion cells must be either "נכון" ("true") and "לא נכון" ("not true", ie "false"). In these cells Oracle Policy Modeling will not properly interpret other Hebrew words that mean "true" and "false" such as "חיובי" and "שלילי".

## Write rules in Italian

### Supported sentence structures

The Italian parser supports Subject – Verb – Object sentences.

### Supported verb forms

Italian verbs conjugate in a significant number of tenses and several moods. The indicative mood is used for factual statements. The subjunctive mood is used for uncertain sentences, as well as some "if ... then ..." sentences.

Although gender (masculine and feminine) is present in Italian, the third person forms in most tenses are the same for both genders. Additionally, the polite second person form in Italian uses the third person verb conjugation.

The verbs list in Oracle Policy Modeling contains entries for the following tense and mood combinations:

Mood	Tense	Verb forms
Gerund	Present	One form

Mood	Tense	Verb forms
Participle	Past	Masculine singular, masculine plural, feminine singular, feminine plural
Indicative	Present	Singular, plural
Indicative	Imperfect	Singular, plural
Indicative	Past absolute	Singular, plural
Indicative	Future	Singular, plural
Subjunctive	Present	Singular, plural
Subjunctive	Imperfect	Singular, plural

The Italian parser supports both simple and compound verb constructions. For example:

- a simple verb sentence is *lo student **studia** la lezione* (the student studies the lesson)
- a compound verb sentence is *la figlia **è stata accompagnata** dal genitore* (the daughter has been accompanied by the parent)

## Limitations

### Substitutions

In order for the substitution to work correctly, every variable and entity should either be preceded by the article, or by a contracted preposition + article.

The following prepositions are supported for substitution:

1. *di* – the contracted forms *del, dello, della, dell', dei, degli* and *delle* are supported.
  - a. For example, *il libro dello studente* (the mug of the student) becomes *il libro di Marco* when Marco is substituted for *lo studente*.
2. *da* – the contracted forms *dal, dallo, dalla, dall', dai, dagli* and *dalle* are supported.
  - a. For example, *l'amico è sostenuto dallo studente* (the friend is supported by the student) becomes *l'amico è sostenuto di Luna* when Luna is substituted for *lo studente*.

The following guidelines must be followed for substitution to work correctly:

1. Each variable attribute should include the article. For example, use *lo studente* instead of *studente*.
2. Each variable used in a nominative sentence should include the article. For example, use *lo studente studia la lezione* instead of *studente studia la lezione*.
3. Each variable used in a sentence with a preposition should include the appropriate contracted preposition + article. For example, use *il libro dello studente è verde* instead of *il libro di studente è verde*.
4. There must be exactly one space between the article/preposition and the variable name. For example, use *il libro dello studente è verde* instead of *il libro dello studente è verde*.

In order for 2nd person substitution of possessives to work correctly, the object possessed may consist of one word only. For example, *il libro dello studente* (the student's book) can be correctly transformed into *il suo libro*. However, *il libro verde dello studente* (the student's green book) cannot be correctly transformed into a 2nd person sentence. It can, however, be transformed into a 3rd person sentence *il libro verde di %varid?%* where "varid" is the public name for the variable *lo studente*.

## Write rules in Japanese

### Supported sentence structures

The Japanese parser supports two kinds of sentences:

1. Verbless sentences

An example of a verbless sentence is 彼の行動は法律的に正しかった(His action was legal).

2. Subject – Object – Verb (SOV) sentences

An example of a SOV sentence is 当人は子供が5人以上いる(The person has more than five children).

### Supported verb forms

Japanese verbs are inflected for politeness level, tense, aspect, voice and sense.

The verb dictionary provides the plain (colloquial) and the polite forms of the verbs.

There are only two tenses in Japanese, past and non-past. The non-past covers both the present and the future tense.

The verb aspect denotes the conjugations for perfect, progressive and potential forms. The perfect aspect is the stative form of the verb.

The verb voice refers to whether the verb is an active or passive mode.

The verb sense indicates whether the verb inflects for a positive or a negative statement. For each of the above, the verbs are inflected by suffixing some ending based on which verb group they belong to.

The verbs do not inflect for gender or person.

The copula *だ* (*da*) which is the infinitive form of *です* (*desu*), and *である* (*dearu*) which is the infinitive form of *であります*, have been included in the verbs list.

For compound verbs where only the second verb is inflected, eg *benkyo + suru*, *suru* is taken to be the active verb. For such noun + *suru* verbs, there is no need to enter the compound verbs separately as long as *suru* is in the verbs list.

The following are the verb forms present in the verb dictionary:

- Dictionary form: the verb conjugations below are derived from the dictionary form which has to end in an *-u* such as *iku*, *kangaeru*.
- Present tense forms:
  - polite
  - plain
- Past tense forms:
  - polite
  - plain

- For each of the polite and plain forms above, the following verb conjugations are provided:
  - positive and negative
  - passive positive and passive negative
  - potential positive and potential negative
  - progressive positive and progressive negative

The automatic verb conjugations works for the majority of the *ichidan* and *godan* verbs. The conjugations for irregular verbs, and verbs where the use of kanji character introduces ambiguity as to whether the verb is *ichidan* or *godan*, will have to be entered manually. See [Configure list of recognized verbs](#) for more information.

#### Verb recognition

The active verb in a sentence is recognized based on the dictionary. When a compound verb is present, the active verb is selected based on the longest match.

For example, verbs *nakatta* (なかつた) and *kawa nakatta* (買った/買わなかつた) are both present in the dictionary. In this case, if a sentence has *kawa nakatta* as its active verb (ie the verb at the end), the parser will recognise the compound verb *kawa nakatta* instead of just *nakatta*.

In cases where the sentence uses a compound verb, where the compound verb itself has not been entered in the dictionary, the parser will try to recognize the longest match it can find. For example, if *nakatta* is in the dictionary, and the verb *shitagawa nakatta* is not in the dictionary then the parses generated for the sentence containing *shitagawa nakatta* will be based on the conjugations of the verb *nakatta*. To avoid this problem, you need to add the missing verb.

#### Adjectives

In an SOV sentence, the verb at the end is taken to be the active verb. If adjectives are present within the sentence, they are not inflected.

In a verbless sentence, the adjectives may be inflected. There are two form of Japanese adjectives, the *-na* adjectives and the *-i* adjectives.

- The *-na* adjectives are followed by some form of copula. In such sentences, the copula inflects to indicate tense, mood, aspect, etc. The adjective remains unmodified.
- The *-i* adjectives can occur on their own at the end of the sentence or they may be followed by some form of copula. If an *-i* adjective is present in a verbless sentence, then the *-i* adjective is inflected. The copula remains untouched.

In both the above scenarios and also for an SOV sentence, when the uncertain form is constructed the copula is omitted.

#### Limitations

The following verb inflections are currently not handled.

1. Presumptive mood – expresses probability, belief or intention (*~daro/~desho* forms)
2. Imperative mood – expresses commands
3. Causative mood – conveys the idea of making or causing someone to do something
4. Conditional mood – conveys 'if', 'unless', 'when' meaning (*~eba/~tara/~nara/~to* forms)
5. Clauses – conveys sequential, parallel or causal relationships (such as the *~te* and *~de* forms)
6. Necessity – expresses 'must' or 'necessity' using the *to-ikenai* form (といけない)
7. Counter words

The first three forms are unlikely to occur in the Oracle Policy Automation rulebase framework. For the fourth and fifth verb forms, Oracle Policy Automation has an existing framework for expressing conditionals and clausal relationships when developing a rule-base. As such, these verb inflections are redundant. For the sixth form, expressing 'must', the sentence should be rephrased, for example using the verb 'obligated'. The parser only supports limited number of counter words such as those for age and number of people.

For example, look at the following sentences.

**Example 1 - Conditional mood**

The person is eligible if the person pays tax.

当人は税金を払ったら、適格である。

In Oracle Policy Modeling this should be written as two separate sentences where the first one is formatted as the conclusion and the second one as the level 1 condition.

The person is eligible.

当人は適格である。

The person pays tax.

当人は税金を払います。

**Example 2 - Clauses**

The person is retired and the person's age is greater than 65.

当人は退職していて、(年齢が) 65歳以上である。

The above sentence should be broken down into two separate discrete sentences.

The person is retired and

当人は退職している。および

The person's age is greater than 65

当人は 年齢が 65歳以上である。

Here the sentences represent two conditions that need to occur simultaneously. This will be reflected by the 'and' rather than inflecting the verb to the *-te* form. Thus, if there are sentences where verb forms that are not covered by the verb editor are used, you should try to rewrite them as separate attributes especially when the sentences are clausal in nature.

**Example 3 - Necessity**

The parser provides the *nakere nara bai* (なければならぬ) form for expressing the notion of 'must' or necessity. This form conjugates only for past and present tense; no conjugations are required for politeness level. If this form does not suit the sentence being expressed in Oracle Policy Modeling, then the sentence can be restructured as follows.

Sentences can be rephrased to use a noun + copula form. Another way is to simply rephrase the sentences. For example, 'A person must have a pension card' changes to

'A person owns a pension card'.

## Write rules in Korean

### Supported sentence structures

The Korean parser supports two kinds of sentences:

1. Verbless sentences

An example of a verbless sentence is 이 사람은 자영업자이다 (The person is self-employed).

2. Subject – Object – Verb sentences

An example of a SOV sentence is 나는 사과를 먹었다 (I ate an apple).

### Supported verb forms

Korean verbs are inflected for politeness level, tense, aspect, voice and sense.

The verb dictionary provides the plain (colloquial) and the polite forms of the verbs.

There are only two tenses in Korean, past and non-past. The non-past covers both the present and the future tense.

The verb aspect denotes the conjugations for perfect, progressive, potential and must forms. The perfect aspect is the stative form of the verb. The must form denotes necessity.

The verb sense indicates whether the verb inflects for a positive or a negative statement. For each of the above, the verbs are inflected by suffixing some ending based on which verb group they belong to.

The verbs do not inflect for gender or person.

The verb voice refers to whether the verb is an active or passive mode. In Korean, the passive form in turn inflects for politeness level and aspect, and is therefore treated as a verb in its own right. The passive verb forms are entered by using their dictionary form in the verb dictionary.

The postpositions 이다 (ida) and 아니다 (anida) have also been included as part of the verb dictionary.

For compound verbs where only the second verb is inflected, for example 유명하다 which is the composite of 유명 and 하다, 하다 is taken to be the active verb. For such (noun + verb) verbs, there is no need to enter the compound verbs separately as long as the active verb belongs in the verb dictionary.

The following are the verb forms present in the verb dictionary:

- Dictionary form: the verb conjugations below are derived from the dictionary form which has to end in -da such as *ha-da*
- Present tense forms:
  - polite
  - plain
- Past tense forms:
  - polite
  - plain
- For each of the polite and plain forms above, the following verb conjugations are provided:
  - positive and negative
  - potential positive and potential negative

- progressive positive and progressive negative
- must positive and must negative

The automatic verb conjugations works for majority of the verbs. The conjugations for irregular verbs will have to be entered manually. See [Configure list of recognized verbs](#) for more information.

Adjectives in Korean behave very much like verbs. For example, all adjectives are conjugated for the politeness level, sentence sense and aspects mentioned above. Thus, the adjectives are also entered using the verb dictionary. In spite of all the similarities with verbs, the adjective conjugation has a few peculiarities. For example, adjectives do not conjugate for progressive aspect. Therefore, the text boxes corresponding to progressive aspect will always be empty for adjectives. When using the verb editor, if you select the dictionary form as being an adjective, these dissimilarities are handled by the verb conjugator.

If you follow the sentence structure guidelines here for creating Korean sentences, then the verbs or the adjectives in the sentence will always end in *-da* (다). Verb forms that end in characters other than *da* are not handled. These include imperative form, inquisitive form, connective and, connective if and certain propositive form.

If adjectives ending in *하다* are used in sentences, then these adjectives must first be entered in the verb editor. This is because the verbs and the adjectives that end in *hada* behave differently. The system has no way of differentiating between a verb and an adjective unless they are already a part of the verb dictionary. The verbs ending in *hada* behave correctly because they use the inflections of *hada* as endings. On the other hand, with adjectives in plain present positive form, the ending gets changed slightly (the ending *hada* is used as opposed to *한다*).

## Adjectives

In an SOV sentence, the verb is taken to be the active verb. If adjectives are present within the sentence, they are not inflected.

In a verbless sentence, the adjectives *are* inflected based on the noun + verb combination or the postposition used at the end of the sentence.

## Sentence parsing

When parsing sentences in Oracle Policy Modeling, the following parts of the sentence are underlined:

1. verbs (that are already included in the verb dictionary)
2. adjectives (that are already included in the verb dictionary)
3. compound verb forms (the parser can recognize when the dictionary entry is preceded by characters making it a compound verb form)

### NOTES:

- i. When two or more adjectives or verbs are present in the sentence, the last adjective or verb gets underlined. This is in accordance with Korean grammar whereby the active component always occurs towards the end in a sentence.
- ii. For compound verbs, if the full verb 'v1 + v2' already exists in the dictionary and is subsequently deleted, this does not impact the parsing of the sentence. This is because the active part of the verb v2 still exists in the verb dictionary. Once the first parse is deleted, the sentence can be successfully reparsed again.

## Limitations

The following verb inflections are currently not handled.

1. Presumptive mood - expresses probability, belief or intention
2. Imperative mood - expresses commands
3. Causative mood - conveys the idea of making or causing someone to do something
4. Conditional mood - conveying 'if', 'unless', 'when' meaning

5. Clauses - conveys sequential, parallel or causal relationships (such as the *~te* and *~de* forms)
6. Counter words

The first three forms are unlikely to occur in the Oracle Policy Automation rulebase framework. For the fourth and fifth verb forms, Oracle Policy Automation has an existing framework for expressing conditionals and clausal relationships when developing a rule-base. As such, these verb inflections are redundant. For expressing 'must', the sentence should be rephrased, for example using the verb 'obligated'.

The parser only supports limited number of counter words such as those for age and number of people.

For example, look at the following sentences.

**Example 1 - Conditional mood**

The person is eligible if the person pays tax.

이 사람은 세금을 납부한다면 자격이 있다

In Oracle Policy Modeling this should be written as two separate sentences where the first one is formatted as the conclusion and the second one as the level 1 condition.

The person is eligible.

이 사람은 자격이 있다

The person pays tax.

이 사람은 세금을 납부한다

**Example 2 - Clauses**

The person is retired and the person's age is greater than 65.

이 사람은 은퇴하였으며 나이는 65세 이상이다

The above sentence should be broken down into two separate discrete sentences.

The person is retired and

이 사람은 은퇴하였다 그리고

The person's age is greater than 65

이 사람의 나이는 65세 이상이다

Here the sentences represent two conditions that need to occur simultaneously. This will be reflected by the 'and' (그리고) rather than inflecting the verb. Thus, if there are sentences where verb forms that are not covered by the verb editor are used, one should try to rewrite them as separate attributes especially when the sentences are clausal in nature.

**Variable sentence generation**

This refers to creating question, uncertain and unknown forms for sentences .

If you want to use "what" or "who" in the question form for variables (eg 'the person's age'), it is really hard to infer the correct form of the words to use since they depend on the semantics of the sentence in question.

As a rough rule if the last word (in this case '나이') ends without a tail consonant, just add "는" after the word. So in this case, the sentence changes to "그 사람의 나이는?".

However, if the last word ends with a tail consonant (for example '생일' or '수입'), you need to add "은" after the word. For instance, 'the person's birthday' is '그 사람의 생일은?' and 'the person's income' is '그 사람의 수입은?'.

## Write rules in Portuguese

### Supported sentence structures

Both Portuguese (European) and Portuguese (Brazilian) parsers support Subject – Verb – Object sentences.

### Supported verb forms

Portuguese verbs conjugate in a significant number of tenses and several moods. The indicative mood is used for factual statements; the subjunctive mood is used for uncertain sentences, as well as some "if ... then ..." sentences.

Although gender (masculine and feminine) is present in Portuguese, the second and third person forms in most tenses are the same for both genders. Additionally, the polite second person form uses the third person verb conjugation.

The verbs list in Oracle Policy Modeling contains entries for the following tense and mood combinations:

Tense	Mood	Verb forms
Present	Indicative	Singular 3rd person, plural 3rd person
Imperfect	Indicative	Singular 3rd person, plural 3rd person
Future	Indicative	Singular 3rd person, plural 3rd person
Preterite	Indicative	Singular 3rd person, plural 3rd person
Present	Subjunctive	Singular 3rd person, plural 3rd person
Imperfect	Subjunctive	Singular 3rd person, plural 3rd person
Future	Subjunctive	Singular 3rd person, plural 3rd person
Gerund	N/A	One form
Past Participle	N/A	Masculine singular, masculine plural, feminine singular, feminine plural

The parser supports both simple and compound verb constructions. For example:

- a simple verb sentence is *a moça **estuda** bem* (the lady studies well)
- a compound verb sentence is *os salários **foram pagos** pelas empresas públicas* (the salaries were paid by the public companies)

### Limitations

#### Substitutions

In order for the substitution to work correctly, every variable and entity should either be preceded by the article, or by a contracted preposition + article.

The following prepositions are supported for substitution:

1. *de* – the contracted forms **do** and **da** are supported.
  - a. For example, *a caneca do candidato* (the mug of the candidate) becomes *a caneca do Leo* when *Leo* is substituted for *o candidato*.
2. *por* – the contracted forms **pelo** and **pela** are supported.
  - a. For example, *pelo candidato* (by the candidate) becomes *por Leo* when *Leo* is substituted for *o candidato*.
3. *a* – the contracted forms **ao** and **à** are supported.
  - a. For example, *ao agente fiscal* (to the fiscal agent) becomes *à Lia* when *Lia* is substituted for *o agente fiscal* and the gender of the variable *o agente fiscal* is set to Feminine at runtime.
4. *para* – the form **para** is supported.
  - a. For example, *para a senhora* (for the lady) becomes *para Lia* when *Lia* is substituted for *a senhora*.

The following guidelines must be followed for substitution to work correctly:

1. Each variable attribute should include the article. For example, use *o candidato* instead of *candidato*.
2. Each variable used in a nominative sentence should include the article. For example, use *o candidato tem a caneca* instead of *candidato tem a caneca*.
3. Each variable used in a sentence with a preposition should include the appropriate contracted preposition + article. For example, use *os pontos foram atribuídos ao agente fiscal* instead of *os pontos foram atribuídos o agente fiscal*.
4. There must be exactly one space between the article/preposition and the variable name. For example, use *os pontos foram atribuídos ao agente fiscal* instead of *os pontos foram atribuídos ao agente fiscal*.

In order for 2nd person substitution of possessives to work correctly, the object possessed may consist of one word only. For example, *a caneca do candidato* can be correctly transformed into *a sua caneca*. However, *a caneca azul do candidato* (the candidate's blue mug) cannot be correctly transformed into a 2nd person sentence. It can, however, be transformed into a 3rd person sentence *a caneca azul do %varid?%* where "varid" is the public name for the variable *o candidato*.

## Write rules in Russian

### Supported sentence structures

The Russian parser supports two kinds of sentences: verbless sentences and Subject – Verbed Predicate – Object sentences.

1. An example of a verbless sentence is *Налогоплательщик счастливый* (The taxpayer [is] happy). Note that the parser only supports single-word predicates in verbless sentences. If a multi-word predicate is required, use the explicit verb *является* ("is").
2. A Subject – Verbed Predicate – Object sentence may have either a simple predicate (one verb), or a compound predicate (multiple verbs). For example, *Кандидат делает это завтра* (The candidate does this tomorrow) is a sentence with a simple predicate. The following sentence has a compound predicate: *Налогоплательщик был обязан уплатить налог* (The taxpayer was required to pay a tax).

## Supported verb forms

The same Russian verb may have two different versions (two infinitive forms), attributing to perfective and imperfective aspects. For example, the verb 'to do' has the versions *сделать* ("perfective", to complete) and *делать* ("imperfective" to be doing). The parser considers each version of the verb as a separate verb, therefore *сделать* and *делать* are two separate entries in the verbs list.

Passive voice in Russian is usually represented by participles, which in the sentence can play the role of either an attribute (like an adjective) or a predicate (like a verb). The parser is only concerned with participles that act like verbs.

The following are the verb forms present in the verbs list in Oracle Policy Modeling:

- Infinitive
- Present tense forms: singular third person, plural third person, plural second person
- Past tense forms: singular third person masculine, singular third person feminine, singular third person neuter, plural
- Short form of past participle, used for forming the passive tense (only for perfective verbs): singular third person masculine, singular third person feminine, singular third person neuter, plural

## Limitations

### Substitutions

Due to the difficulties of modifying case of nouns, the Russian parser only substitutes the following kinds of nouns:

- 3rd person nominative nouns
- 2nd person nominative or genitive nouns

For example, the parser can substitute a name instead of the *налогоплательщик* variable "taxpayer" in the following sentence:

**налогоплательщик** был обязан уплатить налог (the taxpayer was required to pay tax)

becomes **%taxpayer?** был обязан уплатить налог

which becomes, for example: **Иванов** был обязан уплатить налог (Ivanov was required to pay tax)

The 2nd person sentence for this variable is:

**Вы были обязаны** уплатить налог (You were required to pay tax)

However, the parser does not substitute 3rd person names into other noun forms. For example, the following sentence will not have a 3rd person substitution for *налогоплательщик*, since *налогоплательщик* is in genitive:

доход **налогоплательщика** является алиментами (the taxpayer's income is child support payments)

However, the same sentence will have a 2nd person substituted version for the *налогоплательщик* attribute:

**Ваш доход** является алиментами (Your income is child support payments)

NOTE: When substituting 2nd person genitive, the parser always places "yours" at the very beginning of the phrase.

For example, given the attribute:

домашний адрес **студента** (the student's home address)

the 2nd person substitution for *студент* is:

**ваш** домашний адрес (your home address)

### Gender of non-Boolean attributes

In order to form attributes such as *Имя не известно* (the first name is unknown) or *Фамилия не определена* (the family name is uncertain), the parser needs to know the grammatical gender of each noun.

For text attributes, the user should [choose the appropriate gender](#) in the New Attribute dialog. For non-text attributes, the parser attempts to determine the gender by examining the ending of the supplied noun. If the determined gender is incorrect and hence the generated sentences are incorrect, the user may override the generated sentences (see [Customize sentence text](#) for more information).

## Write rules in Spanish

### Limitations

#### Contractions

The Spanish parser does not currently support creating contractions after substitution has taken place. For example, consider the following attributes that allow for substitution:

*la persona*

*la organización*

and the following Boolean attribute, which conveys that "the person belongs to an organization":

*la persona pertenece a una organización*

Oracle Policy Modeling generates the following interrogative form for the Boolean attribute:

*¿La %persona?% pertenece a %organización?%?*

If the person is identified as John and the organization is defined as "the Iron Worker Guild", Oracle Policy Modeling would generate the following question:

*¿John pertenece a el Gremio de Trabajador de Hierro?*

Unfortunately, the question should contract the "a" and "el" as follows:

*¿John pertenece aul Gremio de Trabajador de Hierro?*

Currently, Oracle Policy Modeling will not perform the contraction.

## Write rules in Turkish

### Verb editor

The Turkish verb editor automatically conjugates verbs according to basic sound rules. However, for some verbs, it may create an incorrect 3rd person singular Present Aorist positive form, and derived forms.

For example, the verb *derletmek* ("to make compile") has the following automatically conjugated forms:

Verb Editor

Mastar Edatı: derletmek

Olumlu

	Şimdiki Zamanı	Geniş Zamanı	Gelecek Zamanı
o	derlediyor	derleder	derledecek
oñlar	derlediyorlar	derledeler	derledecekler
şiz	derlediyorsunuz	derledersiniz	derledeceksiniz
	Görülen Geçmiş Zamanı	Öğrenilen Geçmiş Zamanı	
o	derletti	derletmiş	
oñlar	derlettiler	derletmişler	
şiz	derlettiniz	derletmişsiniz	

Tamam Fıllı Çekin İptal

You can update the 3rd person singular Present Aorist positive form with the appropriate verb conjugation:

Verb Editor

Mastar Edatı: derletmek

Olumlu

	Şimdiki Zamanı	Geniş Zamanı	Gelecek Zamanı
o	derlediyor	derletir	derledecek
oñlar	derlediyorlar	derletirler	derledecekler
şiz	derlediyorsunuz	derletirsiniz	derledeceksiniz
	Görülen Geçmiş Zamanı	Öğrenilen Geçmiş Zamanı	
o	derletti	derletmiş	
oñlar	derlettiler	derletmişler	
şiz	derlettiniz	derletmişsiniz	

Tamam Fıllı Çekin İptal

The other related verb forms have now been correctly updated by the verb editor.

# Variables and constant values

Topics in "Variables and constant values"

- Define an attribute to use in a rule
- Choose a name for an entity, relationship or attribute
- Choose a data type for an attribute
- Use variables in rules
- Walkthrough: Creating and using a variable in a rule
- Use constant values in rules
- Check if a value is within a certain range
- Create a synonym for a variable
- Convert a text string into a number or date
- Convert a number or date into a text string
- Combine multiple text strings into a single text variable
- Extract part of a text string
- Check if a text string contains a given substring
- Check if a text string is a number
- Find the length of a text string
- Get a date, day, month or year
- Get a time, second, minute or hour
- Get a date and time
- Get the latest or earliest date or time
- Calculate a relative date
- Find a date in a year
- Count periods between two dates or times
- Calculate the number of days in a month
- Find the day from a date

See also:

- Format a numeric constant for the correct region

## Define an attribute to use in a rule

An attribute is a single unit of data or fact. For example:

- the cost of the movie ticket
- the person is a full-time student

Rules are constructed by combining attributes. For example:

CONCLUSION: the cost of the movie ticket = \$12 if

CONDITION: the person is a full-time student

Attributes can either have a Boolean values (true/false) or take on a Variable value (eg a number, date, text etc.). The following are some examples of attributes and types:

- the person is hungry (Boolean attribute)
- the person's name (Variable attribute – Text)
- the person's date of birth (Variable attribute – Date)
- the number of cookies the person wants to eat (Variable attribute – Number)
- the cost of the person's meal (Variable attribute – Currency)
- the time of sunrise (Variable attribute – Time of day)

A variable attribute must be created before it can be used in a rule. Creating a variable tells Oracle Policy Modeling how you are intending to use the variable and the type of information you want it to represent. (Boolean attributes do not need to be created before they can be used in rules but it can be useful to do so in order to define public names. For more information, see [Define attribute names for use by external applications.](#))

Attributes are typically created in a properties file in Oracle Policy Modeling. This allows the attribute to be added once and used across all rule documents. This also allows you to define public names, validation and other properties for the attribute. Attributes can be added to an existing properties file from within Word while writing the rules.

Attributes can also be created directly in a Word or Excel rules file. This method is only appropriate for variables that are only used in a single rule document.

Every attribute is assigned to an entity. An attribute is assigned to an entity if it contains the entity text.

## What do you want to do?

[Create a new attribute from within a Word document](#)

[Create a new attribute in an Excel document](#)

[Create a new attribute in a properties file](#)

[Check attribute entity levels](#)

Create a new attribute from within a Word document

To add an attribute within Word:

1. Write your rules using your yet-to-be-created attribute, but before compiling your rules, select the attribute text and click the **Add Attribute** button on the Oracle Policy Modeling toolbar. NOTE: The text of the attribute must contain the name of the entity to which it belongs, otherwise it will not be associated with that entity. For more information on naming attributes, see [Choose non-boolean attribute text](#) and [Check attribute entity levels](#).
2. In the **Add Attribute** dialog box, select the **Type** of the attribute from the drop-down list. For more information on attribute data types, see [Choose a data type for an attribute](#).

**Add Attribute**

Public Name:

Type:

Text:

File:

3. Select the properties file that you want to add the attribute to from the **File** drop-down list. Alternatively, if the attribute is only going to be used in this document, you can put the attribute at the top of the document rather than in a properties file by selecting **<Top>**.
4. If you have chosen to add the attribute to a properties file, enter a **Public name** for the attribute if required. (NOTE: All base level attributes and all top level attributes need public names. Important intermediate attributes also need to have public names. For more information, see [Set public identifiers for entities and attributes.](#))
5. If you have chosen to add the attribute to a properties file, select the **Entity** that the attribute belongs to. (NOTE: Usually this will have been automatically determined based on the inclusion of entity text in the attribute text, but if the entity is ambiguous you will need to select the appropriate entity.)
6. Click **OK**.

### Create a new attribute in an Excel document

Attributes that are not used by any other document can be created directly in the Excel document itself, rather than in the project's properties files.

To create an attribute directly in an Excel document:

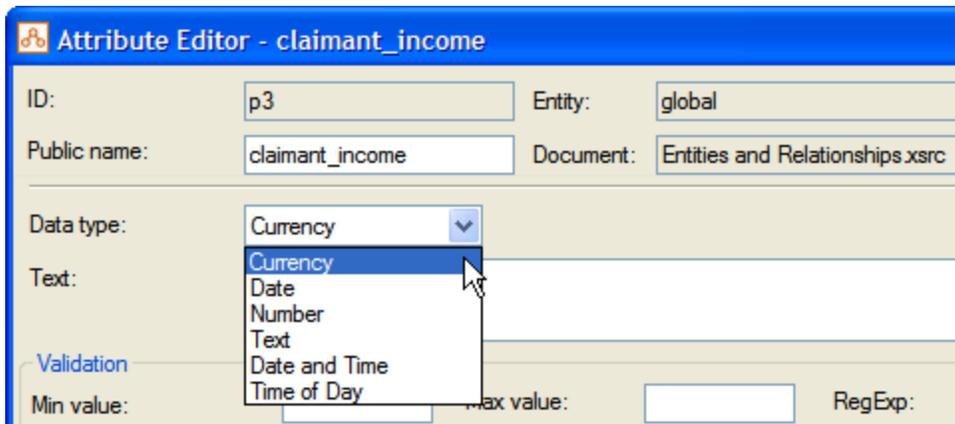
1. Open the **Declarations** worksheet.
2. In the **Attribute Type** column enter the [type of attribute](#), and in the **Attribute Text** column enter the [text of the attribute](#).

TIP: If the default Declarations worksheet has been deleted or altered and these columns don't exist, simply enter your attribute type and attribute text in adjacent cells and apply the appropriate Oracle Policy Modeling styles using either the Oracle Policy Modeling menu or the Oracle Policy Modeling toolbar.

### Create a new attribute in a properties file

To create an attribute in a properties file:

1. In Oracle Policy Modeling, double click the properties file in the Project Explorer to open it for editing.
2. On the Attributes tab, right-click and select **New Attribute...**
3. In the **Data type** drop-down list, select the [type of attribute](#) from the drop-down list.



4. In the **Text** field, enter the [attribute text](#).
5. Click **OK** to create your attribute.

#### Check attribute entity levels

After you have defined an entity, every attribute added to a Word document which contains the entity text will attach to that entity. Attributes which do not contain entity text are global.

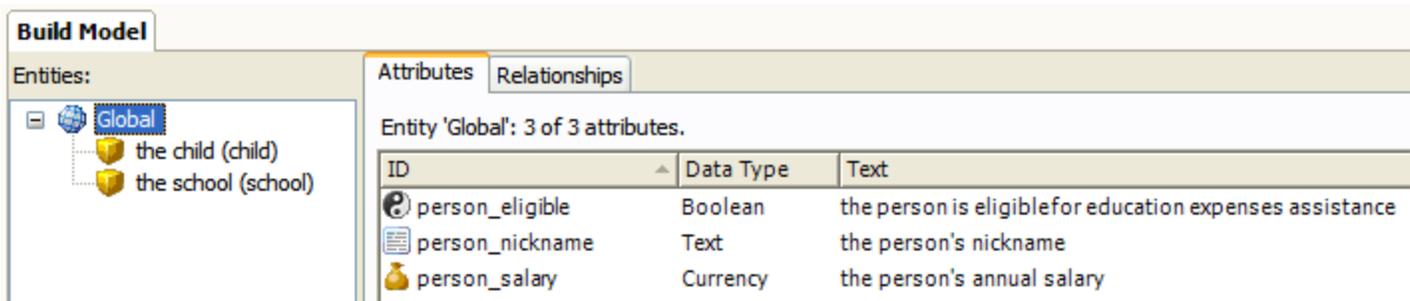
For example, assume the attributes in the following table have been added to a Word document where "the household member" has been defined as an entity in the rulebase:

Attribute Text	Entity Level	Explanation
the household member is male	the household member	contains "the household member"
a household member is eligible	global	"a household member" does not match "the household member"
the former household member has left	global	"former" interrupts the attribute text
the household member's annual income	the household member	adding extra letters or characters on the left or right hand side is ok
the date of birth of the household member	the household member	entity text may appear anywhere in the attribute text

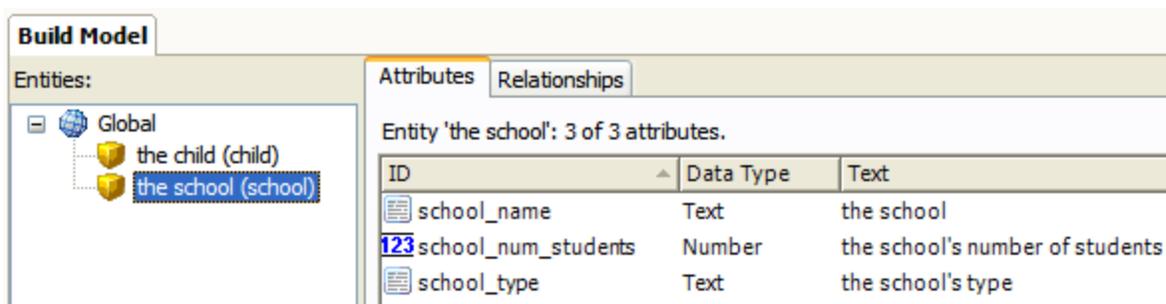
Both Boolean and non-Boolean attributes can be defined to belong to an entity in this way.

Once you have compiled your rules, you can check that all attributes have been associated with the correct entity in the **Build Model** in Oracle Policy Modeling.

Attributes which do not contain any entity text are placed in the **Global** level. The list of global attributes are displayed in the right-hand pane:



To view a list of entity-level attributes, click on the entity name. The list of entity-level attributes will be displayed in the right-hand pane:



See also:

- [View list of entities and attributes](#)

## Choose a name for an entity, relationship or attribute

The naming of entities, relationships and attributes is an important consideration when creating a rulebase.

### What do you want to do?

[Choose a name for an entity](#)

[Choose a name for a relationship](#)

[Choose attribute text](#)

[Document the naming convention for a project](#)

#### Choose a name for an entity

Entities should be named using the definite article 'the', as in 'the family', 'the child', 'the friend', 'the school' etc.

#### Choose a name for a relationship

When creating a relationship you should give the relationship a meaningful name. Remember that the relationship describes the reference from one entity instance to one or more of another entity instance. The relationship name should therefore include the target entity text so that it is clear from the relationship name who the relationship is to.

The name of the relationship should reflect the everyday expression used to describe the relationship (if there is one), and should be clear in and out of context what is being referred to. Try to consider that nature of the relationship you are capturing and give it a name that represents this relationship.

Where you are referring to a single instance ("to-one" relationships), your relationship text must therefore be singular. When you are referring to multiple instances ("to-many" relationships), your relationship text must be plural. Where one entity is the global entity, you may simply refer to the target entity.

### Examples of relationship names

Relationship type	Entity 1	Entity 2	Relationship text
One-to-one	"the child"	"the friend"	"the child's best friend"
Many-to-one	"the child"	"the family"	"the child's family"
One-to-many	Global	"the child"	"the children"
Many-to-many	"the child"	"the friend"	"the child's friends"
Self-referential one-to-one	"the child"	"the child"	"the child's twin"

### Choose attribute text

Selecting correct attribute wording is fundamental to capturing logic accurately in your Oracle Policy Modeling application and conveying information to a user in a meaningful way. Specifically, attribute text influences:

- **The logic of a rule condition**

The logic of a rule is not just captured in the rule levels. There is intrinsic logic in the construction of a sentence and the negation of that sentence. For example: "No child appears in the photo" will be negated as "no child does not appear in the photo" which is logically incorrect.

- **The connections between rules**

Rules are connected in the rulebase using plain text matching. A condition of one rule will only be automatically linked to the conclusion of another rule if the text is exactly the same. For example, the text "the doctor's waiting room is full" will not automatically connect to "the doctors' waiting room is full" as the apostrophe is in a different place in the sentence.

- **The display of question text on interview screens**

The user will see the wording of the attribute on any question screens created for the application unless you override this text.

- **The wording of attributes in decision reports**

The decision report is an important mechanism for understanding how the rules are operating. Incorrect attribute text will make it more difficult to debug errors and may mislead or confuse users.

### Choose boolean attribute text

The following general principles apply to the writing of Oracle Policy Modeling boolean attributes.

#### 1. Boolean attributes should be complete grammatical sentences

An Oracle Policy Modeling boolean attribute must include at a minimum a subject and verb. The subject is what or who the sentence is about. The verb tells us something about the subject. Most sentences also contain an object which is the thing the action is being performed on.

Examples of grammatical sentences are:

the investigation continued (subject – verb)

the lion stalked the gazelle (subject – verb – object)

## 2. Boolean attributes should generally be written in the past tense

The tense of a verb is used to indicate when the action took place. Your top level goal should usually be worded in the present tense as it describes the current state of affairs. However, everything below the top level goal should be written in the past tense as it describes what occurred for the top level conclusion to have been reached.

For example:

**the person is eligible for an award (PRESENT TENSE) if**

the person has demonstrated exceptional conduct (PAST TENSE)

**the person has demonstrated exceptional conduct (PAST TENSE) if**

the person has been commended by peers (PAST TENSE)

This principle applies regardless of the tense of the source material.

## 3. Boolean attributes should be written in the third person

In English grammar we make a distinction between the speaker/s (I, we), the addressee (you), and the one/thing spoken about (he, she, it, they). This is known as person: first, second and third person, respectively. Boolean attributes should be written in the third person. (Note that there is a mechanism in Oracle Policy Modeling for [switching attribute forms to second person](#) for use in interviews.)

For example:

the person can go to the movies

the person has done a good job

Rather than:

I can go to the movies

you have done a good job

## 4. Boolean attributes must be able to be negated

Some boolean attributes can be difficult to negate and for this reason should be avoided.

Examples are attributes which use the conjunctions 'and' and 'or'. In these attributes ambiguity can result from the negation of the attribute as we don't necessarily know how the negation of the verb should affect each of the components. For instance, let's look at the attribute "the cat and the dog ate the man's dinner".

If this attribute is false, this could mean that:

- i. neither the cat or the dog ate the man's dinner
- ii. the cat ate the man's dinner but the dog did not
- iii. the dog ate the man's dinner but the cat did not

Given that there are three possible interpretations means that this attribute cannot be negated conclusively and should not be used.

## 5. Boolean attributes should represent a single concept

In many instances, it may be tempting to word an attribute that could be split into two separate clauses as a single attribute.

However, if it is likely that part of the attribute is going to be used in other attributes, it is best to separate it into two attributes which each represent distinct concepts.

## 6. Boolean attributes should not use contractions

Contractions are used in more informal styles of writing and speech and should not be used in Oracle Policy Modeling attributes. For example, rather than "there's an application pending", you should write "there is an application pending".

## 7. Boolean attributes should make sense without reference to another attribute

Each boolean attribute should be meaningful without reference to another. To do otherwise makes the rulebase more difficult to develop, maintain and audit.

The following are examples of attributes which do not make sense in isolation:

- This section has been satisfied
- That discussion was recorded
- The person qualifies for the reasons above
- The latest of these two dates applies

## 8. Boolean attributes should be kept simple but explicit

The wording of the attribute should be as simple as possible while still retaining its full intended meaning.

## 9. Boolean attributes should indicate entity membership

If the attribute belongs to an entity, the exact text of the entity should be included in the attribute text to make it clear which entity it belongs to. For example, if you have an entity 'the child', then attributes which belong to that entity group should include the text "the child":

the child is happy  
the child's toy is educational  
the birthdate of the child

## 10. Boolean attributes should not use personal pronouns

A variable can be replaced with the appropriate pronoun the second (and any subsequent times) the variable is used in a boolean attribute. For example, if we had a variable 'the claimant' we could write a boolean attribute 'the claimant owns the claimant's home' and then once we know the name and gender of the claimant this would be rendered as 'John owns his home'. This is preferable to hard-coding "his/her" or "their" in the attribute text.

## 11. Boolean attributes which refer to amounts should indicate the unit of measurement

Boolean attributes which refer to amounts should specify the unit of measurement to avoid any ambiguity. For example:

the person was 100 feet from the scene of the crime

See also:

- [Basic English grammar](#)

## Choose non-boolean attribute text

When creating non-boolean attributes (variables) the following guidelines apply:

### 1. Non-boolean attributes need to start with the definite article 'the'

The definite article 'the' is used to refer to some specific thing (in contrast to the indefinite article 'a' or 'an' which does not refer to one specific thing). As variables are always referring to a particular thing, they must start with 'the'. For example,

*the claimant's name*  
*the type of animal*  
*the price of the car*

## 2. Non-boolean attributes should indicate entity membership

If a variable belongs to an entity, the text of the entity should be included in the variable text to make it clear which entity it belongs to. For example, if you have an entity 'the child', then variables which belong to that entity group should include the text "the child":

*the child's age*

*the child's date of birth*

*the school that the child attends*

## 3. Non-boolean attributes which refer to amounts should indicate the unit of measurement

To make it clear what unit of measurement is expected for amount variables, this should be included in the variable text. For example:

*the distance between home and work (kilometers)*

*the weight of the truck (tonnes)*

## 4. Non-boolean attributes should reference their source

References to values derived in other sections of the material should explicitly state the origin of these values in the variable text.

Document the naming convention for a project

A Rulebase Naming Conventions document should be created at the start of every Oracle Policy Modeling project to clearly set out a consistent method of wording attributes. This is critical because automatic linking will only work when attributes are an exact text match. If different rule developers use different text when creating separate chunks of rules the attributes will not tie together. The Rulebase Naming Conventions document should define which nouns will be capitalized and whether particular acronyms should be used.

The Rulebase Naming Conventions document can be kept in the Oracle Policy Modeling project under **Documents/Design**.

## Choose a data type for an attribute

When you create a new attribute you need to define the type of attribute it is, based on the kind of information it represents.

The table below shows the types of attributes that are supported in Oracle Policy Modeling:

Attribute type	Icon	When used	Example
Boolean		for statements	the claimant is eligible for family benefits
Currency		for amounts of money	the claimant's annual income
Number		for any type of number	the claimant's age
Text		for text strings	the claimant's name
Date		for date values	the claimant's date of birth
Date and time		when a date and time together is needed	the date and time of the car accident

Attribute type	Icon	When used	Example
Time of day		for times of day	the store's opening time

Note that for datetime and time of day attributes, you have the option in the Attribute Editor to specify whether seconds will be displayed. If 'Display seconds' is unchecked, any seconds values entered in Web Determinations will be discarded.

The format that values of non-boolean attributes (variables) must take in rules is specified in [Use constant values in rules](#).

The format that values of attributes must take when being entered into input fields, and the format as they appear in decision reports, is specified in [Formatting of attribute values](#).

## Use variables in rules

Variables can be used in rules as conditions and as conclusions. For example, you might want to prove the person's age (a number) from a person's date of birth (a date) and perhaps use this attribute as a condition determining whether the person is over the age of 18 (a boolean).

## What do you want to do?

[Specify the value for a variable in a rule](#)

[Use a variable in a condition](#)

[Use a variable in a mathematical calculation in a rule conclusion](#)

[Use a variable in a straight calculation in a rule calculation](#)

## Specify the value for a variable in a rule

To avoid ambiguity, the Oracle Policy Modeling compiler enforces strict formatting on the values of variables where the value is explicitly used in a rule. For the formatting requirements and other considerations when setting the value of a variable in a rule, see [Use constant values in rules](#).

## Use a variable in a condition

Like boolean attributes, variables can be used as conditions in any rule proving another attribute. When using variables in conditions you must state the value, or range of acceptable values, that are sufficient to satisfy the condition. To do this, you must use one of the standard logical operators. The value of the attribute may either be compared to a fixed value ("= 18") or to the value of another attribute ("= the spouse's date of birth").

NOTE: Where two variable attributes are being compared, they must be of the same variable type. When comparing a variable attribute with a constant value, the value must be in the specified format for that type of variable attribute. See [Use constant values in rules](#) for more information.

Operator	Example
Greater than (>)	<b>the person is over 18 if</b> the person's age > 18
Less than (<)	<b>the employee is early for work if</b>

Operator	Example
	the time the employee starts work < the specified start time for the employee
Equals (=)	<b>the person was born on the same day as the person's spouse if</b> the person's date of birth = the person's spouse's date of birth
Not equal to (<>)	<b>the pet is not a monkey if</b> the type of pet <> "monkey"
Greater than or equal to (>=)	<b>the applicant is eligible for a loan if</b> the applicant's annual income >= 50000
Less than or equal to (<=)	<b>the submission is valid if</b> the submission's date and time <= the latest submission date and time

### Use a variable in a mathematical calculation in a rule conclusion

It is possible to perform a variety of mathematical calculations using variables. These operations include:

- standard arithmetic calculations (eg addition, subtraction, multiplication, division)
- mathematical expressions (eg square root, round, truncation)

For the full list of supported operators and functions, see [Numerical functions](#) in the function reference.

For example,

**the cost of the school lunch = the cost of the meat pie + the cost of the bag of the chips + the cost of the soft drink - the amount of the student discount**

**the person's share of household income = (the person's income + the partner's income)/2**

TIP: Whilst the standard mathematical preference is applied to operators in the absence of parentheses (ie division, multiplication, addition, subtraction), you should make the order explicit with the use of parentheses.

### Use a variable in a straight calculation in a rule conclusion

In the same way that a boolean attribute is set to a value when used in a rule conclusion, a variable can be assigned a value in a conclusion. For example, for the variable "the passenger's allowance in Australian dollars" we can write the following rule:

**the passenger's allowance in Australian dollars = 350**

In this case, no conditions are required so the value is always inferred. Therefore, no alternative conclusion is produced.

See also

- [Walkthrough: Creating and using a variable in a rule](#)
- [Formatting of variable values](#)
- [Use constant values in rules](#)

## Walkthrough: Creating and using a variable in a rule

This walkthrough will demonstrate how to create a variable and use it in a rule.

### Source material

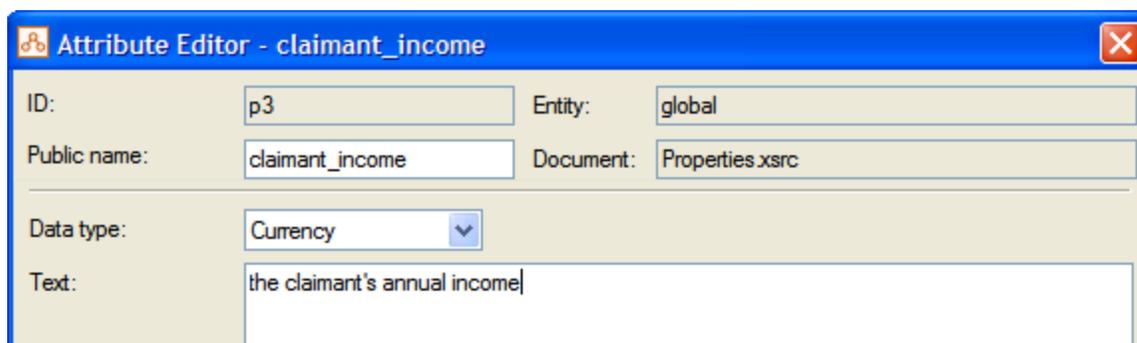
Take, for example, the following source material:

the claimant is eligible for a loan if the claimant's annual income is more than \$15000

### Creating the variable

To create a variable to represent "the claimant's annual income":

1. In Oracle Policy Modeling, double click the properties file in the Project Explorer to open it for editing.
2. On the Attributes tab, right-click and select **New Attribute...**
3. In the **Public name** field, enter "claimant\_income".
4. In the **Data type** drop-down list, select **Currency**.
5. In the **Text** field, enter "the claimant's annual income".



6. Click **OK** to create the variable.

### Using the variable in a Word rule

To use this variable in a rule in your Word document:

1. In Oracle Policy Modeling, double click the Word rules file in the Project Explorer to open it for editing.
2. On a blank line, type "the claimant is eligible for a loan", then click the **Conclusion** button on the Oracle Policy Modeling toolbar or use the shortcut key Alt+C.
3. Place the cursor at the end of this line and press the **Enter** key to create a condition line.
4. Type "the claimant's annual income > 15000". Your rule should look like this:

**the claimant is eligible for a loan if**

the claimant's annual income > 15000

## Use constant values in rules

Constant values can be used to set variable attributes in your rules, or in rule comparisons or calculations. There are some formatting and value requirements to keep in mind when writing your rules in Word or Excel, as detailed below.

Variable type	Format	Range	Example rule	Notes
Number	Any number (supports decimals)	Approximately 30 significant figures may be used for very small or very large numbers, otherwise approximately 15 can be used.	<b>the number of ants on the property = 15,000</b>	Treatment of commas ",", periods "." and spaces " " as decimal and thousand separators are based on the <a href="#">Region</a> settings for the rule-base project. Scientific notation may not be used. There are also considerations in <a href="#">formatting of numeric values in functions</a> .
Currency	Any number (supports decimals)	Approximately 30 significant figures may be used for very small or very large numbers, otherwise approximately 15 can be used.	<b>the person's savings (in dollars) = 534.50</b>	Leading \$ and £ symbols may be used if this enhances readability of your rules, however, note that the formatting of attributes values will be determined by the rulebase region setting when the rule-base is run.
Text	Any string of alphanumeric characters may be used, enclosed by	The limit to the length of a	<b>the household's location = "New York"</b>	To enter a

Variable type	Format	Range	Example rule	Notes
	double quotes "	text string depends on Word/Excel and your system, and should not be a practical limitation in your rule authoring.		double quote character into the text string itself in Word, precede it with a backslash character "\". For example, using the string "the child said \"Hello\"" will produce: the child said "Hello". Double quote characters may be entered directly in Excel and will be treated as entered, except where they surround the entire text string in which case they are ignored.
Date	yyyy-MM-dd	0002-01-01 to 9998-12-31	<b>the date of the last interest rate rise = 2007-10-25</b>	Oracle Web Determinations (OWD) has a slightly different range of acceptable dates (0001-01-02 to 9999-09-09). When writing rules to set constant values for date variables, the relevant date

Variable type	Format	Range	Example rule	Notes
				range restriction is the Oracle Policy Modeling (OPM) restriction, not the OWD restriction. For example, a rule attempting to set a date variable to the value 9999-01-01 will cause a compile error in OPM, even though OWD can process a date of 9999-01-01 as an input. By contrast, the date 9999-12-31 will cause errors in both OWD and OPM. That is, a rule attempting to set a date variable to the value 9999-12-31 will cause a compile error in OPM, and an attempt to input the date 9999-12-31 will cause an error in OWD.
Time of day	hh:mm:ss	00:00:00 to 23:59:59	<b>the store closing time = 17:30:00</b>	
Date and time	yyyy-MM-dd hh:mm:ss	In the ranges detailed above	<b>the submission date time = 2009-08-12 17:30:00</b>	Time zones are

Variable type	Format	Range	Example rule	Notes
		for date and time values		not varied within the scope of a single rule-base, ie there is a single time zone for a rule-base, which is taken to be that of the server on which it is running. If custom processing is required to handle multiple time zones within a rule-base, a <a href="#">custom function</a> may be implemented to perform this.

To see the way Oracle Policy Modeling formats data values in other places such as the debugger or Oracle Web Determinations, see [Formatting of variable values](#).

See also:

- [Use variables in rules](#)

### Check if a value is within a certain range

To check whether a value is within a certain range (eg the claimant is aged between 18 and 25), you write a rule with two conditions. The first condition is used to check if the value is greater than the lower end of the range, while the second condition is used to check if the value is less than the upper end of the range. The two conditions must be connected by an **and**.

For example:

**the claimant is in the eligible age group if**

the claimant's age  $\geq$  18 and

the claimant's age  $\leq$  25

See also:

- [Use variables in rules](#)
- [Comparison operator rule examples](#)

## Create a synonym for a variable

You can define synonyms for variables in your rules to make rules more succinct. To create a synonym for a variable in Word:

1. Put the cursor on a new blank line in your rule document. Type the synonym using the following syntax: **<synonym text> is <variable text>**.
2. Click the **Table Legend** button on the Oracle Policy Modeling toolbar.  
TIP: Alternatively you can use the keyboard shortcut **Alt+L**.
3. Thereafter in your rules document you can refer to the variable using just the synonym text.

### Example

a is the cost of school fees

b is the cost of school uniforms

c is the cost of school excursions

**the total school cost = a + b + c**

## Convert a text string into a number or date

To convert a text string into a number, you can use the following syntax with the Number function:

- `<number variable> = Number(<text>)`

In combination with the [Substring](#) function, you could use the Number function to extract the number part of a text string to a number variable. For example:

**the number at the end of the course code = Number(Substring(the course code, 4, 4))**

If the course code was LAWS2001, this rule would infer the number at the end of the course code to be 2001.

To convert a text string into a date, you can use the following syntax with the Date function:

- `<date variable> = Date (<text>)`

The text string in these functions can be either a value or a variable. If it is a value it must be in quotation marks (eg "2000-04-17").

## Convert a number or date into a text string

To convert a number or date (including date and time, and time of day attributes) into a text string, you can use the following syntax with the Text function:

- `<text variable> = Text(<variable>)`

For example:

**the date of effect text = Text(the date of effect)**

See also:

- [Text function reference](#)

## Combine multiple text strings into a single text variable

To combine the values of two or more variables or text strings into a single text value, you use the string concatenation function. For example, you might want to combine 'the person's first name' (Harriet) with 'the person's last name' (Jones) to create 'the person's name' (Harriet Jones).

The concatenation function is commonly used to create variables that can be [substituted into screen headings and labels](#).

The syntax for using the string concatenation function is:

- the concatenation of <text1> & <text2> & ...
- <text1> & <text2> & ...

For example:

**the person's name = the concatenation of the person's first name & " " & the person's last name**

NOTE: The (" ") part of the rule will insert a blank space between the first name and the last name. Similarly you could use (", ") to insert a comma and blank space in between text variables.

TIP: You can use variables of any type with this function. Values are formatted using the formatter that is installed in the rule session (refer to the topic *Formatter Plugin Overview* in the [Oracle Policy Automation Developer's Guide](#) for more information about formatters). To convert individual non-text variables into text, use the [Text function](#).

## Extract part of a text string

To extract a substring from a text string you use the [Substring](#) function.

The syntax for using the substring function is:

- Substring("<text>", <offset>, <length>)

To use the substring function you need to specify:

- a. the offset number - this represents the number of characters from the beginning of the string, including spaces.
- b. the length number - this is the number of characters collected, starting at the offset point.

For example, to extract the substring 'johns' from the text string 'johnsmith' you could write the following rule:

**username = Substring("johnsmith", 0, 5)**

## Check if a text string contains a given substring

You can check whether a text string contains a particular substring, at the start, end or anywhere within the text string. Either of these text strings can be text variables or text constants. The text comparison is case-insensitive.

### What do you want to do?

Check if a text string contains a particular substring

Check if a text string contains a particular substring at the start of the string

Check if a text string contains a particular substring at the end of the string

## Check if a text string contains a particular substring

To determine if a particular string is contained anywhere in a text variable or text constant, you can use the following syntax with the [Contains](#) function:

- Contains(<text value>, <text substring>)
- <text value> contains <text substring>

For example:

**the system is using US English if**

the system code contains "en-US"

## Check if a text string contains a particular substring at the start of the string

To determine if a particular string is contained at the start of a text variable or text constant, you can use the following syntax with the [StartsWith](#) function:

- StartsWith(<text value>, <text substring>)
- <text value> starts with <text substring>

For example:

**the record was created before 2000 if**

the record identification code starts with "19"

## Check if a text string contains a particular substring at the end of the string

To determine if a particular string is contained at the end of a text variable or text constant, you can use the following syntax with the [EndsWith](#) function:

- EndsWith(<text value>, <text substring>)
- <text value> ends with <text substring>

For example:

**the person has a government email address if**

the person's email address ends with ".gov"

## Check if a text string is a number

To check whether a text string is a number, ie whether it contains only valid number characters, you can use the following syntax with the [IsNumber](#) function:

- IsNumber(<text value>)
- <text value> is a number

For example, to check that an identification number contains only valid number characters, you would write the following rule:

**the identification number is valid if**

the identification number is a number

The text value may be a text attribute, text constant or any expression that returns a text constant.

TIP: any valid number characters may be present in the text string, eg minus sign, decimal point, etc.

## Find the length of a text string

To find the length of a text string, ie the number of characters it contains, you can use the following syntax with the [Length](#) function:

- Length(<text value>)
- the length of <text value>

For example, to check that an identification number is the correct length (10 characters in this case), you would write the following rule:

**the identification number is valid if**

the length of the identification number = 10

The text value may be a text attribute, text constant or any expression that returns a text constant.

## Get a date, day, month or year

There are functions which you can use in your rules to get particular dates, days, months or years.

## What do you want to do?

[Get today's date](#)

[Get the day component of an input date](#)

[Get the month component of an input date](#)

[Get the year component of an input date](#)

[Get the date from a date and time](#)

[Get a date formed from a specified year, month and day](#)

### Get today's date

To insert the system date into a rule, you use the Current Date function. To do this you insert the words "the current date" in the rule. NOTE: These words operate as a function and should therefore not be added as a variable attribute.

For example, the following comparison:

**the date of assessment = the current date**

will infer the date of assessment to be 12/12/2008 if the rule is run on 12/12/2008.

The current date can also be used as an input date in a calculation. For example:

**the date 2 weeks from today = the date 2 weeks after the current date**

will infer the date 2 weeks from today to be 26/12/2008 if the rule is run on 12/12/2008. Here the current date is the input date in an Add Weeks function.

NOTE: The Current Date function returns the system date at the start of the session.

### Get the day component of an input date

To extract the day component of an input date (or date time), you use the Extract Day function. For example:

**the day of expiry = ExtractDay(2009-01-08)**

will infer the day of expiry to be 08. Note that the input date can be a constant as in this example, or a variable, as in the example below:

**the day of expiry = ExtractDay(the use-by date on the packet)**

### Get the month component of an input date

To extract the month component of an input date (or date time), you use the Extract Month function. For example:

**the birth month = ExtractMonth(2004-11-21)**

will infer the birth month to be 11. Note that the input date can be a constant as in this example, or a variable, as in the example below:

**the birth month = ExtractMonth(the date of birth)**

### Get the year component of an input date

To extract the year component of an input date (or date time), you use the Extract Year function. For example:

**the year the warranty expires = ExtractYear(2013-11-21)**

will infer the year the warranty expires to be 2013. Note that the input date can be a constant as in this example, or a variable, as in the example below:

**the year the warranty expires = ExtractYear(the date 5 years after the purchase date)**

### Get the date from a date and time

To extract the date from a date and time attribute, you use the ExtractDate function. For example:

**the date of manufacture = ExtractDate(the datetime of manufacture)**

will infer the date of manufacture to be 2011-12-05 when the datetime of manufacture is 2011-12-05 11:31:45.

### Get a date formed from a specified year, month and day

To form a date from a specified year, month and day, you use the Make date function. For example:

**the calculation date = MakeDate(2007, 10, 17)**

would make the calculation date 17-10-2007.

See also:

- [Date function reference](#)
- [Date function rule examples](#)

## Get a time, second, minute or hour

There are functions which you can use in your rules to get particular times, seconds, minutes or hours.

### What do you want to do?

Get the second component of an input time

Get the minute component of an input time

Get the hour component of an input time

Get the time of day from a date and time

Get the time of day from a text string

### Get the second component of an input time

To extract the second component of an input time (ie from a time of day variable or a date time variable), you use the Extract Second function. For example:

**the second component of the submission time = ExtractSecond(16:30:42)**

will infer the second component of the submission time to be 42. Note that the input time can be a constant as in this example, or a variable, as in the example below:

**the second component of the submission time = ExtractSecond(the submission time)**

In all cases the value returned is a number.

### Get the minute component of an input time

To extract the minute component of an input time (ie from a time of day variable or a date time variable), you use the Extract Minute function. For example:

**the minute component of the submission time = ExtractMinute(16:30:42)**

will infer the minute component of the submission time to be 30. Note that the input time can be a constant as in this example, or a variable, as in the example below:

**the minute component of the submission time = ExtractMinute(the submission time)**

In all cases the value returned is a number.

### Get the hour component of an input time

To extract the hour component of an input time (ie from a time of day variable or a date time variable), you use the Extract Hour function. For example:

**the hour component of the submission time = ExtractHour(16:30:42)**

will infer the hour component of the submission time to be 16. Note that the input time can be a constant as in this example, or a variable, as in the example below:

**the hour component of the submission time = ExtractHour(the submission time)**

In all cases the value returned is a number.

## Get the time of day from a date and time

To extract the time of day from a date and time attribute, you use the Extract Time of Day function. For example, to determine the current time (ie at the start of the session), you would use the Current Date Time function and extract the time from it using the Extract Time of Day function:

**the current time = ExtractTimeOfDay(the current date time)**

This will infer the current time to be 15:30:00 if the rule is run on 2008-12-12 at 15:30:00.

## Get the time of day from a text string

To convert a text string into a time of day variable, you use the Time Of Day function. For example:

**the latest submission time = TimeOfDay("17:00:00")**

will infer the latest submission time to be 17:00:00 if the text string is "17:00:00".

See also:

- [Time of day function reference](#)
- [Time of day function rule examples](#)
- [Date and time function reference](#)
- [Date and time function rule examples](#)

## Get a date and time

There are functions which you can use in your rules to get a particular date and time.

### What do you want to do?

[Get the current date and time](#)

[Get a date and time by joining together a separate date and time](#)

[Get a date and time from a text string](#)

[Get a date and time by adding or subtracting a specified number of hours to another date and time](#)

[Get a date and time by adding or subtracting a specified number of minutes to another date and time](#)

[Get a date and time by adding or subtracting a specified number of seconds to another date and time](#)

### Get the current date and time

To insert the system date and time into a rule, you use the Current Date Time function. For example, the following comparison:

**the date and time of the investigation = CurrentDateTime()**

will infer the date and time of the investigation to be 2007-11-12 15:37:00 if the rule is run on 2007-11-12 at 15:37:00.

NOTE: The Current Date Time function returns the system date/time at the start of the session.

### Get a date and time by joining together a separate date and time

To set a date and time from a separate date and a separate time, you use the Concatenate Date Time function. For example:

**the latest submission time = ConcatenateDateTime(the submission date, the submission closing time)**

will infer the latest submission time to be 2010-01-15 17:00:00 if the submission date is 2010-01-15 and the submission closing time is 17:00:00.

### Get a date and time from a text string

To set the value of a date and time variable from a text string, you use the DateTime function. For example:

**the latest submission date and time = DateTime(the submission date and time specified on the application form)**

will infer the latest submission date and time to be 2012-12-31 18:00:00 if the submission date and time specified on the application form is a text variable with the value of 2012-12-31 18:00:00.

### Get a date and time by adding or subtracting a specified number of hours to another date and time

To add or subtract a specified number of hours to an input date and time to get a new date and time, you use the Add Hours function. For example:

**the start datetime for the B grade runners = the time 2 hours before the start datetime for the A grade runners**

will infer the start datetime for the B grade runners to be 2011-02-03 08:00:00 if the start datetime for the A grade runners is 2011-02-03 10:00:00.

### Get a date and time by adding or subtracting a specified number of minutes to another date and time

To add or subtract a specified number of minutes to an input date and time to get a new date and time, you use the Add Minutes function. For example:

**the datetime that the parking meter expires = the time 60 minutes after the datetime that the parking fee was paid**

will infer the datetime that the parking meter expires to be 2012-10-10 12:04:17 if the datetime that the parking fee was paid is 2012-10-10 11:04:17.

### Get a date and time by adding or subtracting a specified number of seconds to another date and time

To add or subtract a specified number of seconds to an input date and time to get a new date and time, you use the Add Seconds function. For example:

**the completion datetime = AddSeconds(the start datetime, 30)**

will infer the completion datetime to be 2009-01-01 16:30:00 if the start time is 2009-01-01 16:29:30.

See also:

- [Date and time function reference](#)
- [Date and time function rule examples](#)

### Get the latest or earliest date or time

To get the latest or earliest date, date and time, or time of day, you use the Maximum and Minimum functions.

- Maximum(<date/timeofday/datetime1>, <date/timeofday/datetime2>)
- the latest of <date/timeofday/datetime1> and <date/timeofday/datetime2>

- `Minimum(<date/timeofday/datetime1>, <date/timeofday/datetime2>)`
- the earliest of `<date/timeofday/datetime1>` and `<date/timeofday/datetime2>`

For example, to get the date of the most recent event, you could write

**the most recent event date = the latest of the date of the annual Arts Festival and the date of the annual Music Festival**

If the date of the annual Arts Festival was 2001-05-05 and the date of the annual Music Festival was 2001-03-15, then the most recent event date is 2001-05-05.

To get the earliest completion time of two teams, you could write:

**the earliest completion time = Minimum(the completion time of Team A, the completion time of Team B)**

If the completion time of Team A is 16:45:02, and the completion time of Team B is 16:14:18, then the earliest completion time is 16:14:18.

## Calculate a relative date

There are functions that you can use in your rules to calculate a date relative to another date. You can use both constants and variables for both date and number inputs in these rules.

### What do you want to do?

Get the date of the next or previous specified day

Add or subtract a specified number of days to an input date

Add or subtract a specified number of weeks to an input date

Add or subtract a specified number of months to an input date

Add or subtract a specified number of years to an input date

Get the date of the next or previous specified day

To get the date of the next or previous specified day (eg Monday, Tuesday etc) following an input date, you use the Next/Previous Day of the Week function. For example,

**the first Thursday of October = the next Thursday on or after 2009-10-01**

**the last Monday of April = the Monday on or before 2009-04-30**

Add or subtract a specified number of days to an input date

To add or subtract a specified number of days to an input date to get a new date, you use the Add Days function. For example,

**the settlement date for the property = the date 42 days after 2009-04-17**

**the date of the auction listing = the date 9 days before the auction completion date**

Add or subtract a specified number of weeks to an input date

To add or subtract a specified number of weeks to an input date to get a new date, you use the Add Weeks function. For example,

**the end date of the exclusion period = the date 2 weeks after the date of contraction**

**the date the books were borrowed = the date 3 weeks before the due date of the books**

Add or subtract a specified number of months to an input date

To add or subtract a specified number of months to an input date to get a new date, you use the Add Months function. For example,

**the waiting period end date = the date 6 months after 2008-10-16**

**the date the wedding invitations should be sent by = the date 2 months before the wedding date**

Add or subtract a specified number of years to an input date

To add or subtract a specified number of years to an input date to get a new date, you use the Add Years function. For example,

**the warranty expiry date = the date 5 years after the date of purchase**

**the date the application was lodged = the date 2 years before 2006-12-23**

See also:

- [Date function reference](#)
- [Date function rule examples](#)

Find a date in a year

There are functions that you can use to find particular dates in a year.

## What do you want to do?

[Find the first date in the year](#)

[Find the last date in the year](#)

[Find the next instance of the given day/month](#)

[Find the start or the end date for the previous or next UK tax year](#)

Find the first date in the year

You use the Year Start function to return the first date in the year in which the input date falls. For example,

**the start of the relevant year = the first day of the year in which 2000-08-07 falls**

would infer the start of the relevant year to be 01/01/2000.

Find the last date in the year

You use the Year End function to return the last date in the year in which the input date falls. For example,

**the end of the relevant year = the last day of the year in which 2002-03-24 falls**

would infer the end of the relevant year to be 31/12/2002.

Find the next instance of the given day/month

You use the Next Date function to return the next instance of the given day/month. For example,

**the end of the next Australian tax year = NextDate(the current date, 30, 6)**

would infer the end of the next Australian tax year to be 30/6/2010 if the current date is 21/07/2009.

Find the start or the end date for the previous or next UK tax year

You use the UK Tax Year functions to return the start or the end date for the previous or next UK tax year, relative to the input date. (The start date of the UK tax year is 6 April, and the end date is 5 April.) For example,

**the previous UK tax year start date = the previous UK tax year start date on or before 2005-06-01**

**the next UK tax year end date = the next UK tax year end date on or after 2007-11-07**

The first of these rules would infer that the previous UK tax year start date is 06/04/2005, and the second rule would infer that the next UK tax year end date is 05/04/2008.

See also:

- [Date function reference](#)
- [Date function rule examples](#)

Count periods between two dates or times

There are functions that you can use in your rules to count the number of days, weeks, months or years between two input dates, or the number of seconds, minutes or hours between two times.

**What do you want to do?**

[Count the number of weekdays between two dates](#)

[Count the number of whole days between two dates](#)

[Count the number of whole weeks between two dates](#)

[Count the number of whole months between two dates](#)

[Count the number of whole years between two dates](#)

[Count the number of seconds between two times](#)

[Count the number of whole minutes between two times](#)

[Count the number of whole hours between two times](#)

Count the number of weekdays between two dates

To count the number of weekdays between two dates, you use the Weekday Count function. The earlier input date is inclusive and the later input date is exclusive. For example:

**the number of business days in May 2009 = the number of weekdays (inclusive) between 2009-05-01 and 2009-06-01**

would calculate the number of business days in May 2009 to be 21. Note that if the first date in the function is *after* the second date, then the result will be 0.

Count the number of whole days between two dates

To count the number of whole days between two dates, you use one of the day difference functions.

The Day Difference function returns the number of whole days between two dates. This calculation includes only one endpoint. For example:

**the number of days in the billing period = DayDifference(2007-12-01, 2007-12-14)**

would calculate the number of days in the billing period to be 13.

The Day Difference Inclusive function returns the number of whole days (inclusive) between two dates. This calculation includes both endpoints. For example:

**the number of days in the billing period = DayDifferenceInclusive(2007-12-01, 2007-12-14)**

would calculate the number of days in the billing period to be 14.

The Day Difference Exclusive function Returns the number of whole days (exclusive) between two dates. This calculation excludes both endpoints. For example:

**the number of days in the billing period = DayDifferenceExclusive(2007-12-01, 2007-12-14)**

would calculate the number of days in the billing period to be 12.

Date and time values and variables can also be used in these functions.

Note that the order of the two dates (or datetimes) in the function does not affect the result.

### Count the number of whole weeks between two dates

To count the number of weeks between two dates, you use one of the Week Difference functions.

The Week Difference function returns the number of whole weeks between two dates. This calculation includes only one endpoint. For example:

**the number of weeks until the baby is due = WeekDifference(the current date, the baby's due date)**

If the current date is 26/6/2009 and the baby's due date is 25/12/2009, the number of weeks until the baby is due is 26.

The Week Difference Inclusive function returns the number of whole weeks (inclusive) between two dates. This calculation includes both endpoints. For example:

**the number of weeks until the baby is due = WeekDifferenceInclusive(the current date, the baby's due date)**

If the current date is 26/6/2009 and the baby's due date is 25/12/2009, the number of weeks (inclusive) until the baby is due is 27.

The Week Difference Exclusive function returns the number of whole weeks (exclusive) between two dates. This calculation excludes both endpoints. For example:

**the number of weeks until the baby is due = WeekDifferenceExclusive(the current date, the baby's due date)**

If the current date is 26/6/2009 and the baby's due date is 25/12/2009, the number of weeks (exclusive) until the baby is due is 25.

Date and time values and variables can also be used in these functions.

Note that the order of the two dates (or datetimes) in the function does not affect the result.

### Count the number of whole months between two dates

To count the number of months between two dates, you use one of the Month Difference functions.

The Month Difference function returns the number of whole months between two dates. This calculation includes only one endpoint. For example:

**the number of monthly repayments remaining = MonthDifference(2008-01-15, the final payment due date)**

If the final payment due date is 04/09/2009, the number of monthly repayments remaining is 19.

The Month Difference Inclusive function returns the number of whole months (inclusive) between two dates. This calculation includes both endpoints. For example:

**the number of monthly repayments remaining = MonthDifferenceInclusive(2008-01-15, the final payment due date)**

If the final payment due date is 04/09/2009, the number of monthly repayments remaining is 20.

The Month Difference Exclusive function returns the number of whole months (exclusive) between two dates. This calculation excludes both endpoints. For example:

**the number of monthly repayments remaining = MonthDifferenceExclusive(2008-01-15, the final payment due date)**

If the final payment due date is 04/09/2009, the number of monthly repayments remaining is 18.

Date and time values and variables can also be used in these functions.

Note that the order of the two dates (or datetimes) in the function does not affect the result.

### Count the number of whole years between two dates

To count the number of years between two dates, you use one of the Year Difference functions.

The Year Difference function returns the number of whole years between two dates. This calculation includes only one endpoint. For example:

**the person's age = YearDifference(the person's date of birth, the current date)**

If the person's date of birth is 31/10/1910 and the current date is 26/06/2009, the person's age is 98.

The Year Difference Inclusive function returns the number of whole years (inclusive) between two dates. This calculation includes both endpoints. For example:

**the person's age = YearDifferenceInclusive(the person's date of birth, the current date)**

If the person's date of birth is 31/10/1910 and the current date is 26/06/2009, the person's age is 99.

The Year Difference Exclusive function returns the number of whole years (exclusive) between two dates. This calculation excludes both endpoints. For example:

**the person's age = YearDifferenceExclusive(the person's date of birth, the current date)**

If the person's date of birth is 31/10/1910 and the current date is 26/06/2009, the person's age is 97.

Date and time values and variables can also be used in these functions.

Note that the order of the two dates (or datetimes) in the function does not affect the result.

### Count the number of seconds between two times

To count the number of seconds between two times, you use one of the Second Difference functions with date and time inputs.

The Second Difference function returns the number of whole seconds between two datetimes. This calculation includes only one endpoint. For example:

**the number of seconds between first place and second place = SecondDifference(the first place time, the second place time)**

If the first place time is 2008-06-30 09:31:05 and the second place time is 2008-06-30 09:31:10, then the number of seconds between first place and second place is 5.

The Second Difference Inclusive function returns the number of whole seconds (inclusive) between two datetimes. This calculation includes both endpoints. For example:

**the number of seconds between first place and second place = SecondDifferenceInclusive(the first place time, the second place time)**

If the first place time is 2008-06-30 09:31:05 and the second place time is 2008-06-30 09:31:10, then the number of seconds (inclusive) between first place and second place is 6.

The Second Difference Exclusive function returns the number of whole seconds (exclusive) between two datetimes. This calculation excludes both endpoints. For example:

**the number of seconds between first place and second place = SecondDifferenceExclusive(the first place time, the second place time)**

If the first place time is 2008-06-30 09:31:05 and the second place time is 2008-06-30 09:31:10, then the number of seconds (exclusive) between first place and second place is 4.

Note that the order of the two dates (or datetimes) in these functions does not affect the result.

### Count the number of whole minutes between two times

To count the number of whole minutes between two times, you use one of the Minute Difference functions with date and time inputs. The Minute Difference function returns the number of whole minutes between two datetimes. This calculation includes only one endpoint. For example:

**the number of minutes late the plumber is = MinuteDifference(the time the plumber was meant to arrive, the time that the plumber actually arrived)**

If the time the plumber was meant to arrive is 2009-10-18 08:30:00 and the time that the plumber actually arrived is 2009-10-18 09:00:40, then the number of minutes late the plumber is 30.

The Minute Difference Inclusive function returns the number of whole minutes (inclusive) between two datetimes. This calculation includes both endpoints. For example:

**the number of minutes late the plumber is = MinuteDifferenceInclusive(the time the plumber was meant to arrive, the time that the plumber actually arrived)**

If the time the plumber was meant to arrive is 2009-10-18 08:30:00 and the time that the plumber actually arrived is 2009-10-18 09:00:40, then the number of minutes (inclusive) late the plumber is 31.

The Minute Difference Exclusive function returns the number of whole minutes (exclusive) between two datetimes. This calculation excludes both endpoints. For example:

**the number of minutes late the plumber is = MinuteDifferenceExclusive(the time the plumber was meant to arrive, the time that the plumber actually arrived)**

If the time the plumber was meant to arrive is 2009-10-18 08:30:00 and the time that the plumber actually arrived is 2009-10-18 09:00:40, then the number of minutes (exclusive) late the plumber is 29.

Note that the order of the two dates (or datetimes) in these functions does not affect the result.

## Count the number of whole hours between two times

To count the number of whole hours between two times, you use one of the Hour Difference functions with date and time inputs. The Hour Difference function returns the number of whole hours between two datetimes. This calculation includes only one endpoint. For example:

**the number of hours the plane was delayed by = HourDifference(the scheduled arrival time of the flight, the arrival time of the delayed flight)**

If the scheduled arrival time of the flight is 2006-10-13 09:50:00 and the arrival time of the delayed flight is 2006-10-13 11:00:00, then the number of hours the plane was delayed by is 1.

The Hour Difference Inclusive function returns the number of whole hours (inclusive) between two datetimes. This calculation includes both endpoints. For example:

**the number of hours the plane was delayed by = HourDifferenceInclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)**

If the scheduled arrival time of the flight is 2006-10-13 09:50:00 and the arrival time of the delayed flight is 2006-10-13 11:00:00, then the number of hours (inclusive) the plane was delayed by is 2.

The Hour Difference Exclusive function returns the number of whole hours (exclusive) between two datetimes. This calculation excludes both endpoints. For example:

**the number of hours the plane was delayed by = HourDifferenceExclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)**

If the scheduled arrival time of the flight is 2006-10-13 09:50:00 and the arrival time of the delayed flight is 2006-10-13 11:00:00, then the number of hours (exclusive) the plane was delayed by is 0.

Note that the order of the two dates (or datetimes) in these functions does not affect the result.

See also:

- [Date function reference](#)
- [Date function rule examples](#)

## Calculate the number of days in a month

Using a combination of [date functions](#) you can calculate the number of days in a given month.

Basically, you get the first day of the month, add a month to get the first day of the next month, and then get the number of days between them, which will give you the number of days in the month of the specified date ("the date").

The rules are as follows:

**the number of days in the month = DayDifference(the start of the month, the start of the next month)**

**the start of the month = MakeDate(the year, the month, 1)**

**the start of the next month = AddMonths(the start of the month, 1)**

**the year = ExtractYear(the date)**

**the month = ExtractMonth(the date)**

## Find the day from a date

To find the day from a particular date, you can use the [modulo function](#) with a date in the past that was a Monday.

For example, we know that the 7th of January 1980 was a Monday, so every 7th day after that is also a Monday. Subtracting this baseline date from the given date, and then using the modulo operator we can determine a number for each day of the week as follows:

$$\text{the numerical form of the day of the week} = (\text{the date} - 1980-01-07) \text{ modulo } 7$$

This number can then be used in a [rule table](#) to find the day of the week:

the day of the week	
"Monday"	the numerical form of day of the week = 0
"Tuesday"	the numerical form of day of the week = 1
"Wednesday"	the numerical form of day of the week = 2
"Thursday"	the numerical form of day of the week = 3
"Friday"	the numerical form of day of the week = 4
"Saturday"	the numerical form of day of the week = 5
"Sunday"	the numerical form of day of the week = 6
uncertain	otherwise

NOTE: Dates before the baseline date (eg before 1980-01-07) will give a negative modulo (ie Monday 0, Tuesday -6, Wednesday -5, Thursday -4, Friday -3, Saturday -2, Sunday -1). You can either choose to write rules to take this into account, or choose a baseline date so far in the past that it is unnecessary. If you want to take earlier dates into account, you would need a rule table like this:

the day of the week	
"Monday"	the numerical form of day of the week = 0
"Tuesday"	the numerical form of day of the week = 1; or the numerical form of day of the week = -6
"Wednesday"	the numerical form of day of the week = 2; or the numerical form of day of the week = -5
"Thursday"	the numerical form of day of the week = 3; or the numerical form of day of the week = -4
"Friday"	the numerical form of day of the week = 4; or the numerical form of day of the week = -3

<b>"Saturday"</b>	the numerical form of day of the week = 5; or the numerical form of day of the week = -2
<b>"Sunday"</b>	the numerical form of day of the week = 6; or the numerical form of day of the week = -1
<b>uncertain</b>	<b>otherwise</b>

# Data model

## Topics in "Data model"

- [Define a data model](#)
- [Create, modify or delete a properties file](#)
- [Define an entity](#)
- [Define a relationship](#)
- [Choose a name for an entity, relationship or attribute](#)
- [Choose a data type for an attribute](#)
- [Use an attribute in a rule](#)
- [Use an entity or relationship in a rule](#)
- [Rename an entity, attribute or relationship](#)
- [Remove an entity, attribute or relationship](#)
- [Visualize the data model](#)
- [Export or import a data model](#)
- [Check the rulebase against the data model](#)
- [Understand partial knowledge of relationships](#)
- [Understand containment relationships and entity completion](#)

## See also:

- [View list of entities and attributes](#)
- [Find the entity for an attribute](#)
- [View and amend the data model while writing rules](#)

## Define a data model

A data model is defined using one or more properties files in an Oracle Policy Modeling project. These properties files contain the attributes, entities and relationships for the project. Having this information contained in a properties file for the project, rather than in individual Word and Excel documents, eliminates the need to re-add the same attributes, entities and relationships in each rule file.

### Entities

An entity can represent a thing such as a person, a child or a corporation, about which attributes can be collected. An entity can have multiple instances. For example, data can be collected and inferred about more than one child in the same session.

### Relationships

Relationships are the connectors between instances of an entity. By specifying the relationship you are specifying whether an instance of an entity is related to one or more of the instances of another (or even the same) entity group.

## Attributes

An attribute is a single unit of data or fact. An attribute is of a particular data type: boolean, text, number, currency, date, time of day or date and time, and the value of an attribute can be 'known' or 'unknown'. An attribute always belongs to a particular entity even if it is the global entity.

## An example of a data model

Let's say we want to capture the entity relationships for a family who may have children. There may be twins amongst the children. The children may go to school and the siblings may go to the same school or different schools. Each child has friends which may be the same friends as their siblings but they each have a single best friend who they do not share with other siblings.

## Choose entities

We can capture this example using three entities:

- the child
- the friend
- the school

We do not need to create a separate entity for the twin, as we know the twin is one of the children. Similarly, we do not need to create a separate entity for the best friend as we know that the best friend will be one of the child's friends. We do not need to create an entity for the family as we do not need to enter multiple families, so information about the family can be represented in the global entity.

## Choose relationships

[Containment relationships](#) must be defined for each entity. Additional [reference relationships](#) are also defined where required for the data model logic. In our case, we know that:

- a family (represented in the global entity) has children, so there must be a relationship between **the global entity and the child**
- each child has a friend so there must be a relationship between **the child and the friend**
- each child has a best friend, so there must be a more specific relationship between **the child and the friend**
- some children may be a twin so there must be a relationship between **the child and itself**
- children go to school, so there must be a relationship between **the child and the school**

We don't know:

- whether it is important to know, for a particular school, which children go to that school
- whether it is important to know, for a particular friend, which children they are friends with

We can either model these last two relationships 'just in case' or not capture these relationships. We have left these relationships out of this example for simplicity.

## Choose relationship types

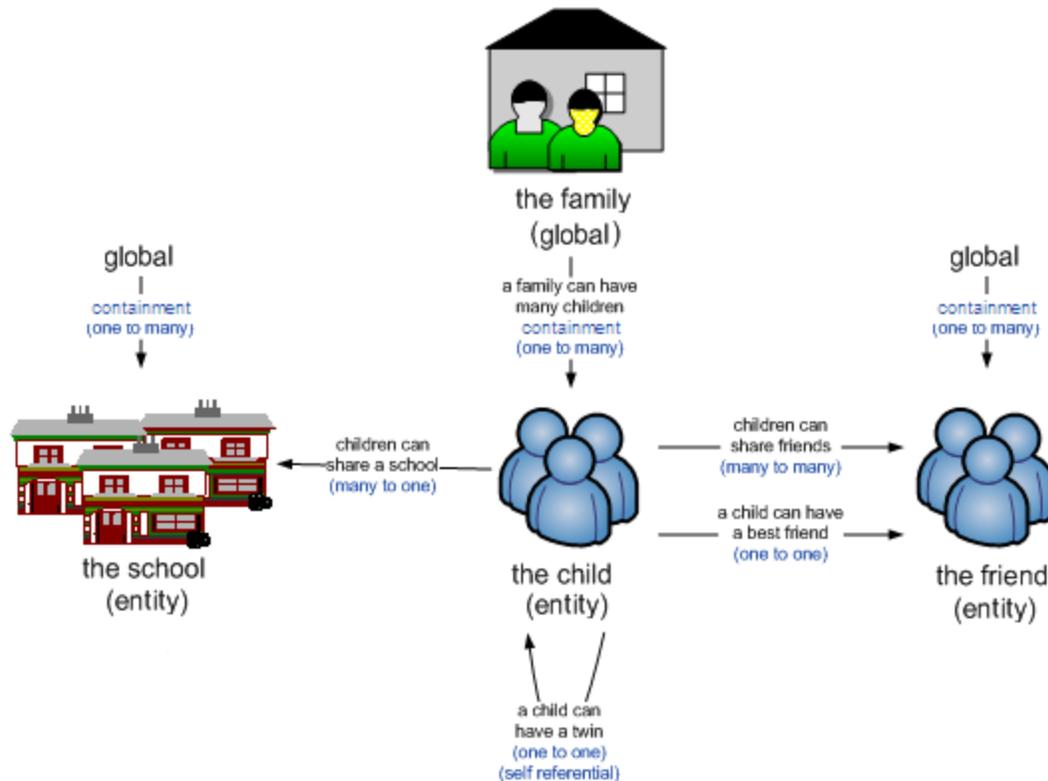
Now we need to decide what type of relationship there is between each of the entities. There are two questions you can ask yourself to help identify the type of relationship. (NOTE: The text in parentheses is the relationship text.) This will also help you see which relationships can be containment relationships (which must be "to-one" relationships to the containing entity).

- Global entity and Child (the children):  
Can more than one family have the same child? No.  
Can there be multiple children in the family? Yes.

Therefore it must be a **one-to-many** relationship.  
This will be the **containment relationship** for the child entity.

- Child and Friend (the child's friends):  
Can the children share friends? Yes.  
Can a child have more than one friend? Yes.  
Therefore it must be a **many-to-many** relationship.
- Child and Friend (the child's best friend):  
Can children share best friends? No.  
Can a child have more than one best friend? No.  
Therefore it must be a **one-to-one** relationship.
- Child and Child (the child's twin):  
Can more than one child share the same twin? No (otherwise it would be a triplet).  
Can a child have more than one twin? No.  
Therefore it must be a **one-to-one** self-referential relationship.
- Child and School (the child's school):  
Can the children share a school? Yes.  
Can a child go to more than one school? No.  
Therefore it must be a **many-to-one** relationship.
- Containment relationships are also required for the Friend and School entities, since the above relationships for these entities are not suitable "to-one" relationships. We would define:
  - Global entity and Friend (the friends): this is a **one-to-many containment** relationship.
  - Global entity and School (the schools): this is a **one-to-many containment** relationship.

The relationships about which we wish to reason therefore look like:



See also:

- [Create a properties file](#)
- [Define attributes](#)
- [Define entities](#)
- [Define relationships](#)

## Create, modify or delete a properties file

Properties files are created and managed in Oracle Policy Modeling. They are the only file type that allows for persistence of the associated data.

Every project should have at least one properties file. This file should contain all the property information for the attributes in the project. This file should also contain all of the data about entities and relationships in the project. It is recommended that this file is named 'Properties' for consistency across projects. It should be created in the Properties folder in Oracle Policy Modeling.

On larger projects with several developers it might make sense to have more than one properties file. For more information, see [Use multiple properties files on a multi-developer project](#).

## What do you want to do?

[Create a properties file](#)

[Modify a properties file](#)

[Delete a properties file](#)

## Create a properties file

To add a new properties file to your project:

1. In Oracle Policy Modeling, select the **Properties** folder in the Project Explorer.
2. Right-click and select **Add New Properties File**. A new properties file will be added to your project. The new file will be selected and highlighted in the list.
3. Type a name for your properties file, for example, "Properties".
4. Save your project by selecting **File | Save All** from the main menu in Oracle Policy Modeling.

## Modify a properties file

To make changes to a properties file:

1. In Oracle Policy Modeling, double-click the properties file in the Project Explorer. The file will open in the right hand pane.
2. Double-click on any item (eg an attribute, an entity, a relationship) to edit it.
3. Save your project by selecting **File | Save All** from the main menu in Oracle Policy Modeling.

## Delete a properties file

To delete a properties file:

1. In the Project Explorer in Oracle Policy Modeling, right-click the properties file and select **Delete**.
2. Click **OK** to confirm the permanent deletion.

TIP: To only remove the file from your Oracle Policy Modeling project (but not delete it from your file system as well), right-click it in Oracle Policy Modeling and select **Remove from Project**.

## Define an entity

An entity is a grouping of things with rules or data in common. An entity often represents a group of people (eg children, applicants, stakeholders) but it can also represent a group of objects (eg textbooks), activities (eg assignments) or concepts (eg school terms).

Entities may be used to allow the same rule to be applied multiple times to make a determination. For example, you may have a rule to say that if any child of the applicant is of school age then the applicant is eligible for a tax rebate. You may collect the details of each of the person's children in order to infer whether each child is of school age, and from that infer whether or not the person is eligible.

Your rules might look something like this:

Rule 1:

**the applicant is eligible for a tax rebate if**  
at least one of the applicant's children is of school age

Rule 2:

**the child is of school age if**  
the child's age > 4

In this situation, the value of "the child's age" (base level) and "the child is of school age" (inferred) may be different for each instance of the child. For example:

Child 1 (Mary)	Child 2 (Darryl)
the child's age = 2	the child's age = 6
the child is of school age = false	the child is of school age = true

These attributes are called entity-level attributes. Rules which use entity-level attributes are called entity-level rules.

A member of the entity group is called an entity instance. In the example above, Mary would be one instance of "the child" entity and Darryl would be another instance of "the child" entity.

## What do you want to do?

[Understand the different types of entities](#)

[Create an entity](#)

[Give an entity a public name](#)

[Understand the different types of entities](#)

There are two types of entities: regular entities and the global entity.

### Entities



An entity can have zero or more entity instances. For example, children in a family, applicants on an application form, taxable events in a tax period. Using entities you can apply the same rules, or collect the same data, for multiple instances of an entity.

An attribute of an entity may hold one value at a time during an investigation for each instance of the entity. That attribute may have as many values as there are instances of the entity, and will only operate within the context of that entity instance.

### Global Entities

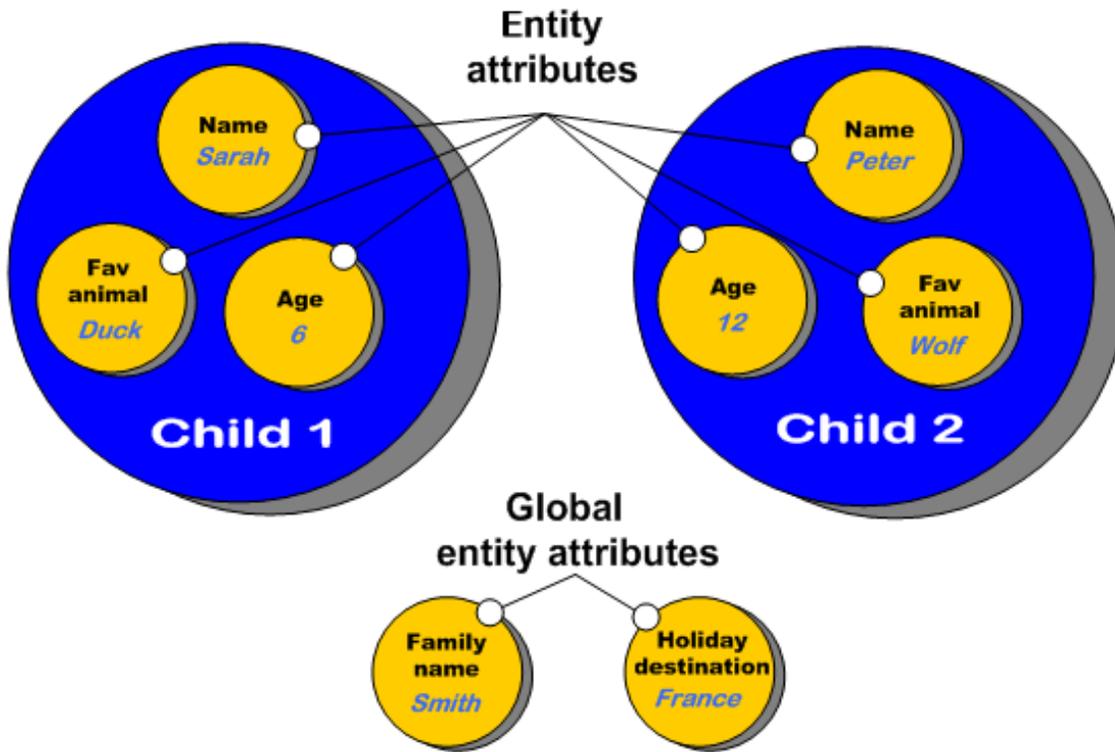


Not all information relevant to your rules may belong to a particular entity. As such, Oracle Policy Modeling has a global entity which acts as a catch-all for information that does not belong to any other entity. For example "the sun is shining" is a global attribute that does not belong to the entity "the family, "the child" and so forth.

An attribute of a global entity may only hold one value at a time during an investigation, and that value persists across the entire rulebase, common to all entities and instances of entities.

The global entity is the default location of attributes. If you do not create entities in your rulebase, or if you create an attribute which does not belong to another entity, the attribute will be stored in the global entity.

The following diagram shows how instances of a child in a family situation have entity attributes:



In the diagram, both children are in the same family and are going to the same holiday destination, but each has their own distinct properties (entity attributes), such as name, age and favorite animal.

### Create an entity

Entities are defined in a properties file for the project, rather than in individual Word and Excel documents. This eliminates the need to re-define the same entities in each rule file.

To add an entity to a properties file:

1. In Oracle Policy Modeling, double-click the properties file in the Project Explorer. The file will open in the right hand pane.
2. Right-click on the Global entity (or other parent entity) in the **Entities** tab and select **New Entity**.
3. Type a name for the new entity, then press **Enter**. TIP: Entities should be named using the definite article 'the', eg 'the family', 'the child', 'the friend', 'the school' etc.

The entity that you have added will now be displayed in the left-hand pane of the properties file:



After you have defined an entity in this way, every attribute which uses the exact entity text (eg 'the child') becomes an entity-level attribute belonging to that entity.

When an entity is created:

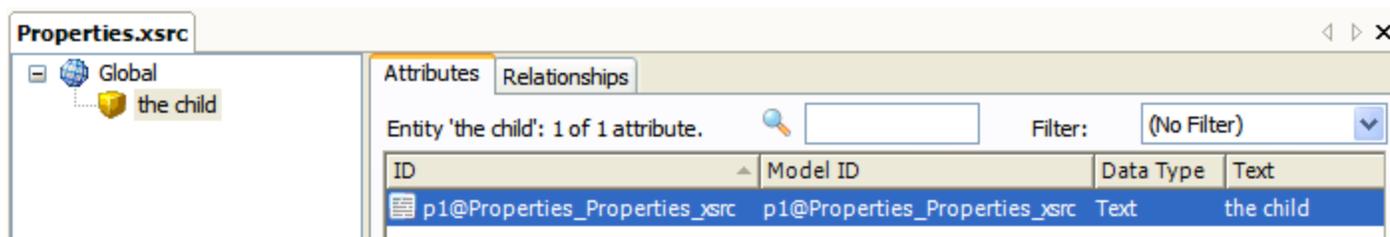
1. a [containment relationship](#) is automatically created for the new entity, to ensure that the containment structure in the rule-base data model is well-defined, and
2. an identifying attribute is automatically created for the new entity (see below).

## Identifying attributes

The identifying attribute for an entity is the text attribute whose value is used for labeling instances of the entity in decision reports, Web Determinations and the debugger. So for example, if you had 'the child' text attribute as the identifying attribute for the entity 'the child', then after you have set values for 'the child' attribute (eg, 'Reid', 'Cohen' and 'Emery'), the child entity instances would be labeled accordingly.



As mentioned earlier, an identifying attribute is automatically created when a new entity is created.



By default, this text attribute has:

- The same text as the entity (eg 'the child').
- No public name defined.
- A gender of Generic (he/she). If the entity is something that does not have a gender, or has a specifically male or female gender, you will need to change this setting. For more information on gender, see [Substitute a gender pronoun for a text variable](#).
- Substitution switched on. You need to consider if this is how you want the attribute to operate. For more information on substitution, see [Substitute the actual value of a variable for its text](#).

If you want to change any of these default settings you will need to edit the attribute. To do this:

1. In Oracle Policy Modeling, double click the properties file in the Project Explorer to open it for editing.
2. In the left hand pane, select the entity that the attribute belongs to.
3. On the **Attributes** tab in the right hand pane, double click the identifying attribute to open the **Attribute Editor**.

4. Make the necessary changes to the attribute.

The screenshot shows the 'Attribute Editor' dialog box for the attribute 'p3' of the entity 'the brand'. The 'Data Type' is set to 'Text' and the 'Text' value is 'the brand'. The 'Error Message' is 'Invalid Value.'. The 'Default gender' is 'Impersonal (it)'. The 'Gender attribute' dropdown menu is open, showing the selected option 'Impersonal (it)' and other options: 'Generic (he/she)', 'Male (he)', and 'Female (she)'. The 'Allow Substitution' checkbox is checked. The 'Question' field contains 'What is the brand?'.

5. Click **OK**.

To change the attribute that is used to identify an entity (or to specify one if there isn't one):

1. In Oracle Policy Modeling, double-click the properties file in the Project Explorer. The file will open in the right hand pane.
2. Double click the entity to open the **Edit Entity** dialog.
3. Click the browse button next to the **Identifying attribute** field.
4. In the **Attribute Selector**, select the appropriate attribute, then click **OK**.
5. Click **OK** again to close the Edit Entity dialog.

### Give an entity a public name

You can define a public name for your entity in the same way as you can define a public name for an attribute of an entity. The public name overrides the entity text. You should define a public name for an entity when the entity name in the data model differs from the entity name in the source material.

To define a public name for an entity, do the following:

1. In Oracle Policy Modeling, double-click the properties file in the Project Explorer. The file will open in the right hand pane.
2. Double click the entity to open the **Edit Entity** dialog.

3. Enter a public name. Then click **OK**.



The screenshot shows a dialog box titled "Edit Entity 'the child'". It has a blue title bar with a close button (X) in the top right corner. The dialog contains four input fields with labels on the left and values in the text boxes:

- Entity text: the child
- Public Name: child
- Identifying attribute: the child (with a list icon to the left and a "..." button to the right)
- Containing entity: global (with a "..." button to the right)

At the bottom, there are two tabs: "Common" (selected) and "Custom Properties". Below the tabs are two buttons: "OK" and "Cancel".

See also:

- [Use an entity in a rule](#)
- [View list of entities and attributes](#)
- [Check attribute entity levels](#)

## Define a relationship

Relationships define how entities relate to one another. All entities must have a [containment relationship](#) defined, which specifies the overall structure of the rulebase. In addition, reference relationships can be defined between entities if appropriate for your data model. You need to have already defined your entities before you can add reference relationships.

By specifying a relationship you are specifying whether an instance of an entity is related to one or more of the instances of another (or even the same) entity group.

For example, if you have:

- An entity 'the person', and
- An entity 'the car', and
- A relationship from the person to the car 'the cars belonging to the person', and
- Instances of the entity 'the person' called "Tom", "Dick" and "Harry", and
- Instances of the entity 'the car' called "VW", "Mazda", "Holden" and "Ford".

Then there is a single relationship, but there are three relationship instances, because each person has a list of the cars that belong to them.

That is, Tom has a VW and Mazda, Dick has a Ford, and Harry has a Holden and Ford.

## What do you want to do?

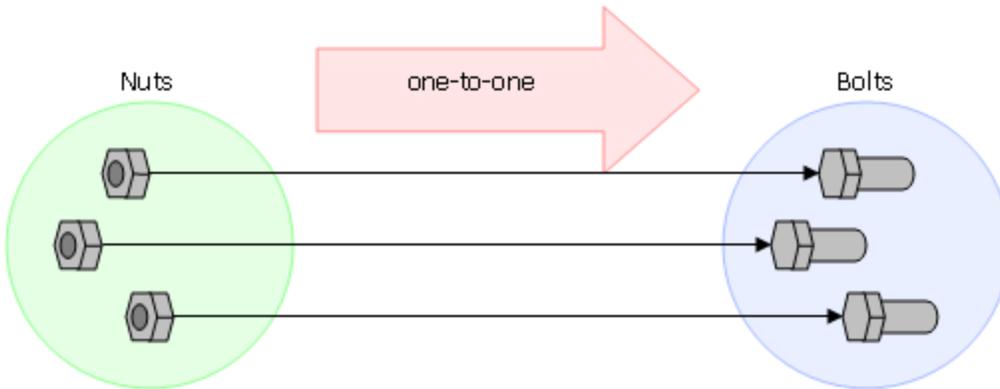
[Understand the different types of relationships](#)

[Create a relationship in a properties file](#)

Understand the different types of relationships

### One-to-One

A one-to-one relationship is where one entity instance interacts only with one other entity instance.



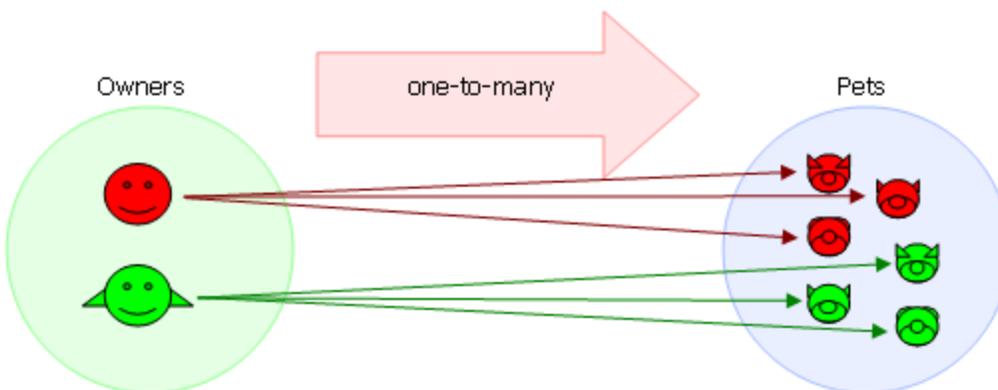
Entity 1: the nut

Entity 2: the bolt

Relationship from the nut to the bolt: the nut's bolt

### One-to-Many

A one-to-many relationship is where one entity instance interacts with many other entity instances. This is the traditional hierarchical model relationship.



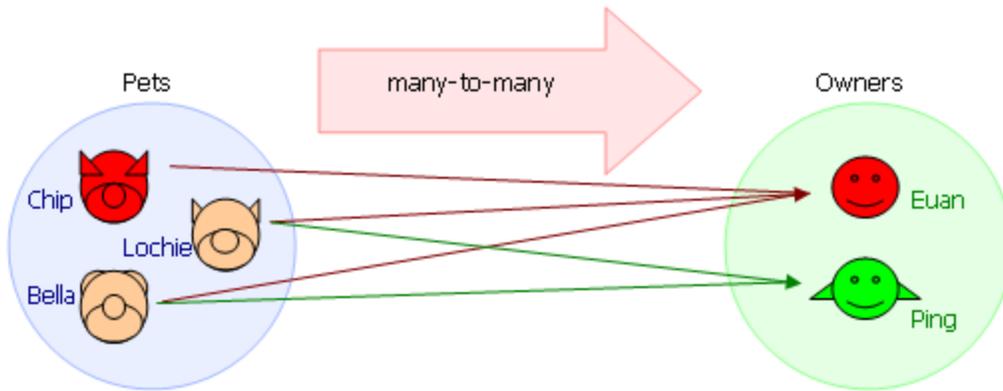
Entity 1: the owner

Entity 2: the pet

Relationship from the owner to the pet: the owner's pets

### Many-to-Many

A many-to-many relationship is where multiple instances can interact with many other entity instances. In the example below, the Pets "Bella" and "Lochie" share the Owners "Ping" and "Euan", whilst Chip has only one owner Euan.



Entity 1: the pet

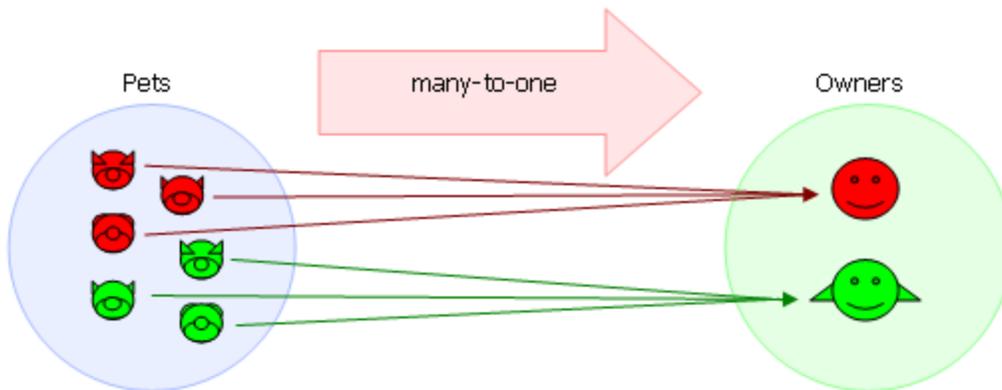
Entity 2: the owner

Relationship from the pet to the owner: the pet's owners

TIP: To see a simple example of a complete rulebase with a many-to-many relationship, open and [run the Parents And Children example rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### Many-to-One

A many-to-one relationship is where many entity instance belong to only one entity instance. This is the reverse of a one-to-many relationship.



Entity 1: the pet

Entity 2: the owner

Relationship from the pet to the owner: the pet's owner

### Reverse Relationships

Reverse relationships occur where an entity has a relationship to another entity, and that entity has a relationship back again. For example, a parent has a child and a child has a parent.

However, not all relationships have a logic reverse and not all relationships require capturing the reverse relationship. For example, it might be useful to collect that an applicant has applied for multiple benefits but there is no need to identify all of the applicants for a particular benefit.

In determining whether or not to capture a reverse relationship, consider whether both directions of the relationship will be useful in your rules. If in doubt, create the reverse relationship - it won't be activated unless you have rules which refer to it.

#### **Primary Direction**

The primary direction of a relationship applies to relationships that also have reverse relationships. The primary direction determines the primary relationship for the pairing of relationship to reverse relationship. This is important for inferred relationships: an inferred relationship may only be proved by a rule in its primary direction.

#### **Self-Referential Relationships**

Sometimes it is necessary to relate one entity instance to another entity instance in the same entity. For example, a child in a family may have a special relationship with another child in the family, such as being twins or sharing a room.

This type of a relationship is treated the same as other relationship types, except that both the source and the target of the relationship are the same entity.

#### **Containment Relationships**

All entities must have a containment relationship defined. The collection of containment relationships that link entities in the rulebase together allows us to see the logical structure of the data model. For example, a rulebase may have two entities 'the guardian' and 'the child', with two containment relationships defined: a one-to-many relationship "the guardians" from the global entity to the guardian entity, and a one-to-many relationship "the guardian's children" from the guardian entity to the child entity. The containment relationship for an entity must be either many-to-one or one-to-one, ie each entity instance must be contained by a single parent entity instance. Additional relationships (reference relationships, see below) between entities can be defined as needed for your rulebase data model. See [Understand containment relationships and entity completion](#) for more information.

#### **Reference Relationships**

Reference relationships define meaningful connections between entities that exist in addition to the entities' containment relationships. For example, an entity 'the person' may have a containment relationship from the global entity called "the people", and an additional reference relationship to capture groups of people who live together called "the person's housemate", which is a self-referential relationship between instances of the person entity.

#### **Inferred Relationships**

An inferred relationship is a many-to-many relationship that has its [membership concluded by rules](#) in the rulebase.

Create a relationship in a properties file

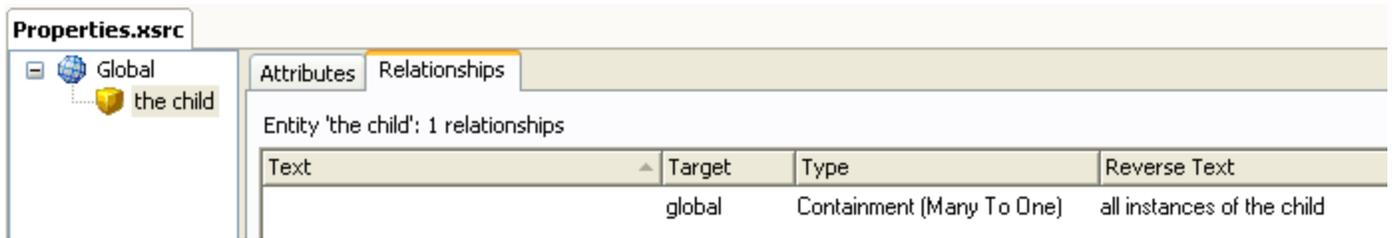
Relationships are defined in a properties file for the project, rather than in individual Word and Excel documents. This eliminates the need to re-define the same relationships in each rule file.

#### **Create a containment relationship**

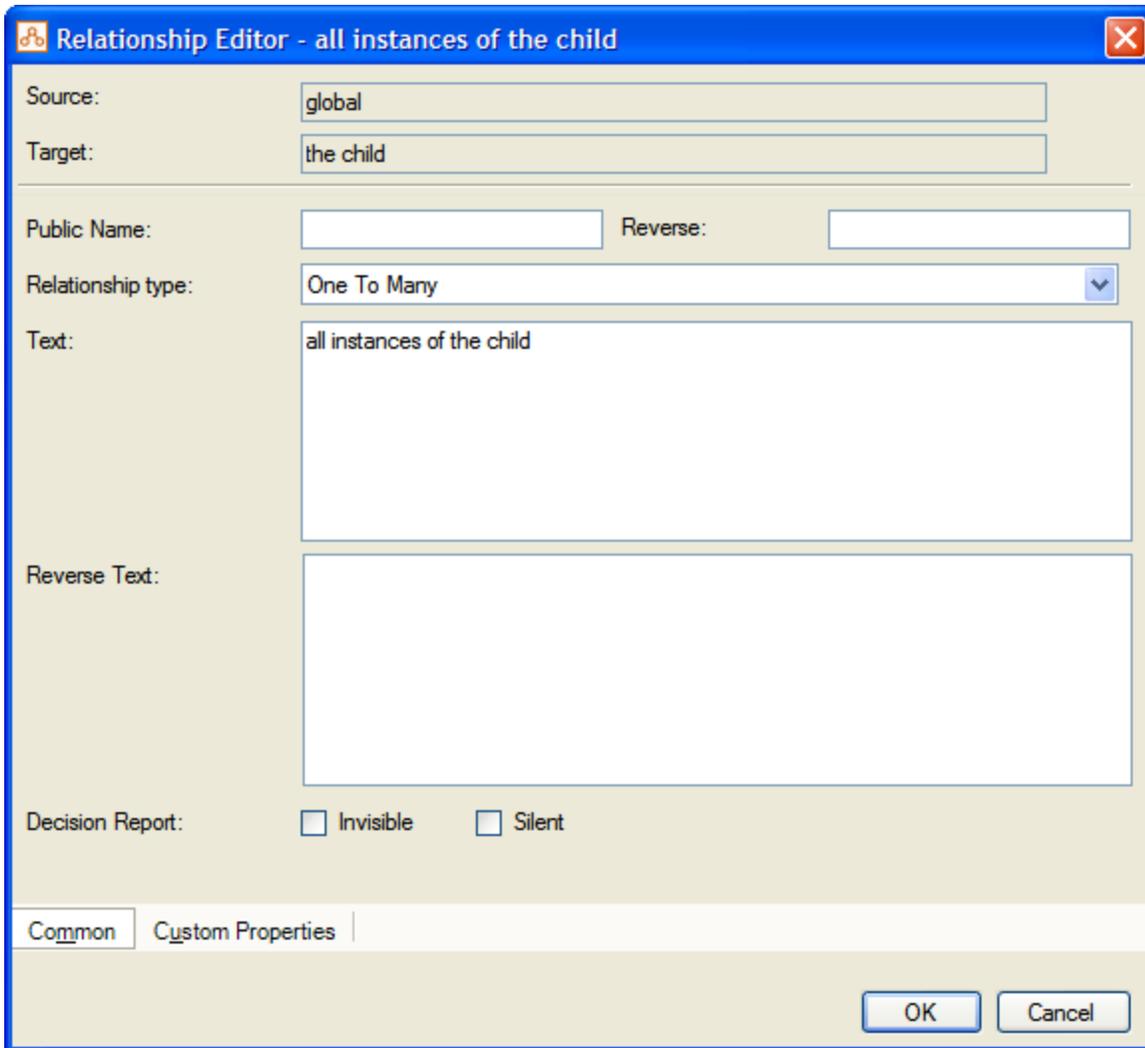
Containment relationships are automatically created when the [associated entities are first created](#) in the rulebase. However, you should provide more meaningful relationship text by editing the relationship before using the relationship in your rules.

To edit a containment relationship:

1. In Oracle Policy Modeling, double click your properties file to open it. Select the entity that you want to edit the containment relationship for. The containment relationships already created for the entity will be shown on the **Relationships** tab.



2. Double click on the relationship to open the **Relationship Editor** dialog box.



3. Change the **Text** for the relationship.

Relationship Editor - all instances of the child

Source: global

Target: the child

Public Name: Reverse:

Relationship type: One To Many

Text: the children

Reverse Text:

Decision Report:  Invisible  Silent

Common Custom Properties

OK Cancel

Click **OK**. You can now use your new relationship text in rules.

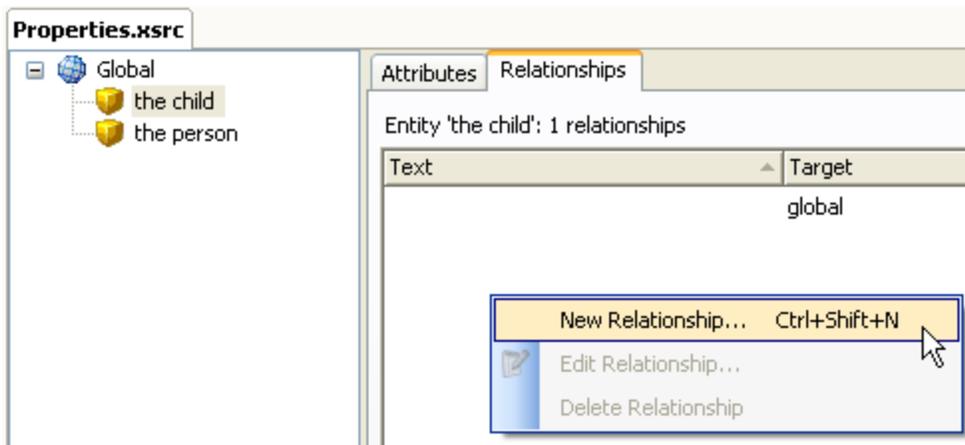
### Create a reference relationship

To create a reference relationship between two entities in a properties file:

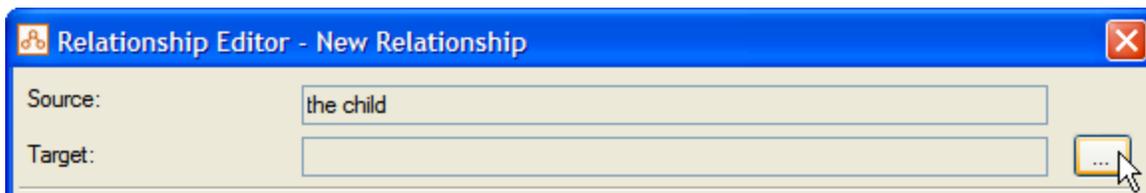
1. In Oracle Policy Modeling, double click your properties file to open it. Select the entity that you want to create a relationship for.



2. In the Relationships window, right-click and select **New Relationship...**



3. In the **Relationship Editor** dialog box, select the browse button next to the **Target** field to select the target entity.



4. In the **Entity Selector** dialog box select the target entity.



Then click **OK**.

5. In the Relationship Editor dialog select the type of relationship from the drop-down menu.

Relationship Editor - New Relationship

Source: the child

Target: the person

Public Name: Reverse:

Relationship type: One To Many

Text:

6. Enter the textual form of the relationship name in the **Text** field. Relationships should be given a meaningful name, usually using the definite article 'the' (for example, 'the children', 'the child's school' etc). For more information, see [Choose a name for a relationship](#).

Relationship Editor - New Relationship

Source: the child

Target: the person

Public Name: Reverse:

Relationship type: Many To Many

Text: the child's parents

7. Enter the textual form of the reverse relationship in the **Reverse Text** field. If a reverse relationship is not specified then by default it is not possible to traverse the relationship backwards.

**Relationship Editor - New Relationship**

Source:

Target:  ...

Public Name:  Reverse:

Relationship type:  ▾

Text:

Reverse Text:

Decision Report:  Invisible  Silent

Common | Custom Properties

OK Cancel

8. Click **OK**. Repeat this process for any additional relationships you want to add. The relationships that you have added will now be displayed on the Relationships tab.

**Properties.xsrc\***

Global

- the child
- the person

Attributes Relationships

Entity 'the child': 2 relationships

Text	Target	Type	Reverse Text
	global	Containment (Many To One)	the children
the child's parents	the person	Many To Many	the person's children

NOTE: For any relationships that you want to export in a module, you need to specify public names on both ends (ie in the **Public Name** and **Reverse** fields in the Relationship Editor).

## Flip the direction of a relationship

The **primary direction** of a relationship is important - you cannot infer a relationship or collect one on a screen from the non-primary direction. The primary direction will be assumed to be the direction in which the relationship was first created. If you want to set it in the reverse direction, you will need to flip it. To do this:

1. In Oracle Policy Modeling, double click your properties file to open it. Select the entity that the relationship relates to.
2. In the Relationships window, double click the relationship to open the **Relationship Editor**.

The screenshot shows the 'Relationship Editor - the child's school' dialog box. It has a blue title bar with a close button. The main area is divided into several sections:

- Source:** the child
- Target:** the school
- Public Name:** childs\_school
- Reverse:** schools\_students
- Relationship type:** Many To One (dropdown menu)
- Text:** the child's school
- Reverse Text:** the school's students
- Decision Report:**  Invisible  Silent

At the bottom right, there is a button labeled 'Flip Primary Direction'. At the bottom left, there are two tabs: 'Common' (selected) and 'Custom Properties'. At the bottom right, there are 'OK' and 'Cancel' buttons.

3. Click the **Flip Primary Direction** button. You will notice that:
  - i. the **Source** and **Target** entities have swapped, and
  - ii. the primary **Public Name** has swapped with the **Reverse** public name, and
  - iii. the **Relationship type** has been reversed, and
  - iv. the primary **Text** and **Reverse Text** have swapped.

Relationship Editor - the child's school

Source: the school

Target: the child

Public Name: schools\_students Reverse: childs\_school

Relationship type: One To Many

Text: the school's students

Reverse Text: the child's school

Decision Report:  Invisible  Silent

Flip Primary Direction

Common Custom Properties

OK Cancel

4. Click **OK**.

## Choose a name for an entity, relationship or attribute

The naming of entities, relationships and attributes is an important consideration when creating a rulebase.

### What do you want to do?

Choose a name for an entity

Choose a name for a relationship

Choose attribute text

Document the naming convention for a project

### Choose a name for an entity

Entities should be named using the definite article 'the', as in 'the family', 'the child', 'the friend', 'the school' etc.

## Choose a name for a relationship

When creating a relationship you should give the relationship a meaningful name. Remember that the relationship describes the reference from one entity instance to one or more of another entity instance. The relationship name should therefore include the target entity text so that it is clear from the relationship name who the relationship is to.

The name of the relationship should reflect the everyday expression used to describe the relationship (if there is one), and should be clear in and out of context what is being referred to. Try to consider that nature of the relationship you are capturing and give it a name that represents this relationship.

Where you are referring to a single instance ("to-one" relationships), your relationship text must therefore be singular. When you are referring to multiple instances ("to-many" relationships), your relationship text must be plural. Where one entity is the global entity, you may simply refer to the target entity.

## Examples of relationship names

Relationship type	Entity 1	Entity 2	Relationship text
One-to-one	"the child"	"the friend"	"the child's best friend"
Many-to-one	"the child"	"the family"	"the child's family"
One-to-many	Global	"the child"	"the children"
Many-to-many	"the child"	"the friend"	"the child's friends"
Self-referential one-to-one	"the child"	"the child"	"the child's twin"

## Choose attribute text

Selecting correct attribute wording is fundamental to capturing logic accurately in your Oracle Policy Modeling application and conveying information to a user in a meaningful way. Specifically, attribute text influences:

- **The logic of a rule condition**

The logic of a rule is not just captured in the rule levels. There is intrinsic logic in the construction of a sentence and the negation of that sentence. For example: "No child appears in the photo" will be negated as "no child does not appear in the photo" which is logically incorrect.

- **The connections between rules**

Rules are connected in the rulebase using plain text matching. A condition of one rule will only be automatically linked to the conclusion of another rule if the text is exactly the same. For example, the text "the doctor's waiting room is full" will not automatically connect to "the doctors' waiting room is full" as the apostrophe is in a different place in the sentence.

- **The display of question text on interview screens**

The user will see the wording of the attribute on any question screens created for the application unless you override this text.

- **The wording of attributes in decision reports**

The decision report is an important mechanism for understanding how the rules are operating. Incorrect attribute text will make it more difficult to debug errors and may mislead or confuse users.

## Choose boolean attribute text

The following general principles apply to the writing of Oracle Policy Modeling boolean attributes.

### 1. Boolean attributes should be complete grammatical sentences

An Oracle Policy Modeling boolean attribute must include at a minimum a subject and verb. The subject is what or who the sentence is about. The verb tells us something about the subject. Most sentences also contain an object which is the thing the action is being performed on.

Examples of grammatical sentences are:

- the investigation continued (subject – verb)
- the lion stalked the gazelle (subject – verb – object)

## 2. Boolean attributes should generally be written in the past tense

The tense of a verb is used to indicate when the action took place. Your top level goal should usually be worded in the present tense as it describes the current state of affairs. However, everything below the top level goal should be written in the past tense as it describes what occurred for the top level conclusion to have been reached.

For example:

- the person is eligible for an award (PRESENT TENSE) if**
  - the person has demonstrated exceptional conduct (PAST TENSE)
- the person has demonstrated exceptional conduct (PAST TENSE) if**
  - the person has been commended by peers (PAST TENSE)

This principle applies regardless of the tense of the source material.

## 3. Boolean attributes should be written in the third person

In English grammar we make a distinction between the speaker/s (I, we), the addressee (you), and the one/thing spoken about (he, she, it, they). This is known as person: first, second and third person, respectively. Boolean attributes should be written in the third person. (Note that there is a mechanism in Oracle Policy Modeling for [switching attribute forms to second person](#) for use in interviews.)

For example:

- the person can go to the movies
- the person has done a good job

Rather than:

- I can go to the movies
- you have done a good job

## 4. Boolean attributes must be able to be negated

Some boolean attributes can be difficult to negate and for this reason should be avoided.

Examples are attributes which use the conjunctions 'and' and 'or'. In these attributes ambiguity can result from the negation of the attribute as we don't necessarily know how the negation of the verb should affect each of the components. For instance, let's look at the attribute "the cat and the dog ate the man's dinner".

If this attribute is false, this could mean that:

- i. neither the cat or the dog ate the man's dinner
- ii. the cat ate the man's dinner but the dog did not
- iii. the dog ate the man's dinner but the cat did not

Given that there are three possible interpretations means that this attribute cannot be negated conclusively and should not be used.

## 5. Boolean attributes should represent a single concept

In many instances, it may be tempting to word an attribute that could be split into two separate clauses as a single attribute. However, if it is likely that part of the attribute is going to be used in other attributes, it is best to separate it into two attributes which each represent distinct concepts.

## 6. Boolean attributes should not use contractions

Contractions are used in more informal styles of writing and speech and should not be used in Oracle Policy Modeling attributes. For example, rather than "there's an application pending", you should write "there is an application pending".

## 7. Boolean attributes should make sense without reference to another attribute

Each boolean attribute should be meaningful without reference to another. To do otherwise makes the rulebase more difficult to develop, maintain and audit.

The following are examples of attributes which do not make sense in isolation:

- This section has been satisfied
- That discussion was recorded
- The person qualifies for the reasons above
- The latest of these two dates applies

## 8. Boolean attributes should be kept simple but explicit

The wording of the attribute should be as simple as possible while still retaining its full intended meaning.

## 9. Boolean attributes should indicate entity membership

If the attribute belongs to an entity, the exact text of the entity should be included in the attribute text to make it clear which entity it belongs to. For example, if you have an entity 'the child', then attributes which belong to that entity group should include the text "the child":

the child is happy  
the child's toy is educational  
the birthdate of the child

## 10. Boolean attributes should not use personal pronouns

A variable can be replaced with the appropriate pronoun the second (and any subsequent times) the variable is used in a boolean attribute. For example, if we had a variable 'the claimant' we could write a boolean attribute 'the claimant owns the claimant's home' and then once we know the name and gender of the claimant this would be rendered as 'John owns his home'. This is preferable to hard-coding "his/her" or "their" in the attribute text.

## 11. Boolean attributes which refer to amounts should indicate the unit of measurement

Boolean attributes which refer to amounts should specify the unit of measurement to avoid any ambiguity. For example:

the person was 100 feet from the scene of the crime

See also:

- [Basic English grammar](#)

## Choose non-boolean attribute text

When creating non-boolean attributes (variables) the following guidelines apply:

### 1. Non-boolean attributes need to start with the definite article 'the'

The definite article 'the' is used to refer to some specific thing (in contrast to the indefinite article 'a' or 'an' which does not refer to

one specific thing). As variables are always referring to a particular thing, they must start with 'the'. For example,

*the claimant's name*

*the type of animal*

*the price of the car*

## 2. Non-boolean attributes should indicate entity membership

If a variable belongs to an entity, the text of the entity should be included in the variable text to make it clear which entity it belongs to. For example, if you have an entity 'the child', then variables which belong to that entity group should include the text "the child":

*the child's age*

*the child's date of birth*

*the school that the child attends*

## 3. Non-boolean attributes which refer to amounts should indicate the unit of measurement

To make it clear what unit of measurement is expected for amount variables, this should be included in the variable text. For example:

*the distance between home and work (kilometers)*

*the weight of the truck (tonnes)*

## 4. Non-boolean attributes should reference their source

References to values derived in other sections of the material should explicitly state the origin of these values in the variable text.

Document the naming convention for a project

A Rulebase Naming Conventions document should be created at the start of every Oracle Policy Modeling project to clearly set out a consistent method of wording attributes. This is critical because automatic linking will only work when attributes are an exact text match. If different rule developers use different text when creating separate chunks of rules the attributes will not tie together. The Rulebase Naming Conventions document should define which nouns will be capitalized and whether particular acronyms should be used.

The Rulebase Naming Conventions document can be kept in the Oracle Policy Modeling project under **Documents/Design**.

## Choose a data type for an attribute

When you create a new attribute you need to define the type of attribute it is, based on the kind of information it represents.

The table below shows the types of attributes that are supported in Oracle Policy Modeling:

Attribute type	Icon	When used	Example
Boolean		for statements	the claimant is eligible for family benefits
Currency		for amounts of money	the claimant's annual income
Number		for any type of number	the claimant's age
Text		for text strings	the claimant's name
Date		for date values	the claimant's date of birth

Attribute type	Icon	When used	Example
Date and time		when a date and time together is needed	the date and time of the car accident
Time of day		for times of day	the store's opening time

Note that for datetime and time of day attributes, you have the option in the Attribute Editor to specify whether seconds will be displayed. If 'Display seconds' is unchecked, any seconds values entered in Web Determinations will be discarded.

The format that values of non-boolean attributes (variables) must take in rules is specified in [Use constant values in rules](#).

The format that values of attributes must take when being entered into input fields, and the format as they appear in decision reports, is specified in [Formatting of attribute values](#).

## Use an attribute in a rule

Variable attributes should be [added](#) in a properties file before being used in rules in Word and Excel. (There is no need to explicitly add boolean attributes before using them in rules.)

To use an existing attribute from your [data model](#) in your rule you simply need to ensure that the exact attribute text is used (ie the same text as in the properties file where the attribute has been added). When the attribute text is exactly the same (including capitalization), the Oracle Policy Modeling compiler recognizes that the text is the same and labels the attributes the same accordingly. Note that you can use the negative form of the attribute and the compiler will recognize it as the negative form of the same attribute.

The easiest way to ensure you are using the same text of an attribute is to use the copy-paste function in Word or 'drag and drop' the text of the attribute from the **Data Model Browser** which is accessed via the Oracle Policy Modeling toolbar in Word.

See also:

- [Use variables in rules](#)
- [Write rules in Word](#)
- [Define decision tables in Excel workbooks](#)

## Use an entity or relationship in a rule

To write rules in Oracle Policy Modeling you need to understand how to refer to the different parts of the data model within your rules.

### What do you want to do?

[Refer to entities connected by a to-many relationship](#)

[Refer to entities connected by a to-one relationship](#)

[Compare entities within the same relationship](#)

[Count the number of instances of an entity](#)

[Get the highest/most recent value of an entity-level variable](#)

[Get the lowest/least recent value of an entity-level variable](#)

Add up numerical values gathered from each instance of an entity

Refer to entities connected by a to-many relationship

Anytime you refer from one entity to another entity in a "to-many" relationship, you need to indicate whether one or all members of the target entity group need to satisfy the rule.

Consider the following rule:

A family may board the plane first if their child is under 8 years of age

We know that families can have more than one child, however, this rule does not specify whether one or all of the family's children must be under 8 years of age in order for the family to board the plane first. If the family had 2 children, one aged 4 and one aged 16, how would you decide?

The rule would be clearer if written in such a way that the reader can tell whether the rule applies to one or all children. For example:

A family may board the plane first if they have at least one child under 8 years of age

We use quantifiers, which are a type of Oracle Policy Modeling syntax, to write these kinds of rules. Quantifiers are operators which access data across the instances of an entity. The two quantifiers we use are:

- the universal quantifier - used to check that the condition returns true for every instance of an entity. For example, "All of the apples are red".
- the existential quantifier - used to check that the condition returns true for at least one instance of an entity. For example, "At least one of the bananas is yellow".

### Check that a condition returns true for every instance of an entity

The universal quantifier must be used when you refer from one entity to another entity in a "to-many" relationship, AND you need to determine whether all members of the target entity group need to satisfy the rule. This quantifier works in much the same way across entities as the 'and' operator does across attributes. This means that the rule using the universal quantifier will only evaluate to true when the condition is true for all instances of an entity. In other words, the conclusion will evaluate to false if its condition is false for one of the targets of the relationship provided. This applies even when the relationship provided is only **partially known**.

There are two types of entity function that are used as universal quantifiers: the For All function and the For All Scope function. This section describes the use of the For All function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the For All Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the For All function is used where there is only one condition. For example, you could have the following rule where 'the family' is an entity (the source entity), 'the child' is an entity (the target entity), and 'the family's children' is the relationship between the entities (the relationship text).

**the family is ready to travel overseas if**

ForAll(the family's children, the child has a passport)

There are several ways of writing a For All function - see the [Entity and relationship function reference](#) for more detail.

Note that if there are zero instances of the entity, then the rule using the For All operator will evaluate to true.

### **Check that a condition returns true for at least one instance of an entity**

The existential quantifier must be used when you refer from one entity to another entity in a "to-many" relationship, AND you need to determine whether any members of the target entity group need to satisfy the rule. This quantifier works in much the same way across entities as the 'or' operator does across attributes. This means that only one instantiation of the entity must be true for the attribute using the operator to be true. In other words, the conclusion will evaluate to true if its condition is true for one of the targets of the relationship provided. This applies even when the relationship provided is only [partially known](#).

There are two types of entity function that are used as existential quantifiers: the Exists function and the Exists Scope function. This section describes the use of the Exists function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the Exists Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the Exists function is used where there is only one condition. For example, you could have the following rule where 'the family' is an entity (the source entity), 'the child' is an entity (the target entity), and 'the family's children' is the relationship between the entities (the relationship text).

#### **the family is eligible for the benefit if**

Exists(the family's children, the child is a qualifying child)

There are several ways of writing an Exists function - see the [Entity and relationship function reference](#) for more detail.

Note that if there are zero instances of the entity, then the rule using the Exists operator will evaluate to false.

### **Refer to entities connected by a to-one relationship**

When you refer from one entity to another entity in a "to-one" relationship that is not a containment relationship, you need to use a particular syntax to connect the two entities together. There are two types of entity functions used for this purpose: the For function and the For Scope function. This section describes the For function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the For Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the For function is used where there is only one condition. For example, you could have the following rule where 'the child' is an entity (the source entity), 'the school' is an entity (the target entity), and 'the child's school' is the many-to-one relationship between the entities (the relationship text).

#### **the child has a day off school if**

For(the child's school, the school is closed)

There are a couple of ways of writing a For function - see the [Entity and relationship function reference](#) for more detail.

#### **NOTES:**

- i. The For syntax can also be used for many-to-many relationships. (The only relationship type that it can't be used with is one-to-many.)

- ii. The For syntax does not need to be used when referring to a parent relationship in the entity's containment relationships. For example, if an entity 'the pet' is contained within an entity 'the child', you could write the following rule without needing to refer to the containment relationship explicitly:

**the pet is playing outside if**

the child is playing outside

### Compare entities within the same relationship

To compare entities within the same relationship, you need to add an alias to the entities involved. Aliasing allows you to provide an alternative name used to refer to an entity instance. For more information, see [Remove ambiguity when reasoning about more than one instance of the same entity](#).

### Count the number of instances of an entity

To count the number of instances there are of an entity, you use the Instance Count function. The syntax for this function is:

- InstanceCount(<relationship text>)
- the number of <relationship text>

For example, the Instance Count function could be used to determine the number of children belonging to the claimant:

**the number of children that the claimant has = InstanceCount(the claimant's children)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant', an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 4 for the following data:

the child
Anthony
Peter
Rebecca
Fiona

### Get the highest/most recent value of an entity-level variable

To obtain the highest or most recent value of an entity-level variable for all instances of the entity, you use the Instance Maximum function. The syntax for this function is:

- InstanceMaximum(<relationship text>, <entity-level variable>)
- the greatest of <entity-level variable> for all of <relationship text>
- <entity-level date> which is the latest for all of <relationship text>

- the latest of all <entity-level date> for <relationship text>
- <entity-level variable> which is the greatest for all of <relationship text>

For example, the Instance Maximum function could be used to determine the highest bank balance for a child of the claimant:

**the highest bank balance for a child of the claimant = InstanceMaximum(the claimant's children, the child's bank balance)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant', an entity 'the child' and a one-to-many relationship 'the claimant's children'.

The function returns a value of \$175 for the following data:

the child	the child's bank balance
Annabel	\$50
Katrina	\$175
Mike	\$120

Get the lowest/least recent value of an entity-level variable

To obtain the lowest or least recent value of an entity-level variable for all instances of the entity, you use the Instance Minimum function. The syntax for this function is:

- InstanceMinimum(<relationship text>, <entity-level variable> )
- the least of <entity-level variable> for all of <relationship text>
- <entity-level variable> which is the least for all of <relationship text>
- <entity-level date> which is the earliest for all of <relationship text>
- the earliest of all <entity-level date> for <relationship text>

For example, the Instance Minimum function could be used to determine the lightest of the claimant's children:

**the lightest weight for a child of the claimant = InstanceMinimum(the claimant's children, the child's weight in kilograms)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant', an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 15 for the following data:

the child	the child's weight in kilograms
Harry	15

the child	the child's weight in kilograms
Sharon	30
Fran	45

Add up numerical values gathered from each instance of an entity

To obtain the sum of all instances of an entity-level variable, you use the Instance Sum function. The syntax for this function is:

- InstanceSum(<relationship text>, <entity-level variable>)
- <entity-level variable> totaled for all of <relationship text>

For example, the Instance Sum function could be used to determine the total Child Care Benefit payable to the claimant:

**the total Child Care Benefit payable to the claimant = InstanceSum(the claimant's children, the Child Care Benefit amount for the child)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant', an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of \$900 for the following data:

the child	the Child Care Benefit amount for the child
Mary	\$500
Sam	\$250
Lizzie	\$150

See also:

- [Reason across multiple entities](#)
- [Entity and relationship function reference](#)
- [Entity and relationship function rule examples](#)

## Rename an entity, attribute or relationship

If you need to rename an entity, attribute or relationship you need to ensure that the change is made throughout the rule project.

### Rename an attribute

To rename an attribute:

1. In Oracle Policy Modeling, select **View | Build Model**.
2. In the Build Model view, find the attribute whose text you want to change in the Attributes tab.
3. Right-click the attribute and select **Change Text Globally**.
4. Click **Replace**.

## Rename an entity

To rename an entity:

1. In the properties file for your project, double-click the entity to open the **Edit Entity** dialog box.
2. Change the **Entity Text**, then click **OK**.
3. You will be asked to confirm that you want to update the text of the attributes belonging to this entity. Select **Yes**.

Any attributes in your property file, including the identifying attribute, will have their text updated to use the new entity name. Note that all other entity-level attributes in your rulebase will need to be manually updated. To do this, open each rules file and do a "find and replace" to change the old entity text to the new entity text.

## Rename a relationship

To rename a relationship:

1. In the properties file for your project, select the source entity for the relationship and then click on the **Relationships** tab.
2. Double click on the relationship to open the **Relationship Editor** dialog box.
3. Change the **Text** for the relationship, then click **OK**.

You now need to update the relationship text in your rules. The easiest way to identify where these changes need to be made is simply to re-compile each rules document and fix the errors that the compiler highlights.

## Remove an entity, attribute or relationship

You may from time to time want to remove an attribute, entity or relationship that is not used or is no longer needed.

### Remove an attribute

To remove an attribute:

1. In Oracle Policy Modeling, select **Tools | Clean Up Unused Attributes and Relationships**.
2. In the **Clean Up Unused Attributes and Relationships** dialog box, select the unused attributes that you want to delete.
3. Click **OK**.

### Remove an entity

To remove an entity:

1. In the properties file for your project, select the entity that you want to remove.
2. Right-click the entity and select **Delete Entity**. Click **Yes** when asked to confirm the deletion.
3. Re-compile all rule documents in the project.

NOTE: If there were attributes using the entity that has been deleted, those attributes will now be global.

### Remove a relationship

To remove a relationship:

1. In Oracle Policy Modeling, select **Tools | Clean Up Unused Attributes and Relationships**.
2. In the **Clean Up Unused Attributes and Relationships** dialog box, select the unused relationships that you want

to delete.

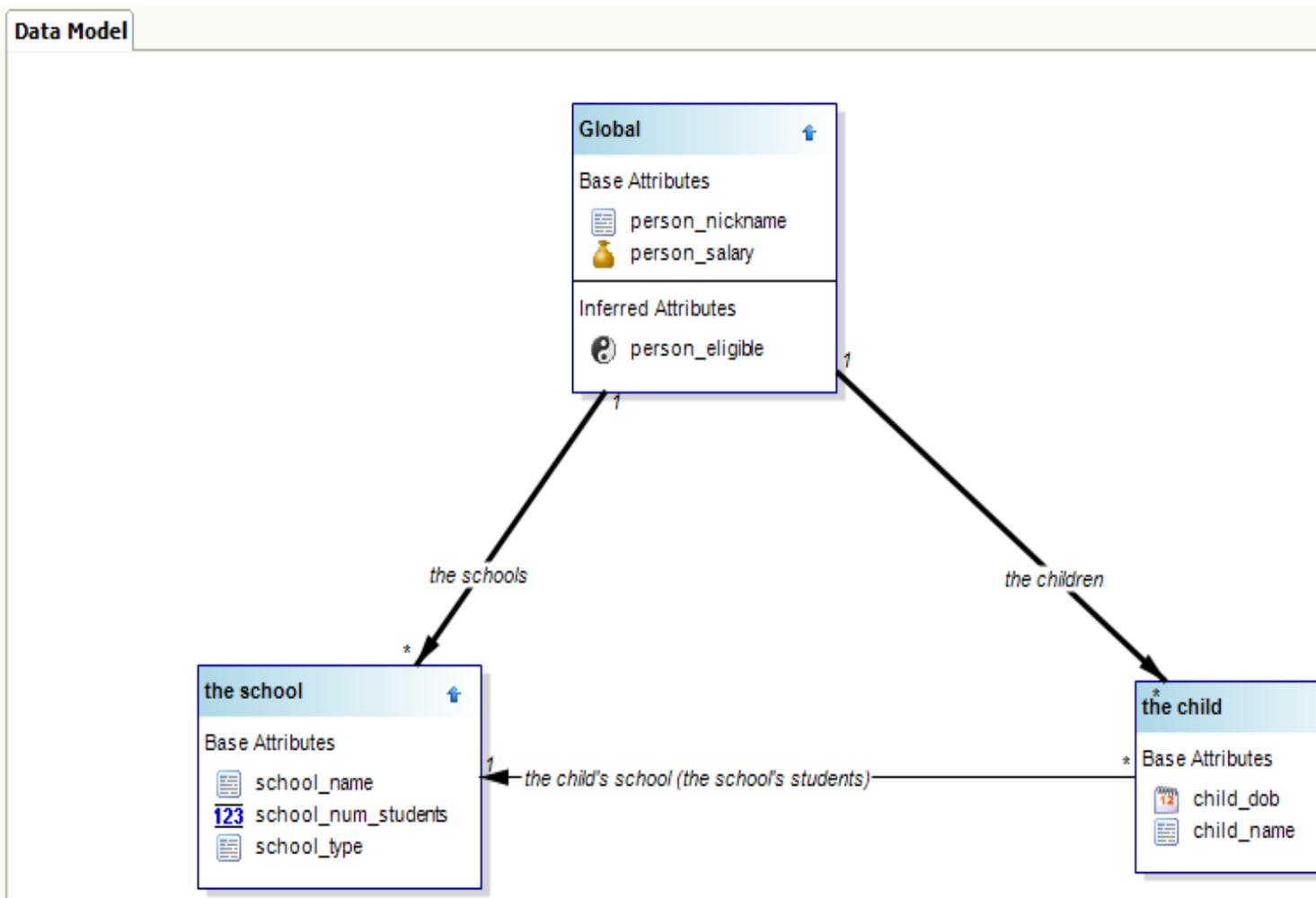
3. Click **OK**.

## Visualize the data model

A data model diagram can be created in Oracle Policy Modeling once you are happy with all of your base level attributes.

To create a data model diagram:

1. In Oracle Policy Modeling, select **View | Data Model**. The data model view will open in the right hand pane.



Each entity in the rulebase is displayed as a separate box with base-level and inferred attributes belonging to that entity displayed in the box.

Relationships between entities are shown as connectors between the entity boxes: a bold connector indicates a containment relationship, a plain connector indicates a reference relationship, and a dashed connector indicates an inferred relationship.

Relationship names are shown on the connector, including a reverse relationship name in brackets if defined.

The relationship type is also shown (eg one-to-many, many-to-many, etc): 1 at the end of the connector indicates a 'to one' relationship, \* indicates a 'to many' relationship.

2. To save this diagram, right-click anywhere in this view and select **Save...**
3. In the **Save As** dialog box specify a location to save the data model to. This process will save the data model diagram in a .wmf picture file format.

**See also:**

[Define an entity](#)

[Define a relationship](#)

## Export or import a data model

Using XML files you can import and export data models to and from Oracle Policy Modeling.

### What do you want to do?

[Export the data model to XML](#)

[Import an existing project using XML](#)

[Import and export a project to and from an external rules repository](#)

#### Export the data model to XML

You can export the data model to integrate with your deployment environment. For example, to show what attributes and entities you have in the rulebase so those entities and attributes can be mapped to the data model of whatever is sending/receiving data to/from the rulebase at runtime.

To export the data model to XML:

1. In Oracle Policy Modeling, select **View | Data Model**. The data model view will open in the right hand pane.
2. In this view, right-click anywhere and select **Export to XML...**
3. In the **Save As** dialog box specify a location to save the data model to. This process will save the data model in the XML format recognized by Oracle Policy Modeling.

#### Import an existing project using XML

A new project can be created in Oracle Policy Modeling by importing an existing project interchange file. To import an existing project in this way:

1. In Oracle Policy Modeling, select **File | Import Project**.
2. In the **Import Project** dialog box, specify the project interchange XML file to import from in the **Interchange file** field.
3. In the **Project folder** field, specify a folder to contain the Oracle Policy Modeling project files. The specified folder should be the Development folder for the imported project, such as "C:\Projects\MyImportedProject\Development". (If the original project contains modules, the imported project should be saved at the same directory level relative to the module as the original project.)
4. Click **Create** to create your project.

Your imported project will open in Oracle Policy Modeling.

## Import and export a project to and from an external rules repository

Oracle Policy Modeling supports the import and export of business rules and associated data and metadata using an intermediate XML file format. The integrity of the content is preserved as it is moved from the external rules repository into Oracle Policy Modeling and back out again. You can view and report on this material in both participating environments.

The steps in this process are given below:

1. Convert rules, data and metadata in the external repository to the standard Oracle Policy Modeling project interchange format.
2. Import the project into Oracle Policy Modeling. The project will be seeded with the various project folders and documents based on the data in the project interchange file. For more information, see [Seeded data in imported projects](#).
3. View and report on the project in Oracle Policy Modeling as necessary. Note that you cannot make changes to the external data model or to the project in Oracle Policy Modeling.
4. Export the project to the standard Oracle Policy Modeling project interchange format.
5. Upload the content of the file to the external repository.

External rulebase data integrators are responsible for steps 1 and 5.

### Export a project to an external rules repository

The contents of an Oracle Policy Modeling project can be exported to an external rules repository using a project interchange file.

To export a project:

1. Select **File | Export...** in Oracle Policy Modeling. The progress of the export process will be shown in the Output window in Oracle Policy Modeling.
2. Click **Export** to export the Oracle Policy Modeling project. NOTE: References to module files are exported, as are the data model elements defined in a module. Rules defined in a module are not exported. For more information on what is re-imported in relation to modules, see [Seeded data in imported projects](#).
3. In the **Export Project** dialog box, browse to the folder or drive where you want to save the file and type a name for the project interchange file. By default this will be the name of the project.
4. Click the **Save** button to save and export the file.

## Check the rulebase against the data model

Once you have created the external data model, you can check that every base level attribute in the project has an attribute with the same ID, data type and entity level in the external data model. Validating against the data model will also check to ensure all base attributes have public names, thus ensuring attribute IDs are reliable and static. Applying data model constraints are turned off by default.

### What do you want to do?

[Check the rulebase against an external data model](#)

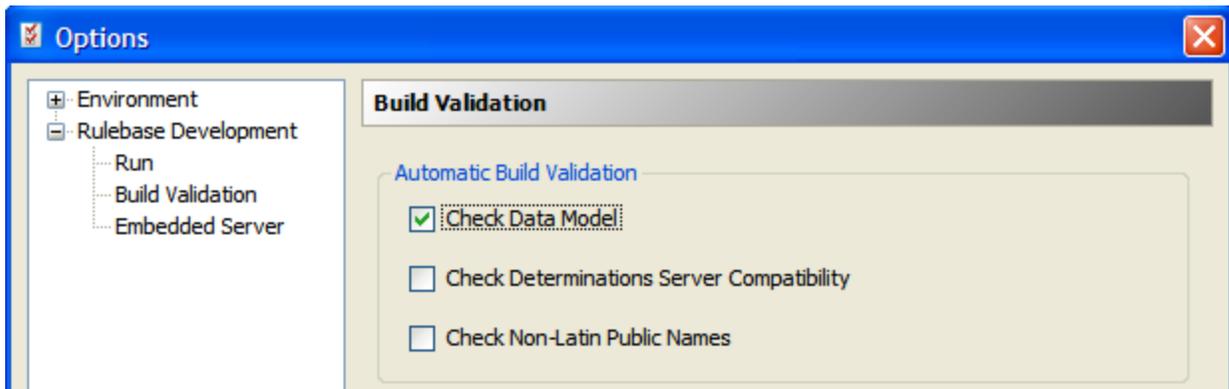
[Create an external data model file for use with Oracle Policy Modeling](#)

[View the data model](#)

Check the rulebase against an external data model

To check the rulebase against an external data model:

1. In the Project Explorer in Oracle Policy Modeling, select the project name, right-click and select **Add Existing File**.
2. Browse to and select your external data model file. Click **Open**. NOTE: The external data model file needs to be in a specific format. For more information, see [Create an external data model file for use with Oracle Policy Modeling](#) below.
3. From the main menu in Oracle Policy Modeling, select **Tools | Options | Rulebase Development | Build Validation** and then select the option **Check Data Model**.



The validation will now be performed when the rulebase is built.

NOTE: The validation setting is a user-specific setting and will need to be performed on every developer's machine.

### Create an external data model file for use with Oracle Policy Modeling

An external data model file can either be created by Oracle Policy Modeling or it can be created elsewhere.

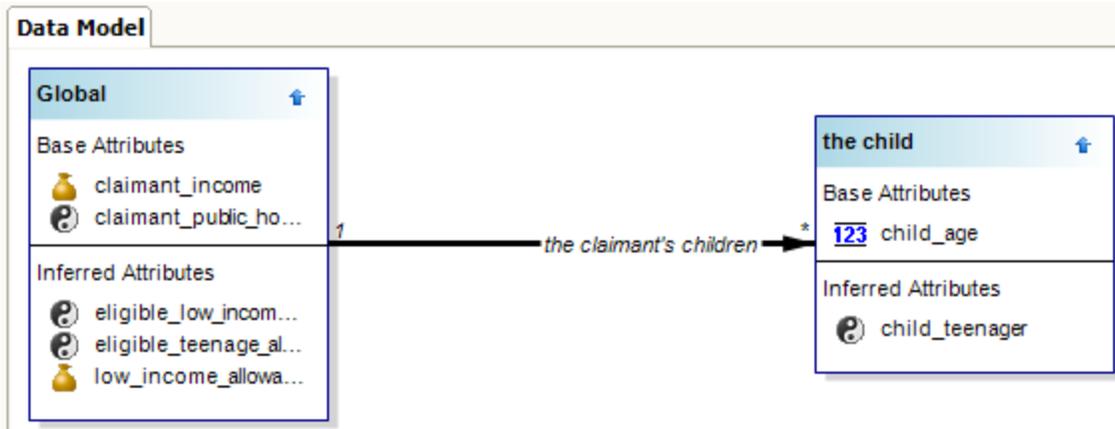
Creating the data model file from within Oracle Policy Modeling requires having a rulebase which already reflects the desired data model. This means creating all the entities and attributes you need in a blank new project or, if the rules are already quite well progressed, making sure the rulebase you are working with already has the desired data model. You then [export the data model to XML](#) in the usual way, to create an external data file that you can use with Oracle Policy Modeling.

If the data model is not created by Oracle Policy Modeling, then the file needs to be transformed into the correct format. See [Oracle Policy Automation Developer's Guide](#) for more information.

### View the data model

The Data Model view in Oracle Policy Modeling shows the rulebase data model.

To open the Data Model view select **View | Data Model**. The Data Model view will open in the top right hand pane of Oracle Policy Modeling.



Each entity in the rulebase is displayed as a separate box with base-level and inferred attributes belonging to that entity displayed in the box. Relationships between entities are shown as connectors between the entity boxes.

### Understand partial knowledge of relationships

In some situations it is possible to draw valid conclusions where attributes or relationships used in a rule have an unknown value, as enough information is known to still make a decision.

Partially complete relationships are those for which some, but not all of the targets are known. Because all the targets are not known, such a relationship is marked as unknown. Both inferred and static relationships can be partially known.

### What do you want to do?

[Understand how partial knowledge reasoning works](#)

[Make a partially known relationship known in the debugger](#)

[Understand how partial knowledge reasoning works](#)

### Partial knowledge of inferred relationships

An inferred relationship will be partially known if the rule used to infer it returns unknown for some, but not all, of the potential target entity instances. Take the following example rule where 'the customer' entity has a many-to-many relationship ('the customer's triple A products') to 'the product' entity.

**the product is a member of the customer's triple A products if**  
 the product's rating = "AAA"

If we have the following entity instances:

- Customer "customer0"
- Product "product0" with rating "AAA"
- Product "product1" with rating "BBB"
- Product "product2" with unknown rating

then when the above rule is evaluated to infer customer0's "the customer's triple A products" relationship, the rule will return true for product0, false for product1, and unknown for product2. Hence product0 is a target of the relationship, and product1 is not a

target of the relationship. product2 on the other hand may or may not be part of the relationship - this cannot be determined until its rating is provided.

Therefore customer0's "the customer's triple A products" relationship is partially known, with one known target (product0), and one known "not target" (product1).

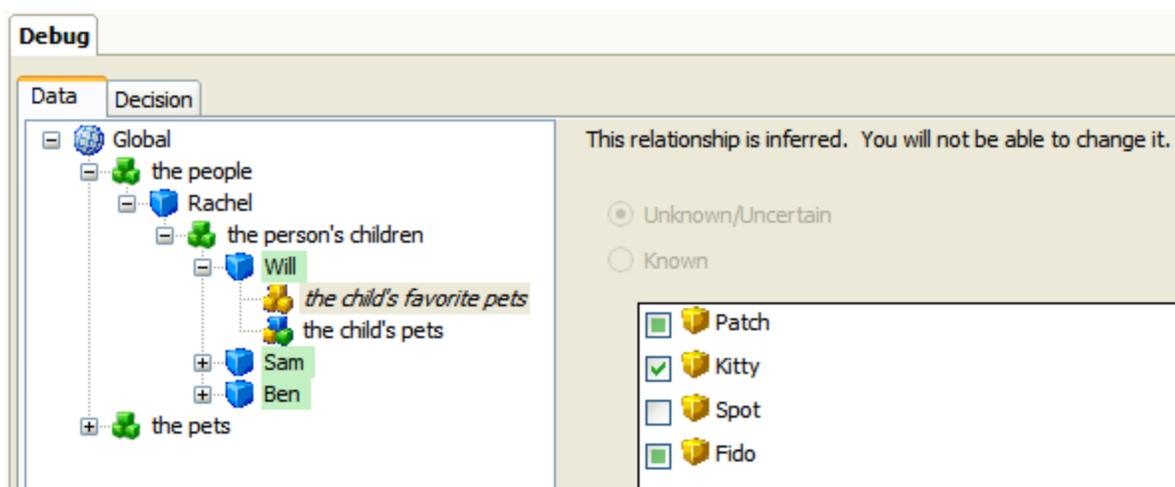
### Entity completion status and inferred relationships

An inferred relationship can also be partially known because the target entity is *not complete*. This is because more target entity instances may yet be added, and some of these may satisfy the inferred relationship's membership rule.

For more information, refer to the topic [Understand containment relationships and entity completion](#).

#### Partially known inferred relationships in the debugger

The screenshot below shows how a partially known inferred relationship is displayed in the debugger:



Here the relationship 'the child's favorite pets' is being examined for the child Will. The relationship editor is showing that:

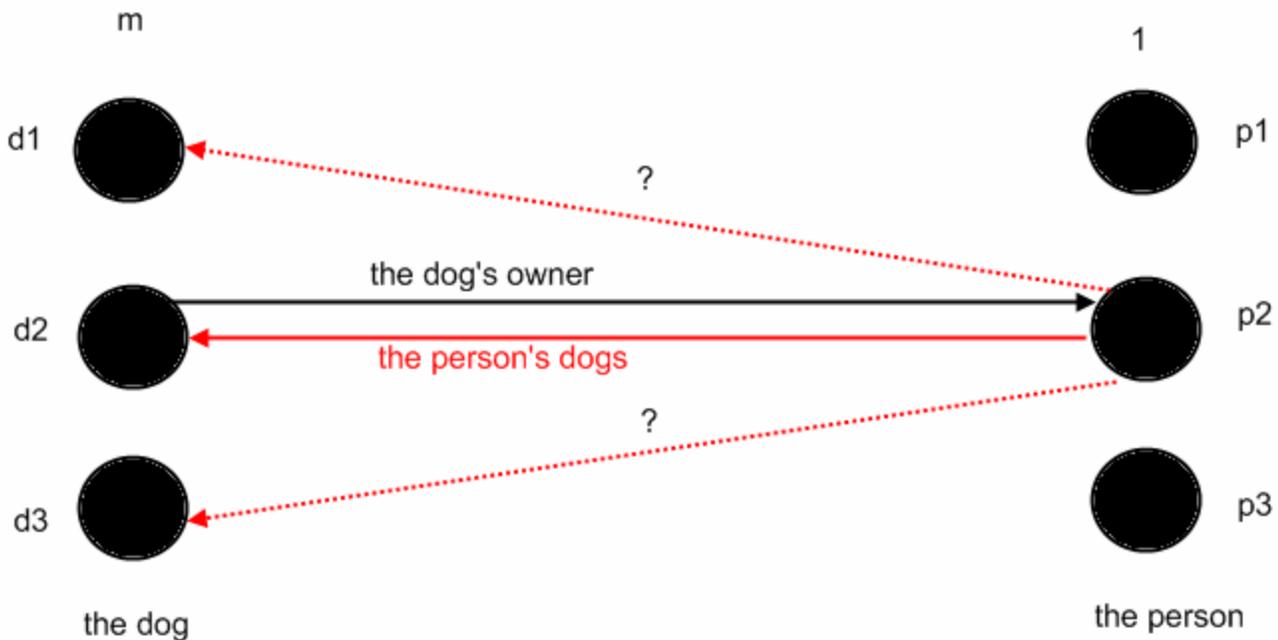
- the relationship is unknown - therefore this is a partial knowledge situation
- the pet Kitty is known to be a target of the relationship
- the pet Spot is known to *not* be a target of the relationship
- the pets Patch and Fido may or may not be targets of the relationship - this cannot be determined until more information is provided.

A tri-state checkbox is used in the relationship editor. It is important to understand the meaning of each state of the checkbox:

- Green tick - entity instance is known to be a target of the relationship
- No tick - entity instance is known to *not* be a target of the relationship
- Green square - entity instance may or may not be a target of the relationship.

### Partial knowledge of static relationships

When a relationship is set, the rule engine automatically sets the reverse relationship. In the case of many-to-one and many-to-many relationships, this can result in the reverse relationship being partially known. Take the following example:



Here we have:

- Two entities: the dog and the person
- Three instances of the dog: d1, d2, and d3
- Three instances of the person: p1, p2, and p3
- A many-to-one relationship between the dog and the person called the dog's owner. The reverse (one-to-many) relationship is called the person's dogs.

If d2's owner is set to be p2 (the solid black line), then the rule engine will set p2's dogs automatically. It is known that d2 is one of the person's dogs (the solid red line). There is no information about d1 and d3 however; the dog's owner is unknown for both of these entity instances. Hence it is not known whether d1 or d3 are member's of p2's dogs - they may or may not be, hence they are represented with a red dotted line.

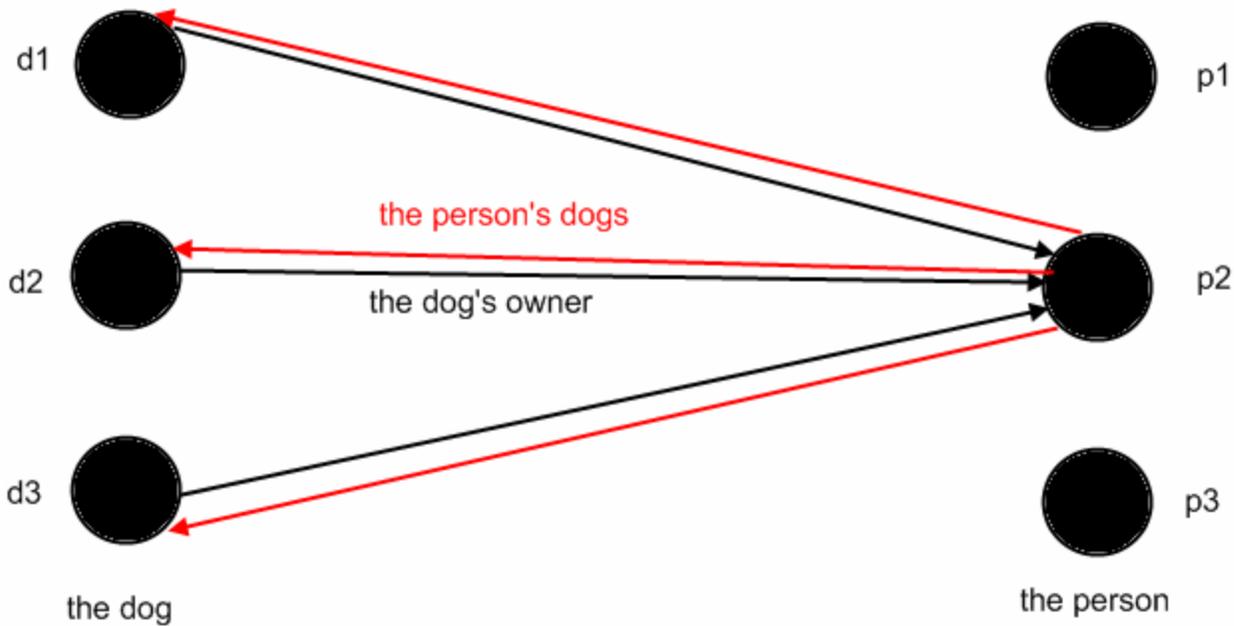
This leads to a situation of partial knowledge. For p2, the person's dogs is a partially known relationship. There is one known target, d2, and d1 and d3 may or may not be members of the relationship.

In the same way, setting many-to-many relationships can lead to a partially known reverse relationship.

NOTE: The rule engine does not currently determine "not targets" for partially known static relationships. A partially known static relationship can currently only consist of known targets, and entity instances that may or may not be members of the relationship.

**Static relationships and entity completion status**

Take the following example:



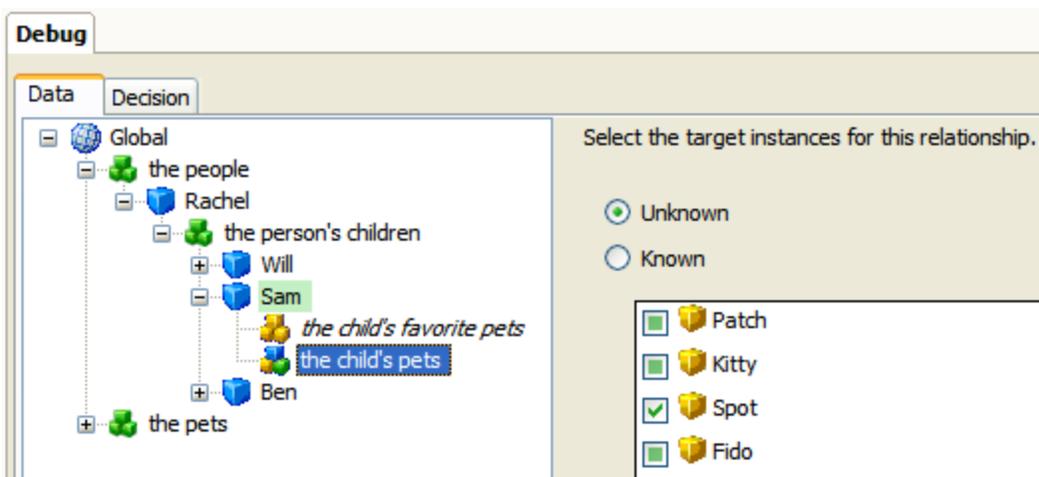
Here:

- d1, d2, and d3 all have the dog's owner set to p2.
- the engine then determines that the person's dogs for p2 has three known targets, d1, d2 and d3.

If the entity the dog is "not complete" (ie not all instances of the entity have potentially been collected), then despite all the available dogs being known targets, we still have a partial knowledge situation. This is because extra dogs could be created, which may or may not be members of the relationship. On the other hand, if the entity the dog is "complete" (ie all the instances of the entity are known to have been collected), then this cannot occur, and the engine will determine that the person's dogs is a fully known relationship for p2.

**Partially known static relationships in the debugger**

Partially known static (ie non-inferred) relationships are displayed in the debugger in a similar fashion to inferred relationships, as seen below:



Here the relationship 'the child's pets' is being examined for the child Sam. The relationship editor is showing that:

- the pet Spot is a target of the relationship
- the pets Patch, Kitty and Fido may or may not be a target of the relationship - this cannot be determined until more information is provided.

Note that unlike for inferred relationships, only two states are shown when displaying a partially known static relationship in the debugger:

- Green tick - entity instance is known to be a target of the relationship
- Green square - entity instance may or may not be a target of the relationship.

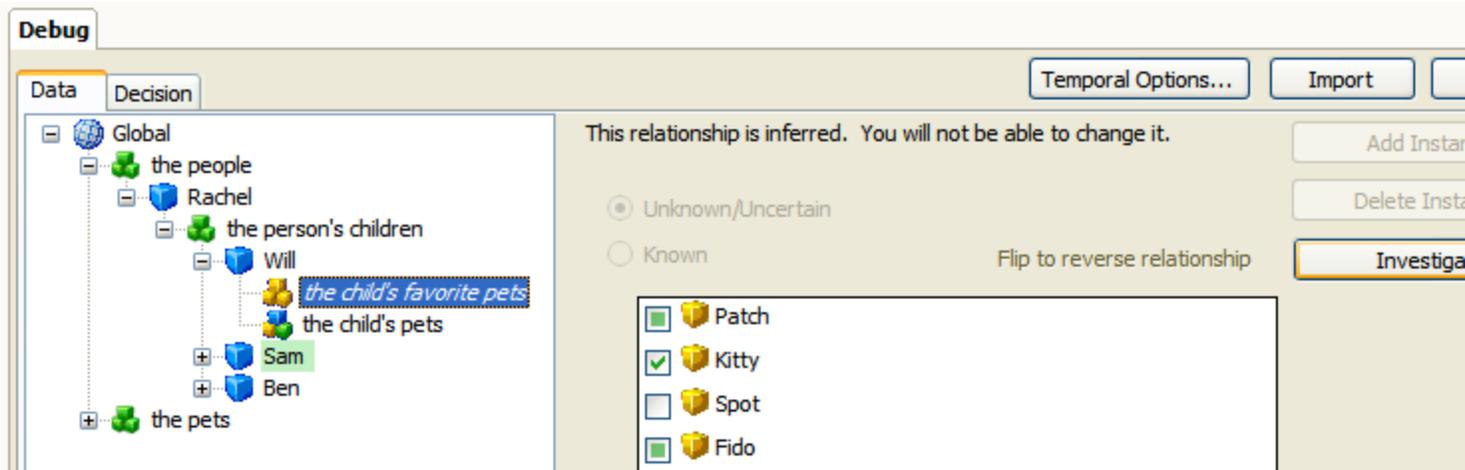
This is because the rule engine does not support "not targets" for static relationships.

Make a partially known relationship known in the debugger

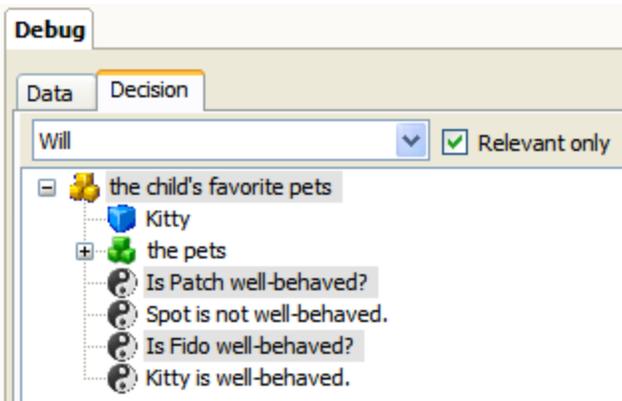
### Make a partially known inferred relationship known in the debugger

To make an inferred relationship known in the debugger you need to investigate the relationship. To do this:

1. In the Data view, select the inferred relationship that you want to investigate. (TIP: Inferred relationships are shown by a yellow multi-cube icon.)
2. In the right hand pane, click the **Investigate** button.



3. The Decision view will be shown with any relevant unknown attributes or relationships highlighted.

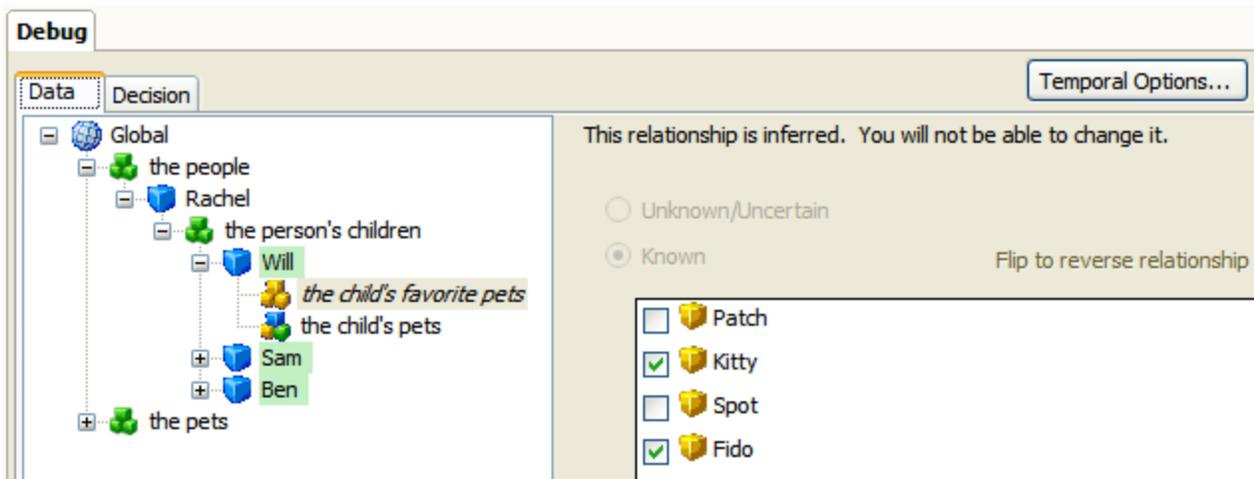


(If any unknown containment relationships are highlighted, right-click and choose **Edit Relationship**, to go to the relationship editor in the Data view. Complete the relationship by adding entity instances or using the **Containment Complete** option on the relationship in the Data view, then return to the Decision view.)

Double-click any unknown attributes to set values for them. The Decision view will then update to show which attributes contributed to this conclusion.



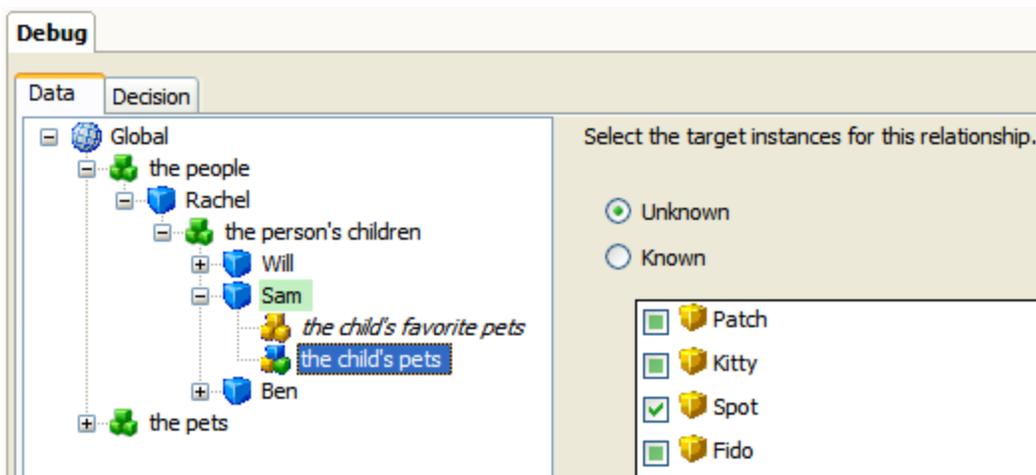
4. You can also switch back to the Data view to see which entity instances, if any, have been inferred as target instances of this relationship.



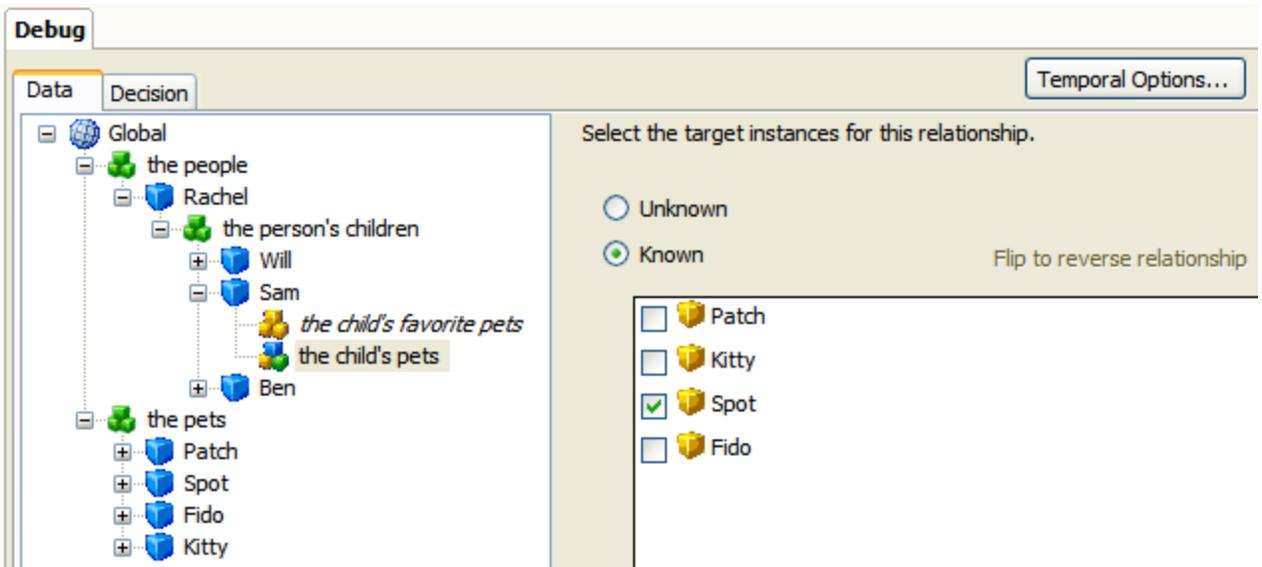
### Make a partially known static relationship known in the debugger

Unlike partially known inferred relationships in the debugger, partially known static relationships can be directly set to being known. To do this:

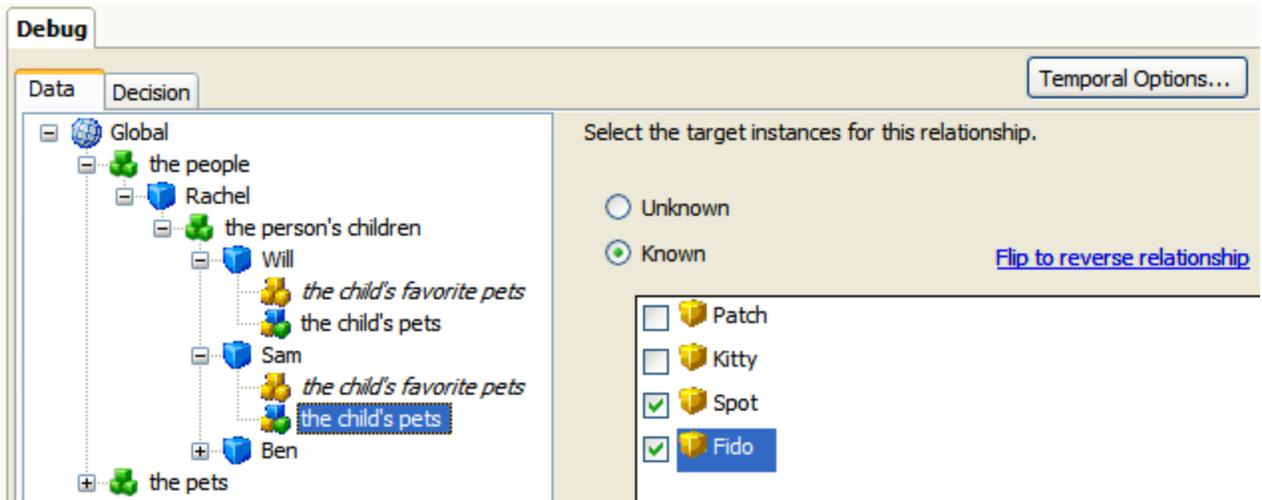
1. In the Data view, select the static relationship in the left hand pane. (TIP: Static relationships are shown by a multi-coloured cube icon.) This will display any existing entity instances in the relationship editor in the right hand pane.



2. In the right hand pane, select the **Known** option. NOTE: Any entity instances which may or may not have been targets of that relationship (the checkboxes with the green square) will now be set as *not* being targets of the relationship. (In this example, the pet Spot was a known target and remains this way. The pets Patch, Kitty and Fido on the other hand, may or may not have been targets and are now marked as not being targets of the relationship.)



3. Select the check box for any existing entity instances that you want to associate with that relationship.



### Understand containment relationships and entity completion

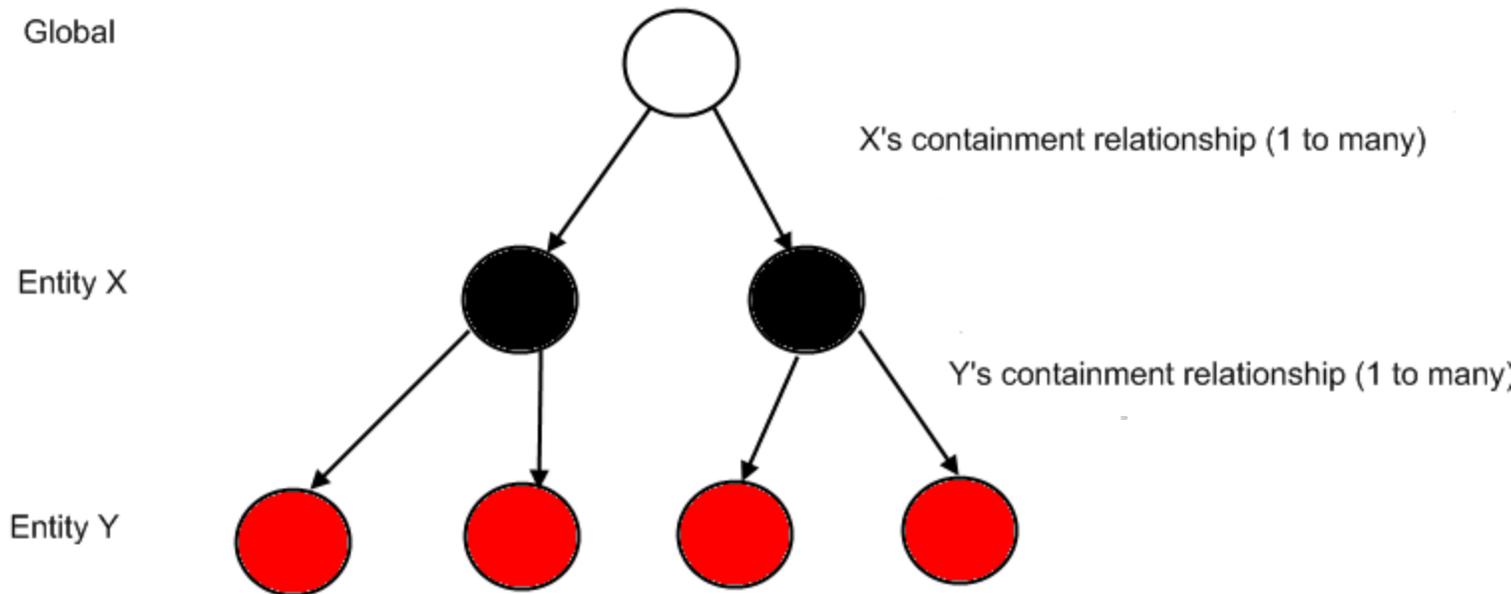
When an entity is considered to be *complete*, the rule engine assumes that it knows about the entire set of instances for that entity. An entity's completion status (whether or not it is considered to be complete) is of major importance when determining whether or not a relationship is partially known. See [Understand partial knowledge of relationships](#).

The completion status of an entity is determined by the engine through the use of *containment relationships*. A containment relationship is a one-to-many relationship from a parent entity to a child entity, and is created automatically when an entity instance is added, based on the entity and containment definition defined in the properties file for the rulebase. An entity Y is considered to be complete if:

1. A one-to-many containment relationship is defined from some other entity X to entity Y. We say that *Y is contained by X*, and we refer to the relationship as *entity Y's containment relationship*.
2. Entity Y's containment relationship is set (ie it is known) for all instances of entity X.
3. Entity X is also considered to be complete.

NOTE: The global entity is always automatically complete. It is not necessary (or possible) to create a containment relationship for the global entity.

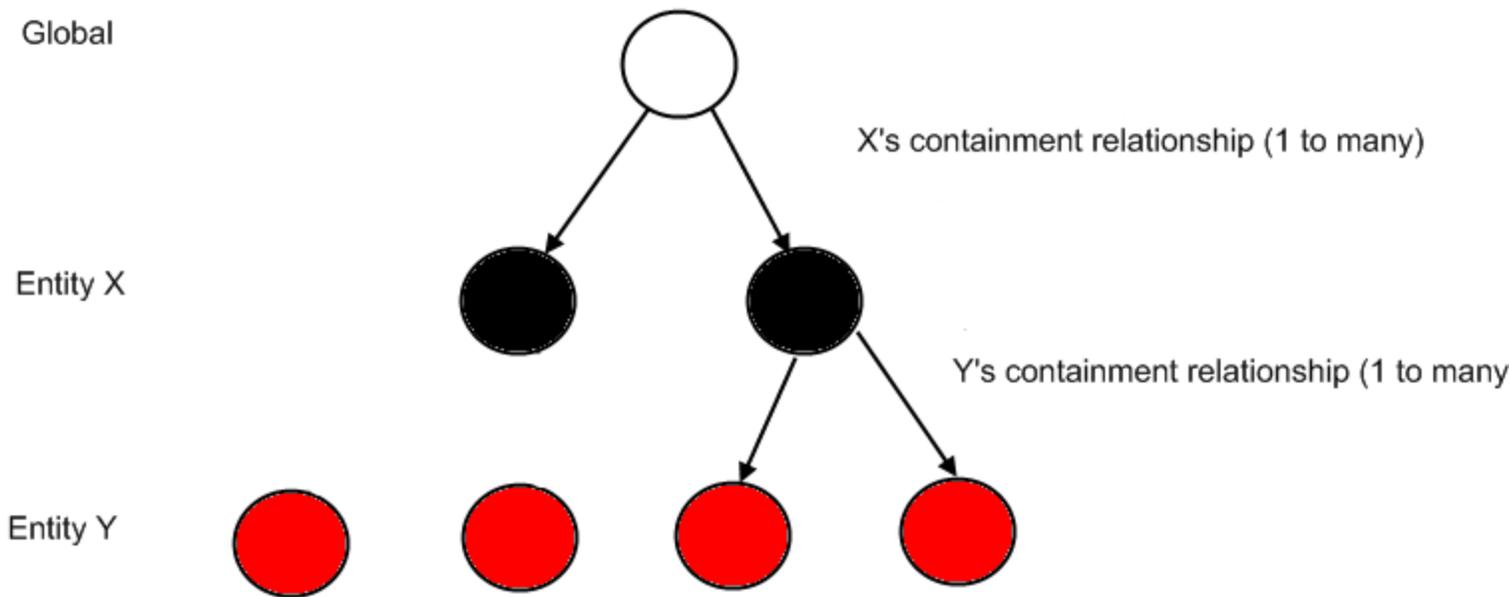
Consider the example provided in the following diagram:



In this scenario:

1. The global entity is automatically complete.
2. Entity X is complete. This is because its containment relationship from the global is known.
3. Entity Y is complete. This is because its containment relationship is known for all instances of entity X, and entity X is complete.

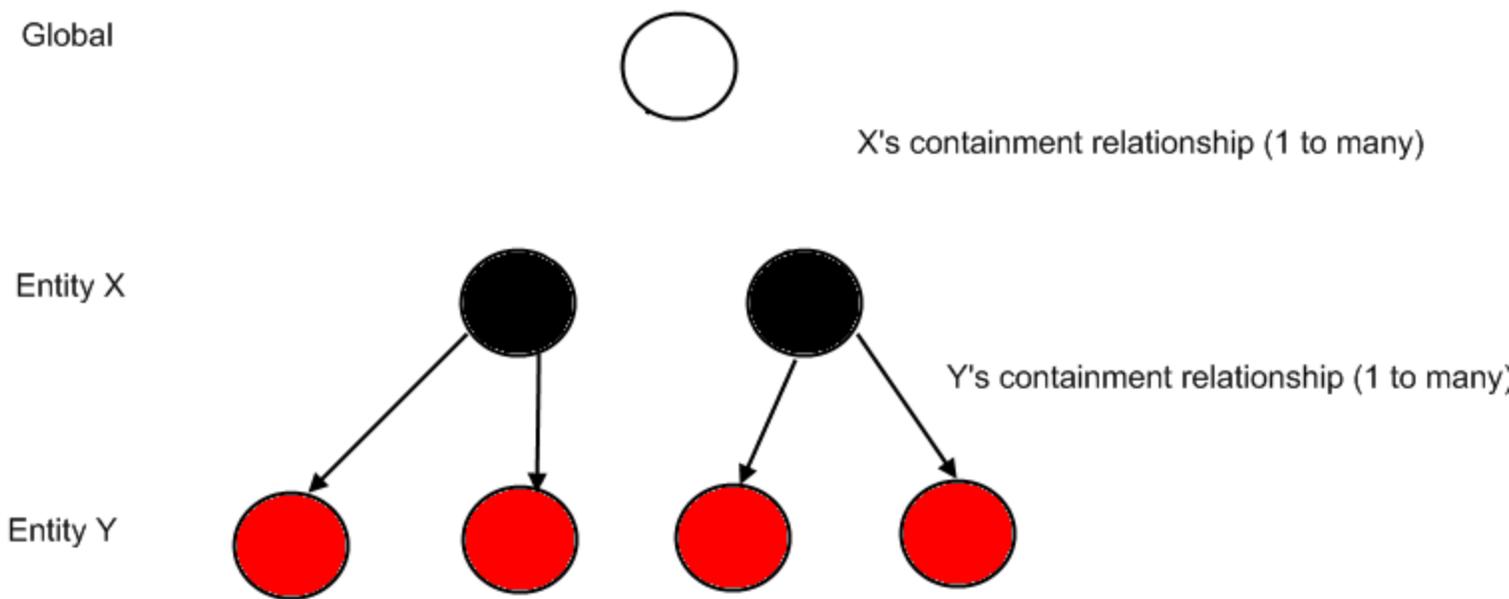
Consider this second scenario:



In this scenario:

1. The global entity is automatically complete.
2. Entity X is complete. This is because its containment relationship from the global is known.
3. Entity Y is **not** complete, because its containment relationship is not known for all instances of entity X.

Consider this third scenario:



In this scenario:

1. The global entity is automatically complete.
2. Entity X is **not** complete, as its containment relationship is not known.
3. Entity Y is **not** complete, as entity X is not complete. This is the case even though entity Y's containment relationship is known for all instances of entity X.

## Rules using entity instances

Topics in "Rules using entity instances"

- Use an entity or relationship in a rule
- Check whether entity instances match a condition
- Reason across multiple entities
- Write rules that infer relationships and entities
- View and amend the data model while writing rules

See also:

- Use an entity or relationship in a rule
- Build a temporal value from entity instances

### Use an entity or relationship in a rule

To write rules in Oracle Policy Modeling you need to understand how to refer to the different parts of the data model within your rules.

### What do you want to do?

Refer to entities connected by a to-many relationship

Refer to entities connected by a to-one relationship

Compare entities within the same relationship

Count the number of instances of an entity

Get the highest/most recent value of an entity-level variable

Get the lowest/least recent value of an entity-level variable

Add up numerical values gathered from each instance of an entity

### Refer to entities connected by a to-many relationship

Anytime you refer from one entity to another entity in a "to-many" relationship, you need to indicate whether one or all members of the target entity group need to satisfy the rule.

Consider the following rule:

A family may board the plane first if their child is under 8 years of age

We know that families can have more than one child, however, this rule does not specify whether one or all of the family's children must be under 8 years of age in order for the family to board the plane first. If the family had 2 children, one aged 4 and one aged 16, how would you decide?

The rule would be clearer if written in such a way that the reader can tell whether the rule applies to one or all children. For example:

A family may board the plane first if they have at least one child under 8 years of age

We use quantifiers, which are a type of Oracle Policy Modeling syntax, to write these kinds of rules. Quantifiers are operators which access data across the instances of an entity. The two quantifiers we use are:

- the universal quantifier - used to check that the condition returns true for every instance of an entity. For example, "All of the apples are red".
- the existential quantifier - used to check that the condition returns true for at least one instance of an entity. For example, "At least one of the bananas is yellow".

### **Check that a condition returns true for every instance of an entity**

The universal quantifier must be used when you refer from one entity to another entity in a "to-many" relationship, AND you need to determine whether all members of the target entity group need to satisfy the rule. This quantifier works in much the same way across entities as the 'and' operator does across attributes. This means that the rule using the universal quantifier will only evaluate to true when the condition is true for all instances of an entity. In other words, the conclusion will evaluate to false if its condition is false for one of the targets of the relationship provided. This applies even when the relationship provided is only [partially known](#).

There are two types of entity function that are used as universal quantifiers: the For All function and the For All Scope function. This section describes the use of the For All function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the For All Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the For All function is used where there is only one condition. For example, you could have the following rule where 'the family' is an entity (the source entity), 'the child' is an entity (the target entity), and 'the family's children' is the relationship between the entities (the relationship text).

#### **the family is ready to travel overseas if**

```
ForAll(the family's children, the child has a passport)
```

There are several ways of writing a For All function - see the [Entity and relationship function reference](#) for more detail.

Note that if there are zero instances of the entity, then the rule using the For All operator will evaluate to true.

### **Check that a condition returns true for at least one instance of an entity**

The existential quantifier must be used when you refer from one entity to another entity in a "to-many" relationship, AND you need to determine whether any members of the target entity group need to satisfy the rule. This quantifier works in much the same way across entities as the 'or' operator does across attributes. This means that only one instantiation of the entity must be true for the attribute using the operator to be true. In other words, the conclusion will evaluate to true if its condition is true for one of the targets of the relationship provided. This applies even when the relationship provided is only [partially known](#).

There are two types of entity function that are used as existential quantifiers: the Exists function and the Exists Scope function. This section describes the use of the Exists function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the Exists Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the Exists function is used where there is only one condition. For example, you could have the following rule where 'the family' is an entity (the source entity), 'the child' is an entity (the target entity), and 'the family's children' is the relationship between the entities (the relationship text).

### **the family is eligible for the benefit if**

Exists(the family's children, the child is a qualifying child)

There are several ways of writing an Exists function - see the [Entity and relationship function reference](#) for more detail.

Note that if there are zero instances of the entity, then the rule using the Exists operator will evaluate to false.

### **Refer to entities connected by a to-one relationship**

When you refer from one entity to another entity in a "to-one" relationship that is not a containment relationship, you need to use a particular syntax to connect the two entities together. There are two types of entity functions used for this purpose: the For function and the For Scope function. This section describes the For function which is used where there is only one condition (ie the rule only refers to one relationship). The use of the For Scope function, where there are one or more conditions (eg when you want to reason across several different relationships in the one rule), is more advanced and is covered in [Extend the For, For All and Exists functions](#).

As mentioned above, the For function is used where there is only one condition. For example, you could have the following rule where 'the child' is an entity (the source entity), 'the school' is an entity (the target entity), and 'the child's school' is the many-to-one relationship between the entities (the relationship text).

### **the child has a day off school if**

For(the child's school, the school is closed)

There are a couple of ways of writing a For function - see the [Entity and relationship function reference](#) for more detail.

#### **NOTES:**

- i. The For syntax can also be used for many-to-many relationships. (The only relationship type that it can't be used with is one-to-many.)
- ii. The For syntax does not need to be used when referring to a parent relationship in the entity's containment relationships. For example, if an entity 'the pet' is contained within an entity 'the child', you could write the following rule without needing to refer to the containment relationship explicitly:

### **the pet is playing outside if**

the child is playing outside

### **Compare entities within the same relationship**

To compare entities within the same relationship, you need to add an alias to the entities involved. Aliasing allows you to provide an alternative name used to refer to an entity instance. For more information, see [Remove ambiguity when reasoning about more than one instance of the same entity](#).

### **Count the number of instances of an entity**

To count the number of instances there are of an entity, you use the Instance Count function. The syntax for this function is:

- InstanceCount(<relationship text>)
- the number of <relationship text>

For example, the Instance Count function could be used to determine the number of children belonging to the claimant:

**the number of children that the claimant has = InstanceCount(the claimant's children)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 4 for the following data:

the child
Anthony
Peter
Rebecca
Fiona

Get the highest/most recent value of an entity-level variable

To obtain the highest or most recent value of an entity-level variable for all instances of the entity, you use the Instance Maximum function. The syntax for this function is:

- InstanceMaximum(<relationship text>,<entity-level variable>)
- the greatest of <entity-level variable> for all of <relationship text>
- <entity-level date> which is the latest for all of <relationship text>
- the latest of all <entity-level date> for <relationship text>
- <entity-level variable> which is the greatest for all of <relationship text>

For example, the Instance Maximum function could be used to determine the highest bank balance for a child of the claimant:

**the highest bank balance for a child of the claimant = InstanceMaximum(the claimant's children, the child's bank balance)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.

The function returns a value of \$175 for the following data:

the child	the child's bank balance
Annabel	\$50
Katrina	\$175

the child	the child's bank balance
Mike	\$120

Get the lowest/least recent value of an entity-level variable

To obtain the lowest or least recent value of an entity-level variable for all instances of the entity, you use the Instance Minimum function. The syntax for this function is:

- InstanceMinimum(<relationship text>, <entity-level variable>)
- the least of <entity-level variable> for all of <relationship text>
- <entity-level variable> which is the least for all of <relationship text>
- <entity-level date> which is the earliest for all of <relationship text>
- the earliest of all <entity-level date> for <relationship text>

For example, the Instance Minimum function could be used to determine the lightest of the claimant's children:

**the lightest weight for a child of the claimant = InstanceMinimum(the claimant's children, the child's weight in kilograms)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 15 for the following data:

the child	the child's weight in kilograms
Harry	15
Sharon	30
Fran	45

Add up numerical values gathered from each instance of an entity

To obtain the sum of all instances of an entity-level variable, you use the Instance Sum function. The syntax for this function is:

- InstanceSum(<relationship text>, <entity-level variable>)
- <entity-level variable> totaled for all of <relationship text>

For example, the Instance Sum function could be used to determine the total Child Care Benefit payable to the claimant:

**the total Child Care Benefit payable to the claimant = InstanceSum(the claimant's children, the Child Care Benefit amount for the child)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of \$900 for the following data:

the child	the Child Care Benefit amount for the child
Mary	\$500
Sam	\$250
Lizzie	\$150

See also:

- [Reason across multiple entities](#)
- [Entity and relationship function reference](#)
- [Entity and relationship function rule examples](#)

## Check whether entity instances match a condition

When using entities in a rulebase, you can check whether entity instances match particular conditions.

### What do you want to do?

[Count the number of instances of an entity for which a particular attribute is true](#)

[Get the highest/most recent value of an entity-level variable for which a particular attribute is true](#)

[Get the lowest/least recent value of an entity-level variable for which a particular attribute is true](#)

[Add up numerical values gathered from each instance of an entity for which a particular attribute is true](#)

### Count the number of instances of an entity for which a particular attribute is true

To count the number of instances there are of an entity for which a particular entity-level attribute has a particular value, you use the Instance Count If function. The syntax for this function is:

- InstanceCountIf(<relationship text> ,<entity-level condition> )
- the number of <relationship text> for which it is the case that <entity-level attribute>

For example, the Instance Count If function could be used to determine the number of school students for the claimant:

**the number of school students that the claimant has = InstanceCountIf(the claimant's children, the child is a school student)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 1 for the following data:

the child	the child is a school student
Rachel	false
Michael	false
Simon	true

NOTES:

- i. You can only put one attribute as the 'If' parameter in the function, but that attribute can be proven in a separate rule by any number of other conditions.
- ii. The InstanceCountIf() function will return unknown if the relationship supplied to it is unknown, regardless of whether or not any of the relationship's targets are known. It will also return unknown if the attribute being examined is unknown for any of the relationship's targets.

Get the highest/most recent value of an entity-level variable for which a particular attribute is true

To obtain the highest or most recent value of an entity-level variable for all instances of the entity for which a particular entity-level attribute has a particular value, you use the Instance Maximum If function. The syntax for this function is:

- InstanceMaximumIf(<relationship text>, <entity-level variable>, <entity-level condition>)

For example, the Instance Maximum If function could be used to determine the most recent date of employment of a permanent employee by a company:

**the most recent date of employment of a permanent employee by the company = InstanceMaximumIf(the company's employees, the employee's date of employment, the employee is a permanent employee)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the company' , an entity 'the employee' and a one-to-many relationship 'the company's employees'.)

The function returns a value of 15/05/2006 for the following data:

the employee	the employee's date of employment	the employee is a permanent employee
David	01/01/2006	true
Shaun	24/08/2006	false
Anita	15/05/2006	true

NOTES:

- i. You can only put one attribute as the 'If' parameter in the function, but that attribute can be proven in a separate rule by any number of other conditions.

- ii. The InstanceMaximumIf() function will return unknown if the relationship supplied to it is unknown, regardless of whether or not any of the relationship's targets are known. It will also return unknown if the attribute being examined is unknown for any of the relationship's targets.

Get the lowest/least recent value of an entity-level variable for which a particular attribute is true

To obtain the lowest or least recent value of an entity-level variable for all instances of the entity for which a particular entity-level attribute has a particular value, you use the Instance Minimum If function. The syntax for this function is:

- InstanceMinimumIf(<relationship text>,<entity-level variable>,<entity-level condition>)

For example, the Instance Minimum If function could be used to determine the youngest of the claimant's female children:

**the youngest of the claimant's female children = InstanceMinimumIf(the claimant's children, the child's age, the child is female)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of 4 for the following data:

the child	the child's age	the child is female
Sam	3	false
Alex	4	true
Shannon	6	false
Paris	8	true

NOTES:

- i. You can only put one attribute as the 'If' parameter in the function, but that attribute can be proven in a separate rule by any number of other conditions.
- ii. The InstanceMinimumIf() function will return unknown if the relationship supplied to it is unknown, regardless of whether or not any of the relationship's targets are known. It will also return unknown if the attribute being examined is unknown for any of the relationship's targets.

Add up numerical values gathered from each instance of an entity for which a particular attribute is true

To obtain the sum of all instances of an entity-level variable for which it is true of the entity that a specific entity-level boolean attribute is true, you use the Instance Sum If function. The syntax for this function is:

- InstanceSumIf(<relationship text>,<entity-level variable being summed>,<entity-level condition>)
- total for all <relationship text>,<entity-level variable> only where <entity-level attribute>
- <entity-level variable> totaled for all of <relationship text> for which it is the case that <entity-level attribute>

For example, the Instance Sum If function could be used to determine the total boarding school fees for the claimant:

**the total cost of boarding school fees for the claimant = InstanceSumIf(the claimant's children, the annual school fees for the child, the child attends a boarding school)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the claimant' , an entity 'the child' and a one-to-many relationship 'the claimant's children'.)

The function returns a value of \$33000 for the following data:

the child	the annual school fees for the child	the child attends a boarding school
Sally	\$18000	true
James	\$15000	true
Bob	\$10000	false

#### NOTES:

- i. You can only put one attribute as the 'If' parameter in the function, but that attribute can be proven in a separate rule by any number of other conditions.
- ii. The InstanceSumIf() function will return unknown if the relationship supplied to it is unknown, regardless of whether or not any of the relationship's targets are known. It will also return unknown if the attribute being examined is unknown for any of the relationship's targets.

See also:

- [Check that a condition returns true for every instance of an entity](#)
- [Check that a condition returns true for at least one instance of an entity](#)
- [Count the number of instances of an entity](#)
- [Get the highest/most recent value of an entity-level variable](#)
- [Get the lowest/least recent value of an entity-level variable](#)
- [Add up numerical values gathered from each instance of an entity](#)

### Reason across multiple entities

Using extended forms of the For, Exists and ForAll functions, you can reason across several different entities in a single rule. You can also reason with different instances of the same entity, and compare instances of the same entity.

### What do you want to do?

[Extend the For, For All and Exists functions](#)

[Use relationship membership as a rule input](#)

[Remove ambiguity when reasoning about more than one instance of the same entity](#)

[Compare instances of the same entity](#)

## Extend the For, For All and Exists functions

Entity functions, such as For, Exists and For All, allow you to reason across a single relationship from a source entity to a target entity. If, however, you want to cross multiple relationships in a rule and reason against several entities in that rule, you need to use different entity functions (For Scope, For All Scope and Exists Scope) that specify the 'scope' of the entities.

### The concept of scope in rules

To fully understand cross entity reasoning, it is important to understand the concept of scoping in relation to a rule. The Word compiler processes a rule in a top-to-bottom fashion. The top (conclusion) line is evaluated, then the second line, and so on. For a particular line in a rule, the scope is the set of entity instances that have been previously mentioned in parent rule lines. These are the only entity instances that reasoning can be done on. In most cases, the conclusion line introduces a single entity instance into scope. (The exception to this is a membership conclusion which introduces two entities into scope - the source of the inferred relationship and the target of the inferred relationship. See [Write rules that infer relationships and entities](#) for more information). Once an entity instance has been introduced to the scope, it can be reasoned with in all rule lines that are children of the line that introduced it. The global entity is always available to the rule scope, as are the parent entities in the entity's containment relationship.

The For Scope, For All Scope and Exists Scope functions are used as scoping statements to cross a relationship and thereby introduce the target of the relationship into scope.

### An example of cross entity reasoning

This example is based on a telecommunications retail style model where a customer entity has a one-to-one relationship to a plan entity (the customer's current plan) which has a many-to-many relationship to specific products (the plan's products). That is:

the customer -> the customer's current plan (one-to-one) -> the plan -> the plan's products (many-to-many) -> the product

You could then have the following rule which uses the For Scope and Exists Scope entity functions:

#### **the customer has incompatible products if**

```
ForScope(the customer's current plan)
  ExistsScope(the plan's products)
    the plan's network <> the product's network
```

In the example above, the initial scope for this rule is "the customer", established by the conclusion on line 1. "The plan" was brought into the scope in line 2 by crossing the relationship "the customer's current plan", and "the product" was brought into the scope in line 3 by crossing the relationship "the plan's products". Finally in line 4 we compare an attribute of "the plan" with an attribute of "the product". Both entities are now in scope so this reasoning is possible.

An example of the For All Scope function using the same data model would be:

#### **the customer is satisfied if**

```
in the case of the customer's current plan
  ForAllScope(the plan's products)
    the product's rating = "AAA"
```

## Use relationship membership as a rule input

Relationship membership can be used as a rule input by creating a membership statement and using it as a condition in a rule. A membership statement always reasons against the source entity and the target entity. The membership statement will be true if the target entity is the target of the relationship for the source entity. A membership statement can be used for any type of relationship.

A membership statement used as a condition takes one of the following forms:

- Positive form
  - <target entity> is a member of <relationship text>
  - IsMemberOf(<target entity>, <relationship text>)
- Negative form
  - <target entity> is a not member of <relationship text>
  - IsNotMemberOf(<target entity>, <relationship text>)

In the example rule below, a membership statement is used as a condition to determine if a dog is happy based on whether it is a member of the person's favorite dogs.

### **the dog is happy if**

```
ForScope(the dog's owner)
  IsMemberOf(the dog, the person's favorite dogs)
```

Both entities (ie the source entity and the target entity) must be brought into the scope of the rule, otherwise the compiler will attempt to create an attribute "<target entity> is a member of <relationship text>".

#### NOTES:

- a. A membership statement that is used as a condition cannot have any children under it in that rule. It must, however, be proved by another rule. For example, the membership statement in the example above must be proved by another rule (eg "the dog is a member of the person's favorite dogs if the dog is well-behaved").
- b. A membership statement can true when the relationship is [partially known](#). So long as it is known that the entity instance in question is a member of the relationship, the membership statement will return as true.

## Remove ambiguity when reasoning about more than one instance of the same entity

When you want to reason with more than one instance of the same entity, it can become ambiguous as to which entity instances your rules are referring to. You can use an alias for the desired entity instances in your rules to remove this ambiguity.

Aliasing allows you to use an alternative name to refer to an entity instance. An alias can be used in a conclusion or condition, and its use is limited to that particular conclusion or condition. NOTE: Once an alias is defined it must be used in place of the regular name for the associated entity, otherwise an error will occur. Also note that the name of an entity cannot be used as an alias.

An alias can be used in two places to resolve ambiguity: in a scope entity function, and in a relationship conclusion.

## Using an alias in a scoped entity function

In a scoped entity function (ie the functions For Scope, For All Scope and Exists Scope) an alias can be assigned to the target entity instance when the entity is already in the rule scope and you need to discriminate between those entities instances.

For example, if you wanted to compare two person entity instances through the relationship "the person's spouse", you could define an alias to the target of "the person's spouse". Once the alias is defined, you can then refer to attributes of the target instance as "the spouse":

### **the person has the highest taxable income if**

```
ForScope(the person's spouse, the spouse)
    the person's income > the spouse's income
```

Note that in this rule, you only need to add an attribute for 'the person's income', not for 'the spouse's income', as the compiler knows that any attributes using the alias text belong to the associated entity (in this case 'the person').

If you want to compare entity instances in a relationship that does not include the global entity (eg from 'the toy' to 'the child'), you need to traverse up and down the relationship as shown in the example below:

### **the toy is the same type as another owned by the same child if**

```
ForScope(the child who owns the toy)
    ExistsScope(the child's toys, the other toy)
        the toy type = the other toy type and
        the toy name <> the other toy name
```

## Using an alias in a relationship conclusion

In a relationship conclusion (ie the function Is Member Of) an alias can be assigned to the target entity instance in the membership statement. This is useful for situations where the source and the target of the relationship are the same entity. For example, in the rule below, "the person's co-workers" is a many-to-many relationship whose source is "person", and whose target is also "person":

### **the person (the colleague) is a member of the person's co-workers if**

```
the person's workplace = the colleague's workplace
```

## Compare instances of the same entity

When you are reasoning with more than one instance of the same entity, you may want to compare attribute values across entity instances. The comparison of the entity attribute values does not differentiate whether the target entity instance is the same as the one in which the rule is operating. This can result in the attribute comparisons being satisfied with values of the same entity instance, which may not be the logic that you wish to represent in your rules.

The InstanceEquals and InstanceNotEquals functions allow you to compare the entity instances themselves, so you can avoid this situation.

For example, the following rule examines instances of the employee entity to see whether any employee ID is used by multiple employees:

### **the employee has a conflicting ID if**

```
ExistsScope(the employees, the other employee)
    the employee's ID = the other employee's ID and
    InstanceNotEquals(the employee, the other employee)
```

The ID for each employee entity instance is compared against all employee IDs, one of which will be a match between the same entity instance. To eliminate this match from the concluded outcome of the rule, the InstanceNotEquals function is used to ensure only ID matches from different entity instances cause the rule to evaluate to true.

See also:

- [Use an entity or relationship in a rule](#)
- [Write rules that infer relationships and entities](#)
- [Understand how partial knowledge reasoning works](#)
- [Entity and relationship function reference](#)
- [Entity and relationship function rule examples](#)

## **Write rules that infer relationships and entities**

Rules that infer relationships and entities can be useful for grouping entity instances in order to refer to the group as a whole in your rules and use the standard entity functions in a more powerful way. For example, you could:

- Collect payments and write rules to sum all payments made within the same year
- Determine eligibility for benefits and write rules to sum all eligible benefits or create a payment plan for all eligible benefits
- Collect product information and write rules to determine which services should be created based on the customer's product

Further examples are provided under Worked Examples below.

## **What do you want to do?**

[Infer membership of a relationship](#)

[Infer existence of entities to satisfy the relationship](#)

[See worked examples](#)

### **Infer membership of a relationship**

The syntax to use to infer that existing entity instances are members of a relationship is:

- <target entity> is a member of <relationship text> if
- IsMemberOf(<target entity>, <relationship text>) if

Note that membership rules must be written in the positive form. That is, it is not possible to infer that an entity is not a member of a relationship.

All subsequent rule levels for this conclusion must have the source entity and the target entity in its reasoning scope. The relationship used must be defined as a many-to-many relationship type in the properties file for the project. (See [Define a relationship](#) for more information.)

In the example rule below, a membership statement is used to conclude membership of the inferred relationship 'the parent's school-aged children'.

**the child is a member of the parent's school-aged children if**

the child is of school age

### Notes / Limitations

1. A relationship conclusion can only ever be the top line of a rule. If the syntax is used anywhere else in a rule, then it will be treated as a membership statement (see [Use relationship membership as a rule input](#)).
2. An inferred relationship will be **partially known** if the rule used to infer it returns unknown for any of the potential target entity instances.
3. An inferred relationship will be **partially known** if its target entity is not **complete**.
4. Combining manually created relationships with inferred relationships is not allowed.
5. Combining inferred relationships with temporal values is not supported.
6. A relationship must only be inferred in its **primary direction**.

### Infer existence of entities to satisfy the relationship

You can also write a rule that creates entity instances to become members of a relationship.

The syntax to use to infer that entity instances should be created (or deleted) as members of a relationship is:

- <relationship> (<identifying value>) exists if
- InferInstance(<relationship>, <identifying value>) if

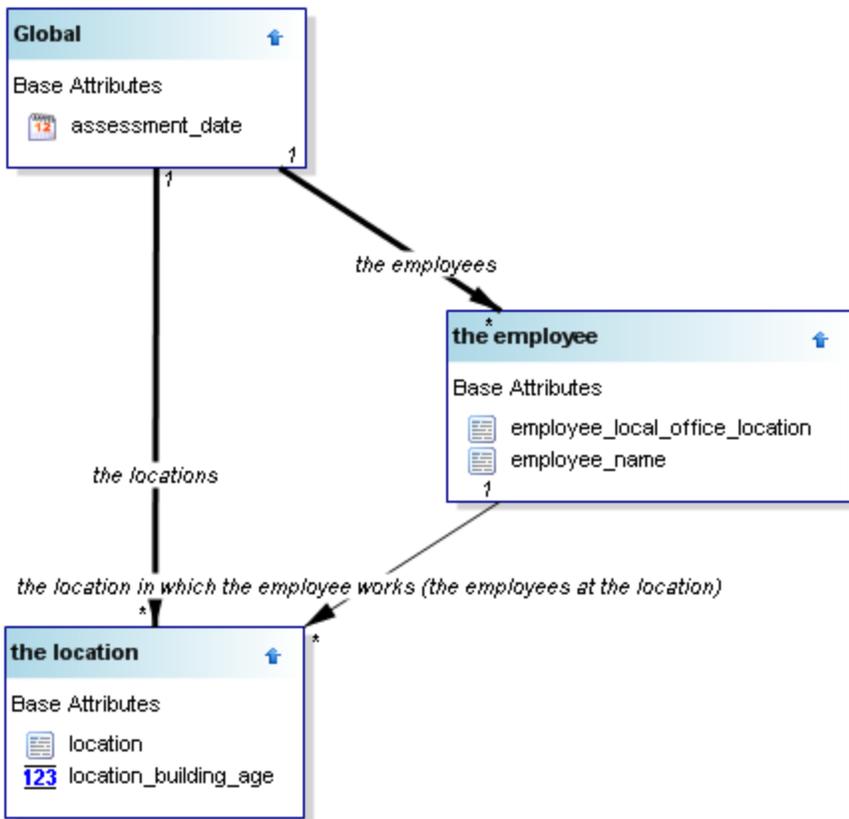
or in table form (where multiple instances are needed):

Relationship	
<identifying value>	Condition
<identifying value>	Condition

The identifying value can either be a fixed value ("spouse") or a variable (the tax year) which is then used as the **identifying attribute** for the entity instances created.

At runtime, the engine will evaluate each rule in the above form, evaluate the condition(s) and will create an instance for any condition that is true, and destroy any instance for which no condition returns true.

For example, assuming you have the following data model:



### Example 1: Creating a single instance

Writing the rule:

**the locations ("Main office") exists**

will create a single instance of the entity "the location" which is a member of the containment relationship "the locations". The instance will have "Main office" as the value of the identifying attribute.

The screenshot shows a "Debug" window with two tabs: "Data" and "Decision".

- Data Tab:** Displays a tree structure:
  - Global
    - the locations
      - Main office
      - the employees at the location
      - the employees

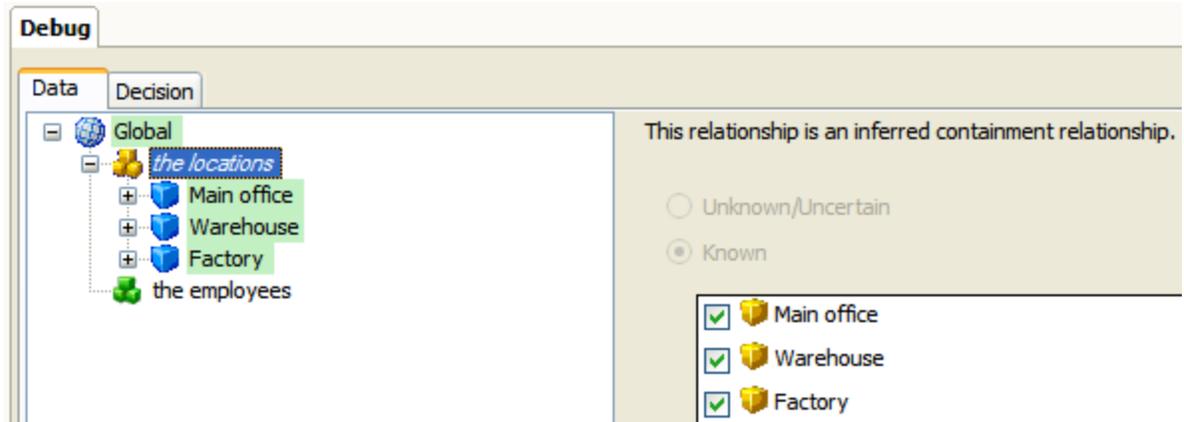
- Decision Tab:** Contains the text "This relationship is an inferred containment relationship." and two radio buttons:
- Unknown/Uncertain
- Known
- At the bottom, there is a checked checkbox next to a yellow cube icon and the text "Main office".

### Example 2: Creating multiple instances using a rule table

Writing the rule:

<b>the locations</b>	
<b>"Main office"</b>	the assessment date >= 2009-10-01
<b>"Warehouse"</b>	the assessment date >= 2000-01-15
<b>"Factory"</b>	the assessment date >= 2000-01-15

will create instances of the entity "the location" (depending on the assessment date), which are members of the containment relationship "the locations". These instances will have "Main office", "Warehouse" and "Factory" as the value of their identifying attributes.

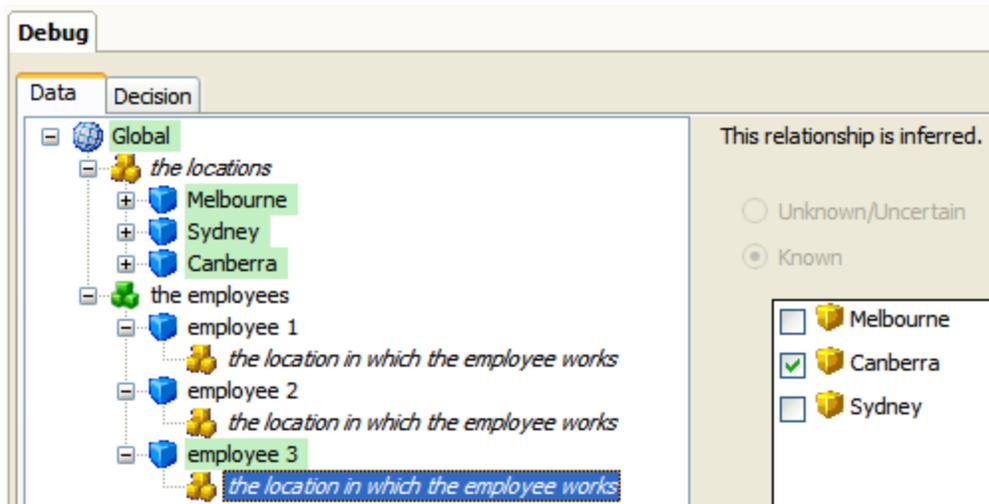


### Example 3: Creating multiple instances from a single entity level-attribute

Writing the rule:

**the location in which the employee works (the employee's local office) exists**

will create an instance of the location entity for each unique value of "the employee's local office". These instances will be members of the relationship "the location in which the employee works" and have the value of the employee's local office as the value of the identifying attribute.



## Notes /Limitations

1. Combining manually created instances with inferred instances is not allowed.
2. Combining inferred entity instances with temporal values is not supported.
3. Only a single attribute of the entity instance can be inferred as a part of the entity rule. For example, the type of benefit ("unemployment benefit") can be set but the amount of the benefit would have to be set via a separate rule.
4. Inferred entity instances may not contain base level attributes.
5. A relationship that participates in an inferred entity instance rule is considered to be an inferred relationship. This means that an inferred relationship rule cannot be used to prove the same relationship used in an inferred entity instance rule.

## See worked examples

The following example rulebases installed with Oracle Policy Modeling demonstrate the inferred entity instance functionality. For how to view these rulebases, see [Open an example rulebase](#).

- Inferred Brand Discount rulebase  
This rulebase models a generic purchase order scenario using inferred entity instances to group order items by brand and then apply a brand discount for purchases over \$100 for any given brand.
- Inferred Benefits rulebase  
This rulebase infers the existence of benefits and tallies the number of people eligible for each benefit. It also demonstrates inferred instances using rule tables.
- Inferred Tax Years rulebase  
This rulebase infers the existence of tax year entity instances so that further rules related to those tax years can be applied. This can be helpful if you want to ask further information about previous years (ie "did you submit a tax return for <tax year>") but only ask about tax years relevant to the interview, without pre-populating every possible tax year.
- Inferred Service Delta rulebase  
This rulebase infers the existence of service entity instances in order to identify which services should be started, stopped or retained when a customer changes phone plans. It also demonstrates inferred instances from global values.

See also:

- [Reason across multiple entities](#)
- [Investigate an inferred relationship](#)

## View and amend the data model while writing rules

Using the Data Model Browser, you can view and change the rulebase data model while writing your rules in Microsoft Word.

## What do you want to do?

[View the attributes, entities and relationships](#)

[Edit an attribute from within Word](#)

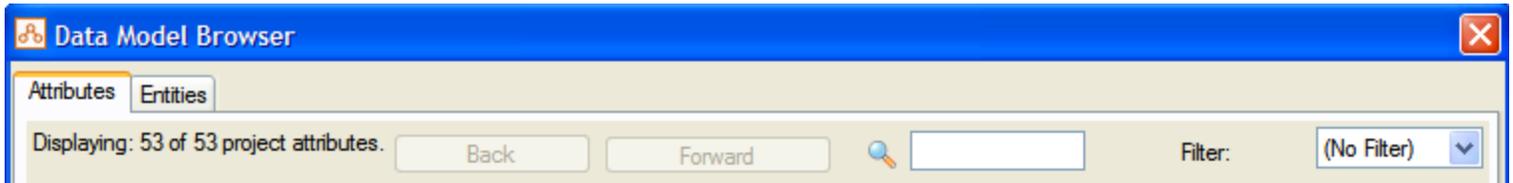
[Edit an entity from within Word](#)

[Edit a relationship from within Word](#)

## View the attributes, entities and relationships

To open the Data Model Browser from within Microsoft Word, press **Alt+D** or click the **Data Model Browser** button on the Oracle Policy Modeling toolbar.

At the top of the Data Model Browser are several options to help you navigate and filter the display of attributes and entities.



The **Back** and **Forward** buttons allow you to move back and forward between previous views.

The search field allows you to filter the lists according to the text provided. This search is case-insensitive.

The **Filter** drop down list allows you to filter the lists by attribute type.

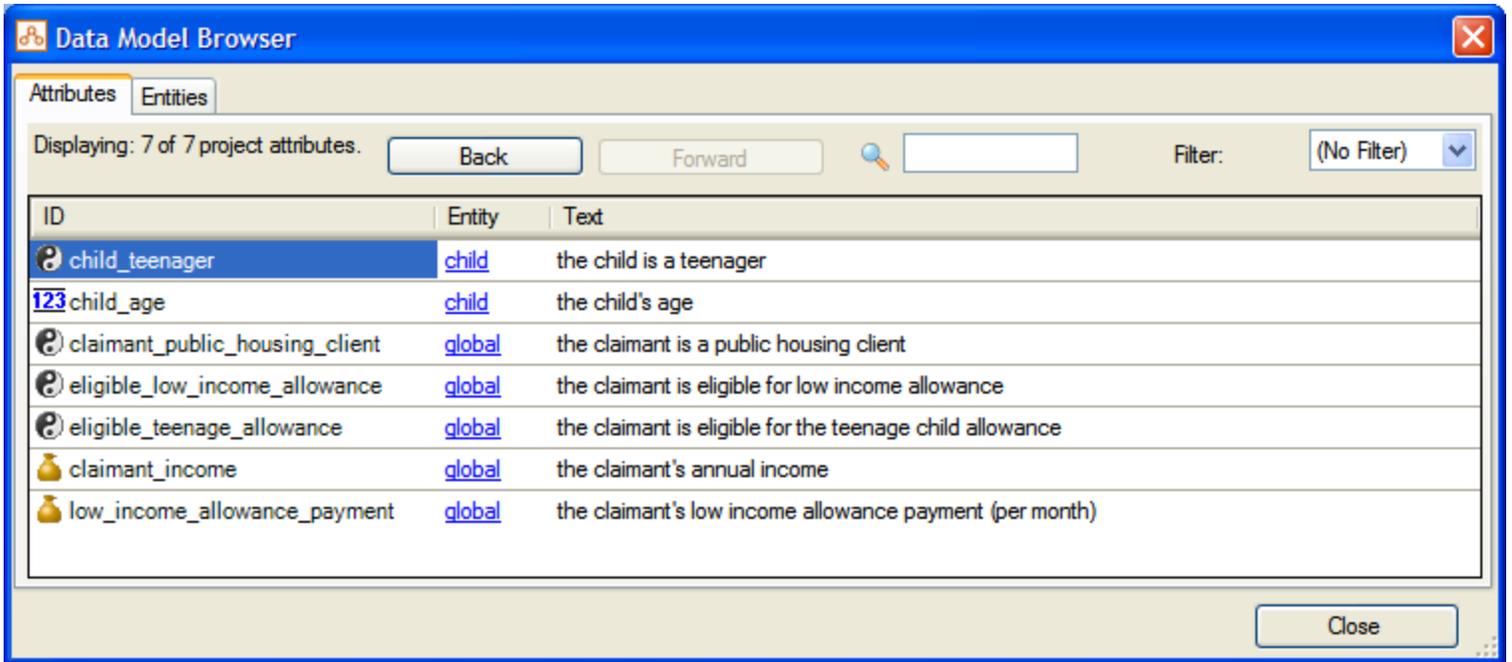
You can also sort lists by clicking on the column header - this will alternate between ascending and descending order.

The Data Model Browser shows four different views of the model. These views are:

- [project attributes](#)
- [project entities](#)
- [entity attributes](#)
- [entity relationships](#)

### View the project attributes

The project attributes view lists the attributes for all the entities in the project. This view is accessed by clicking on the **Attributes** tab.

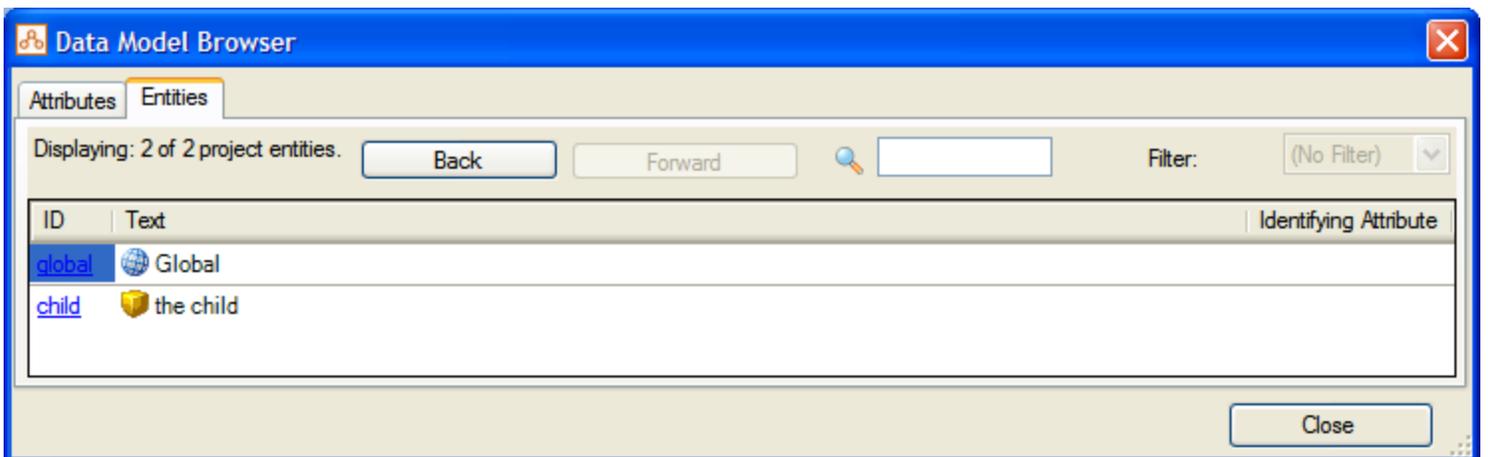


The following attribute properties are displayed:

- attribute ID (public name, if defined, otherwise build model id) and attribute type (indicated by an icon)
- entity that the attribute belongs to (this is a link to the [entity attributes view](#) for that entity)
- attribute text

### View the project entities

The project entities view lists all the entities in the project. This view is accessed by clicking on the **Entities** tab (and clicking the Back button if the view is showing the attributes for a particular entity).

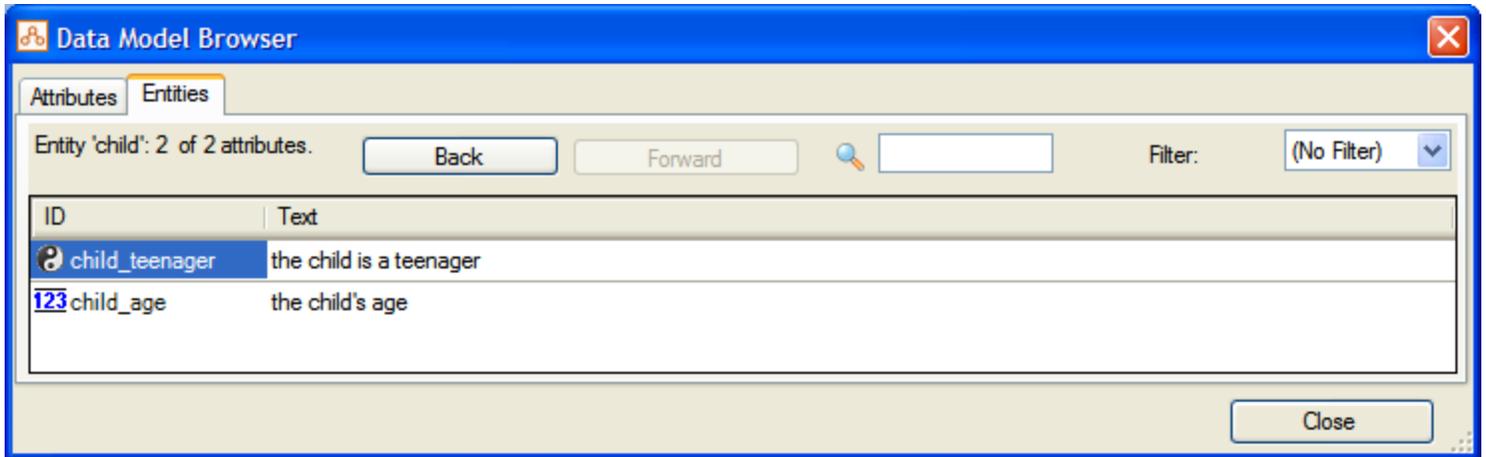


The following entity properties are displayed:

- entity ID (this is a link to the [entity attributes view](#) for that entity)
- entity text and entity type (shown by a globe for global entities and a yellow cube for non-global entities)
- identifying attribute

### View the entity attributes

The entity attributes view lists all the attributes for a particular entity. This view is accessed by clicking on an entity link in any of the other views or by selecting **Show Attributes** from the context menu in the [project entities view](#).

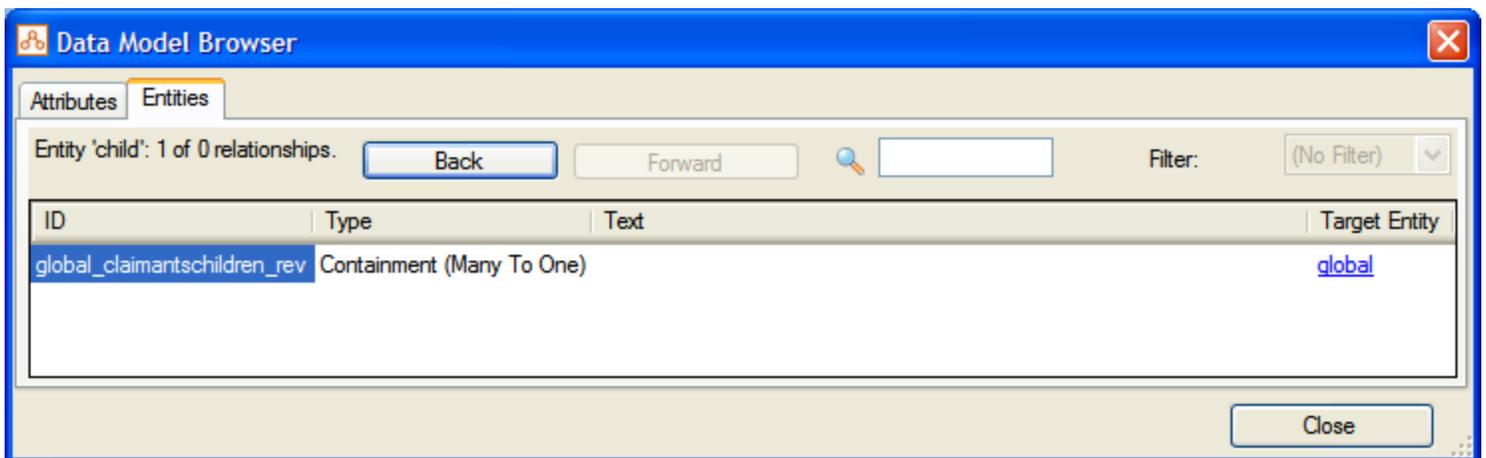


The following attribute properties are displayed:

- attribute ID (public name, if defined, otherwise build model id) and attribute type (indicated by an icon)
- attribute text

### View the entity relationships

The entity relationships view lists all the relationships for a particular entity. This view is accessed by selecting **Show Relationships** from the context menu in the project entities view.



The following relationship properties are displayed:

- relationship ID
- relationship type
- relationship text
- the target entity for the relationship (this is a link to the [entity attributes view](#) for that entity)

### Edit an attribute from within Word

To edit an attribute from within Word:

1. In the Data Model Browser, open the [project attributes view](#) or the [entity attributes view](#).
2. Right-click the attribute text and select **Edit Attribute....**The **Attribute Editor** dialog box will open for the selected attribute in Oracle Policy Modeling.
3. Edit the attribute properties as required, then click **OK**.

### Edit an entity from within Word

To edit an entity from within Word:

1. In the Data Model Browser, open the [project entities view](#).
2. Right-click the entity text and select **Edit Entity in....**The **Edit Entity** dialog box will open for the selected entity in Oracle Policy Modeling. (This option is only available for non-global entities.)
3. Edit the entity properties as required, then click **OK**.

### Edit a relationship from within Word

To edit a relationship from within Word:

1. In the Data Model Browser, open the [entity relationships view](#).
2. Right-click the relationship and select **Edit Relationship....**The **Relationship Editor** dialog box will open for the selected relationship in Oracle Policy Modeling.
3. Edit the relationship properties as required, then click **OK**.

# Temporal reasoning

## Topics in "Temporal reasoning"

- [Decide if temporal reasoning is needed](#)
- [Set the date a rule comes into effect](#)
- [Calculate an amount in a time period](#)
- [Calculate a monthly amount](#)
- [Find the maximum or minimum amount in a period](#)
- [Check if a condition is true within a time period](#)
- [Build a temporal value from entity instances](#)
- [Set the time period to use for calculations](#)
- [Determine a rule attribute on a given date](#)
- [Find the closest date when an attribute was true](#)
- [Calculate the number of days/weeks/months/years since a given date](#)
- [Check if a condition is true relative to a given date](#)

## See also:

- [Create test cases with temporal data or outcomes](#)
- [Debug temporal rules and data](#)

## Decide if temporal reasoning is needed

Temporal reasoning refers to Oracle Policy Modeling's ability to reason with rulebase attributes or outcomes whose values change over time. Rules written in Oracle Policy Modeling are thus time-aware, operating simultaneously both at a specific point in time (eg the time at which you run an investigation, or some specific point in the past or future), as well as across time periods (eg 'in the last three months', or 'until the person's 18th birthday').

When analyzing potential rulebase source material, you should take particular note of rules, data or circumstances that may change over time. Oracle Policy Modeling's temporal reasoning functionality may be the ideal choice for modeling situations that suggest changeability. Using temporal reasoning functions, even in some situations that could be modeled without them, can considerably reduce the effort needed both to write the rules and to maintain them in the future.

## What do you want to learn about?

[How conclusions can change over time](#)

[What kinds of temporal variation can Oracle Policy Modeling deal with?](#)

[Temporal reasoning and areas of change](#)

[What does temporal reasoning offer?](#)

[When to use temporal reasoning](#)

[A worked example of temporal reasoning](#)

## How conclusions can change over time

All attributes have a value. However, when you view a value (eg using the debugger or a decision report), you are only seeing the value of the attribute at a particular point in time (eg the current time, or 'the date of the investigation'). That value may change depending on when we look at it.

For instance, take a simple rule which infers whether a person can obtain a driver licence:

**the person can obtain a driver licence if**

the person has passed a driving test and

the person's age is greater than or equal to 16

Imagine that we ran an investigation using this rulebase in 2006, and provided the information that the person had passed a driving test and was born on 1 January 1992. The rulebase would infer that the person can not obtain a driver licence, as they would not be 16 years of age. However, if we saved that interview and reopened it two years later (in 2008), it would immediately tell us that the person can obtain a driver licence, because they would be 16 years old. The interview was not altered - no new information has been given, and no existing information has been changed. Yet the value of some attributes have changed, due solely to the lapse of time.

Hence, an inherent property of every attribute is its value at a point in time. Oracle Policy Modeling allows you to write rules that reach conclusions based not only on values as they exist at a particular instant, but also based on how that value changes over time.

## What drives changes in rulebase conclusions?

There are two ways in which a rulebase conclusion may change over time. The first is where the rule's outcome changes based solely on time. In this case, a conclusion can change even though the values of the rule's conditions have not. The age-based rule above is an example of this - different outcomes are reached at different times, even though the input data is always identical.

The second way in which conclusions might change over time is where the data that proves a goal itself changes. For instance, the interest rate of a bank account, or a legislatively-mandated amount of pension can change. As a result, other attributes that depend on this changeable data (eg the monthly amount of interest, the total pension payable) will also inherit different values as time passes. This differs from the age-based example above, because in this case, it is the change of information over time that dictates how the outcome value changes, not merely the passage of time by itself.

## What kinds of temporal variation can Oracle Policy Modeling deal with?

Oracle Policy Modeling includes a large number of functions to reason with the changing values of attributes over time. Some examples of rules that can be expressed are:

- Whether a particular condition is true for a given number of days/months/years in a given time period. For instance 'the employee has been sick for three or more days in the last month'.
- The total amount for a currency or numeric variable based on complex logic spanning a given time period. For instance 'the cumulative amount of interest earned on the account for the previous financial year'. Oracle Policy Modeling takes account of variations in how relevant amounts are calculated over that time (eg time periods spanning interest rate changes).
- Whether or not a condition is true, false, uncertain or unknown on, before or after a specified time or time period. For instance 'has the person been continually employed for all of the previous 12 months', or 'will the applicant be eligible on this day next year'.

## Temporal reasoning and areas of change

Temporal reasoning is used to handle three intersecting areas of change: changes in policy and rules, changes in rates and other reference data, and changes in circumstances. Common scenarios to watch out for include:

- Calculations of premiums payable by insurance companies;
- Payment of pensions or other government benefits that are affected by personal circumstances (eg unemployment, housing situation, income, age) - includes both determinations of eligibility for the benefit and also calculating the amount of payments;
- Calculation of interest rates to debtors and creditors of a financial institution;
- Calculation of taxes payable;
- Payment of salaries or wages, which may be affected by varying pay rates, overtime hours worked etc. Such data can change on a daily or even hourly basis. Temporal reasoning allows you to determine wages due over any desired time-frame (you are not tied to static, predetermined pay periods).

### **Changes in policy and rules**

Policy and legislation are constantly changing. Business rules need to keep pace with that change if they are to be useful and accurate. Temporal reasoning functionality allows you to extend a rulebase's ability to cope with changing rules beyond what can be achieved by hard coded trigger dates alone.

For example, changing social security laws may lead to the introduction of a Government benefit, or a bank may implement a tough new policy for high risk debtors. In these cases, there are likely to be certain 'trigger dates' on which new parts of a rulebase need to become active. However, there may be calculations performed over time periods which overlap these dates, or new rules may apply to new clients in a different fashion than existing clients. Thus there is a need to write rules that can handle situations where both old and new rules may have a simultaneous role in reaching the overall conclusion. Temporal reasoning allows you to do this.

### **Changes in rates and other reference data**

It is common for rulebases to feature reference data that is periodically changed. This data is generally kept either in the rulebase or an external database and is known at runtime (ie it is not user-entered data). Typically, these pieces of data take the form of rates (eg pay rates or interest rates) or thresholds (eg the minimum allowable pension payable, the monthly fee cap for a telephone plan). Oracle Policy Modeling allows you to make updates to reference data easily, while keeping deprecated or historical reference data intact. Temporal reasoning functionality then allows you to reuse a rule to calculate outcomes based on any time period, whether that period uses older, newer or a mix of reference data. Decision reports for outcomes that encompass changing reference data allow you to easily see the components of that calculation or result attributable to each reference data period.

### **Changes in circumstances**

In rulebases that calculate outcomes based on the circumstances surrounding a particular entity or group of entities (eg people, businesses), difficulties can arise when those circumstances change on a rapid (eg daily) basis. As an example, the total amount of money a health insurer pays to a customer may be dependent on the severity of the illness or injury, which can vary from day to day. Similarly, a government might pay an allowance that is affected by whether the recipient is co-habiting with someone else. If the recipient concerned is continually moving in and out of co-habitation status, it quickly becomes onerous to calculate the cumulative amount of allowance payable over, say, a year, unless temporal reasoning is used.

### What does temporal reasoning offer?

Temporal reasoning provides:

- A simple way of representing data for a period of time, over which a calculation can then be made (eg over a financial year or over the last three months);
- A simple way of showing the results of these calculations, identifying the rates or rules applied to each time period and aggregating these into a total amount for the period;
- The capacity to readily change the rules and reference data and measure the impact of that change on those affected by the rule change.

### When to use temporal reasoning

An inherent property of every attribute is its value at a point in time. Temporal operators are provided to tap into this property, including functions to calculate time-dependent items like:

- Whether a particular condition is true for a given number of days/months/years in a specific time period.
- The total amount for a data item based on complex logic spanning any given time period. For example, the total amount of a social security benefit over any given time period.
- Whether or not a condition is true on, before or after a specified time period.

These functions enable logic which is natural to a person to be captured in a readily understandable way, naturally handling conditions like the following:

- "You should have at least three alcohol-free days each week."
- "Retirement age is 55 if you started work before 1950, and 65 if you started work in or after 1950."
- "You are eligible for disability pension if you have been off work due to an injury for three consecutive months in any twelve month period."
- "Until the end of the current financial year, the levy is 1%, at which time it goes up to 2%. However, if your age at the end of the financial year is over 65, it will initially stay at 1%, and increase by 0.25% a year until it reaches 2%."

### A worked example of temporal reasoning

How to model temporal rules is best illustrated with a worked example.

#### **Pension calculation rules**

For this example, a pension payment is payable based on the following rules:

- To receive their payment, the person must satisfy an age threshold:
  - Up until 1 January 2007, this age threshold was 55 years of age;
  - From 1 January 2007 inclusive, the age threshold has changed to 65 years of age.
- The standard daily rate of a person's benefit is calculated according to the following:
  - \$5 per day regardless of marital status up until 1 July 2006;
  - After 1 July 2006, it is either:
    - \$6 per day if the person is not married; or
    - \$7 per day if the person is married.
- The actual daily rate paid to a person (the amount they are entitled to) is based on the following, regardless of which time period they fall into:
  - 1 x the standard daily rate if the person is not married;
  - 1.5 x the standard daily rate if the person is married.

## Oracle Policy Modeling rules

The business logic described above is captured in the following rules written in Microsoft Word:

### Total entitlement

**the person's total entitlement for pension for the period = IntervalDailySum(the start of the period, the end of the period, the person's daily entitlement for pension)**

### Daily entitlement

<b>the person's daily entitlement for pension</b>	
<b>the standard daily rate of benefit</b>	the person is not married and the person satisfies the age requirement
<b>the standard daily rate of benefit * 1.5</b>	the person is married and the person satisfies the age requirement
<b>0</b>	<b>otherwise</b>

### Standard daily rate

<b>the standard daily rate of benefit</b>	
<b>5</b>	TemporalBefore(2006-07-01)
<b>6</b>	TemporalOnOrAfter(2006-07-01) and the person is not married
<b>7</b>	TemporalOnOrAfter(2006-07-01) and the person is married
<b>0</b>	<b>otherwise</b>

### Age requirements

**the person satisfies the age requirement if**

both

TemporalBefore(2007-01-01) and  
the person's age in years >= 55

or

both

TemporalOnOrAfter(2007-01-01) and  
the person's age in years >= 65

### Person's age

**the person's age in years = TemporalYearsSince(the person's date of birth, the current date)**

## Simple scenario

Take a simple scenario, in which the person who will receive the pension:

- Is born on the 1st January 1950; and
- Was initially single, then married on 1 April 2007.

The assessment period is 1 January 2005 until 1 January 2020.

### Input timeline

The person is assessed over a period from 1 January 2005 until 1 January 2020, generating the following timeline for the inputs:

Date	Relevant Change	Type of Change
1 January 1950	The person is born	Circumstance
1 July 2006	Rate change for single/married people	Rate
1 January 2007	New rules for age criteria	Rules
1 April 2007	The person is married	Circumstance

### Output timeline

The inputs above generate the following results for the person. Note there is no change in result on 1 April 2007, as the person's rate does not change on that date (they do not satisfy the age requirements).

Date	Conclusion
1 January 2005	The person turns 55 The person's daily entitlement for pension is \$5 per day
1 July 2006	The person's daily entitlement for pension is \$6 per day
1 January 2007	The person's daily entitlement for pension is \$0 per day as they no longer satisfy the age requirements which have changed
1 January 2015	The person's daily entitlement for pension is \$10.50 per day

### Set the date a rule comes into effect

To apply one set of rules before a particular date, and another set of rules after that date, you can use the [Temporal Before](#) and [Temporal On Or After](#) functions.

For example, you could have the following simple rule to determine the age requirements for a pension:

**the person satisfies the age requirement if**

both

TemporalBefore(2007-01-01) and

the person's age in years >= 55

or

both

TemporalOnOrAfter(2007-01-01) and  
the person's age in years >= 65

You can also use these functions in tabular rules, for example, to determine a person's standard daily rate of benefit:

<b>the standard daily rate of benefit</b>	
<b>5</b>	TemporalBefore(2006-07-01)
<b>6</b>	TemporalOnOrAfter(2006-07-01) and the person is not married
<b>7</b>	TemporalOnOrAfter(2006-07-01) and the person is married
<b>0</b>	<b>otherwise</b>

## Calculate an amount in a time period

To calculate an amount within a time interval you use the Interval Aggregate functions. These functions aggregate the values of a time-varying attribute within a time interval, into a single value. You can also specify that the value of the attribute is only to be included in the aggregation if a given boolean attribute is true at that time.

In general the result of these functions will not vary over time, however, if time-varying start or end dates are passed in as parameters, the result will vary too.

The functions are: Interval Count Distinct, Interval Count Distinct If, Interval Daily Sum, Interval Daily Sum If, Interval Weighted Average, and Interval Weighted Average If.

## What do you want to do?

Calculate the number of distinct values for a variable in a time period

Calculate the number of distinct values for a variable in a time period only when a condition is true

Calculate the sum of a variable in a time period

Calculate the sum of a variable in a time period only when a condition is true

Calculate the average value of a variable in a time period

Calculate the average value of a variable in a time period when a condition is true

Calculate the number of distinct values for a variable in a time period

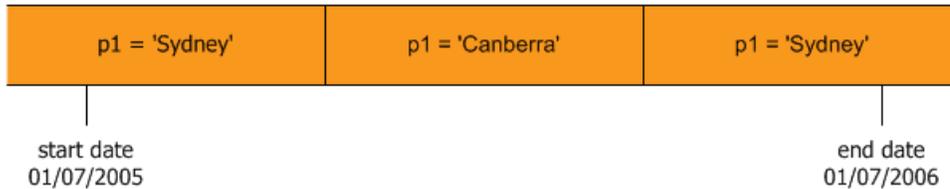
The Interval Count Distinct function counts the number of known distinct values for a variable, in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

- IntervalCountDistinct(<start date>, <end date>, <variable>)

For example, the Interval Count Distinct function could be used to determine the number of distinct addresses the client had between 1 July 2005 and 30 June 2006 (inclusive). In Word you would write this rule as:

**the client's distinct address count = IntervalCountDistinct(2005-07-01,2006-07-01,the client's address)**

This function returns a value of 2 for 'the client's address count' for the following data where p1 is 'the client's address':



Calculate the number of distinct values for a variable in a time period only when a condition is true

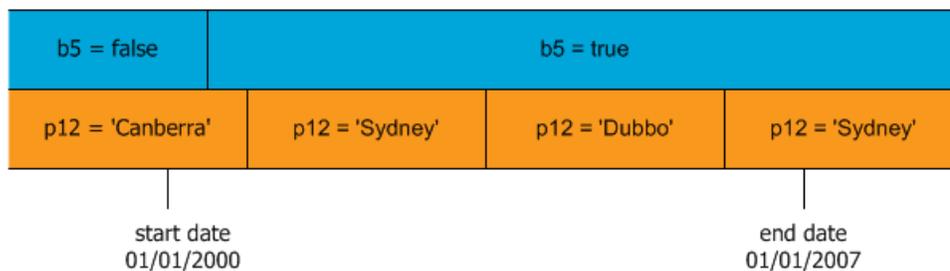
The Interval Count Distinct If function counts the number of known distinct values for an attribute, in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true. The syntax for this function is:

- IntervalCountDistinctIf(<start date>,<end date>,<variable>,<boolean filter>)

For example, the Interval Count Distinct If function could be used to determine the number of distinct addresses the client had between 1 January 2000 and 31 December 2006 (inclusive) where the client was aged over 18. In Word you would write this rule as:

**the client's distinct address count = IntervalCountDistinctIf(2000-01-01,2007-01-01,the client's address,the client is aged over 18)**

This function returns a value of 3 for 'the client's distinct address count' for the following data where b5 is 'the client is aged over 18' and p12 is 'the client's address':



Calculate the sum of a variable in a time period

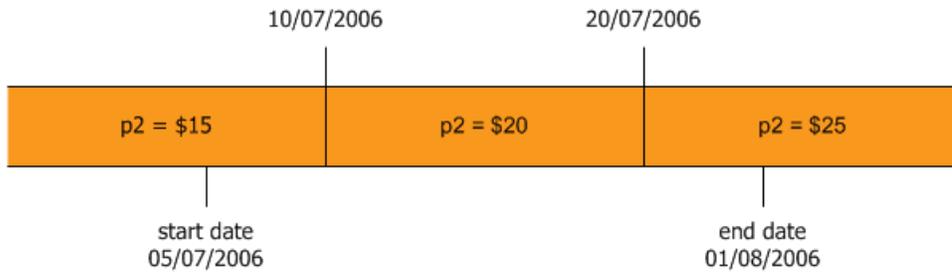
The Interval Daily Sum function calculates the sum of a currency or number variable, in the interval from the specified start date (inclusive) to the end date (exclusive). The attribute is assumed to be a daily quantity. The syntax for this function is:

- IntervalDailySum(<start date>,<end date>,<currency|number>)

For example, the Interval Daily Sum function could be used to sum the daily rate of benefit into the amount of benefit payable for the assessment period between 5 July 2006 and 31 July 2006 (inclusive). In Word you would write this rule as:

**the amount of benefit payable for the assessment period = IntervalDailySum(2006-07-05,2006-08-01,the daily rate of benefit)**

This function returns a value of \$575 for 'the amount of benefit payable for the assessment period' for the following data where p2 is 'the daily rate of benefit':



That is, \$15 \* days from 5 July 2006 to 9 July 2006 = \$15 \* 5 = \$75  
 + \$20 \* days from 10 July 2006 to 19 July 2006 = \$20 \* 10 = \$200  
 + \$25 \* days from 20 July 2006 to 31 July 2006 = \$25 \* 12 = \$300  
 Total = \$575

Calculate the sum of a variable in a time period only when a condition is true

The Interval Daily Sum If function calculates the sum of all the daily values for a currency or number variable, in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true. The syntax for this function is:

- IntervalDailySumIf(<start date>,<end date>,<currency|number>,<boolean filter>)

For example, the Interval Daily Sum If function could be used to determine the total amount spent on weekends in December 2006. In Word you would write this rule as:

**the total amount spent on weekends in December = IntervalDailySumIf(2006-12-01,2007-01-01,the daily amount spent,the day is a weekend)**

This function returns a value of \$530 for 'the total amount spent on weekends in December' for the following data:

b2 'the day is a weekend'	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true	false	true
p4 'the daily amount spent'	\$10	\$25	\$30	\$50	\$15	\$20	\$45	\$30	\$75	\$45	\$15	\$90	\$70	\$35	\$50	\$70	\$65	\$40	\$80	\$10	\$35	\$85	\$25	\$75	\$100	\$110	\$55	\$70	\$20	\$45	\$70	\$30	\$45	\$70
December 2006	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			

That is,  $\$30 + \$50 + \$45 + \$15 + \$65 + \$40 + \$75 + \$100 + \$70 + \$40 = \$530$

### Calculate the average value of a variable in a time period

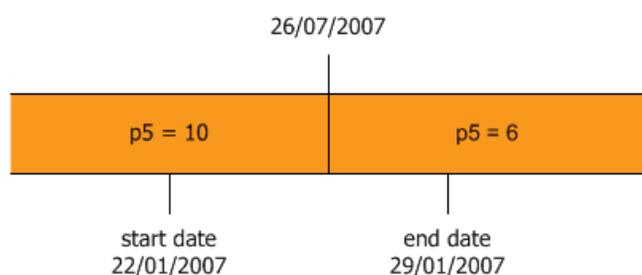
The Weighted Average function calculates the average value of a currency or number variable in the interval from the specified start date (inclusive) to the end date (exclusive) weighted by the time span to which each value applies. The syntax for this function is:

- `IntervalWeightedAverage(<start date>,<end date>,<currency|number>)`

For example, the Interval Weighted Average function could be used to determine the average number of children in care in a particular week. In Word you would write this rule as:

**the average number of children in care = IntervalWeightedAverage(2007-01-22,2007-01-29,the number of children in care)**

This function returns a value of 8.28571 for 'the average number of children in care' for the following data where p5 is 'the number of children in care':



### Calculate the average value of a variable in a time period when a condition is true

The Weighted Average If function calculates the average value of a currency or number variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true (weighted by the time span to which each value applies and where the filter is true). The syntax for this function is:

- `IntervalWeightedAverageIf(<start date>,<end date>,<currency|number>,<boolean filter>)`

For example, the Interval Weighted Average If function could be used to determine the average number of children in care for the weekdays in a specified period. In Word you would write this rule as:

**the average number of children in care for the weekdays in the assessment period = IntervalWeightedAverageIf(2007-01-22,2007-01-29,the number of children in care,the day is a weekday)**

This function returns a value of 9.2 for 'the average number of children in care for the weekdays in the assessment period' for the following data where b4 is 'the day is a weekday' and p10 is 'the number of children in care':



## Calculate a monthly amount

Temporal rules can be used to calculate an amount for each month within a specified time period. To do this you would use an Interval Aggregate function (or a Filtered Interval Aggregate function if there are dependencies on the inclusion of values in the calculation). (See [Calculate an amount in a time period](#) for more information on these functions.)

For example, a family benefit is calculated on a daily rate that is summed to give an amount that is paid monthly. To calculate the family's monthly benefit you would use the [Interval Daily Sum](#) function as follows:

**the family's monthly benefit = IntervalDailySum(the start of the payment month, the start of the following month, the family's daily benefit)**

To calculate the start of the following month, you use the Add Months function as follows:

**the start of the following month = AddMonths(the start of the payment month, 1)**

Using these rules you would need to enter the start date of the payment month as an input. It might be preferable, however, to have the start of the payment month as a time-varying attribute which gives the start date of the current month. To do this, you would add the following rule using the [Temporal Months Since](#) function:

**the start of the payment month = AddMonths(the start date, TemporalMonthsSince(the start date, the end date))**

If you wanted to calculate the start date of every month from the specified start date onwards for 20 years, you could replace 'the end date' in the rule above with a variable which calculates 20 years from the specified start date (using the Add Years function).

**the start of the payment month = AddMonths(the start date, TemporalMonthsSince(the start date, AddYears(the start date, 20)))**

Using these rules, the rulebase would return the family's monthly benefits for each month within the specified 20 year time period.

In a similar way you could calculate a weekly or yearly amount using the Add Weeks and Add Years functions in place of the Add Months functions.

## Find the maximum or minimum amount in a period

To find the maximum or the minimum amount in a specified period you use the Interval Maximum and Interval Minimum functions. There are also filtered equivalents of these functions where a value is only included in the aggregation if a given boolean attribute is true at that time. These functions are Interval Maximum If and Interval Minimum If.

In general the result of these functions will not vary over time, however, if time-varying start or end dates are passed in as parameters, the result will vary too.

### What do you want to do?

Find the maximum amount in a period

Find the minimum amount in a period

Find the maximum amount in a period when a boolean attribute is true

Find the minimum amount in a period when a boolean attribute is true

### Find the maximum amount in a period

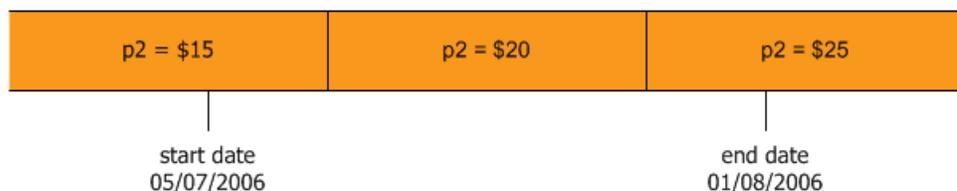
To find the maximum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive) you use the Interval Maximum function. The syntax for this function is:

- `IntervalMaximum(<start date>,<end date>,<variable>)`

For example, to determine the maximum rate of daily benefit calculated between 5 July 2006 and 31 July 2006 (inclusive), you would write the following rule in Word:

**the maximum rate of benefit during the assessment period = IntervalMaximum(2006-07-05,2006-08-01,the daily rate of benefit)**

This function returns a value of \$25 for 'the maximum rate of daily benefit during the assessment period' (p2) for the following data:



### Find the minimum amount in a period

To find the minimum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive) you use the Interval Minimum function. The syntax for this function is:

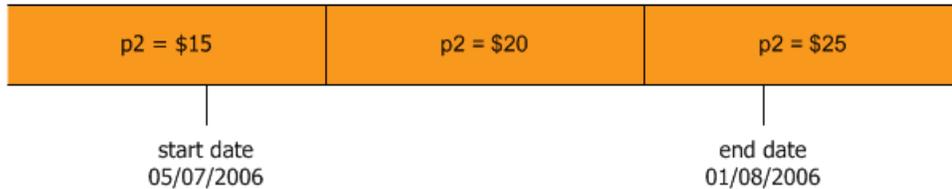
- `IntervalMinimum(<start date>,<end date>,<variable>)`

For example, to determine the minimum rate of daily benefit calculated between 5 July 2006 and 31 July 2006 (inclusive), you would write the following rule in Word:

**the minimum rate of benefit during the assessment period = IntervalMinimum(2006-07-05,2006-08-**

### 01,the daily rate of benefit)

This function returns a value of \$15 for 'the minimum rate of benefit during the assessment period' (p2) for the following data:



Find the maximum amount in a period when a boolean attribute is true

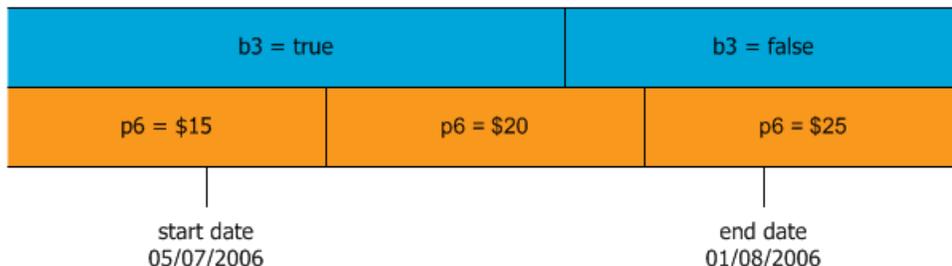
To find the maximum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true, you use the Interval Maximum If function. The syntax for this function is:

- IntervalMaximumIf(<start date>,<end date>,<variable>,<boolean filter>)

For example, to determine the maximum rate of benefit calculated between 5 July 2006 and 31 July 2006 (inclusive) where the client is also eligible for the benefit, you would write the following rule in Word:

**the maximum rate of benefit payable during the assessment period = IntervalMaximumIf(2006-07-05,2006-08-01,the maximum daily rate of benefit,the client is eligible for the benefit)**

This function returns a value of \$20 for 'the maximum rate of benefit payable during the assessment period' (p6) where b3 is 'the client is also eligible for the benefit' for the following data:



Find the minimum amount in a period when a boolean attribute is true

To find the minimum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true, you use the Interval Minimum If function. The syntax for this function is:

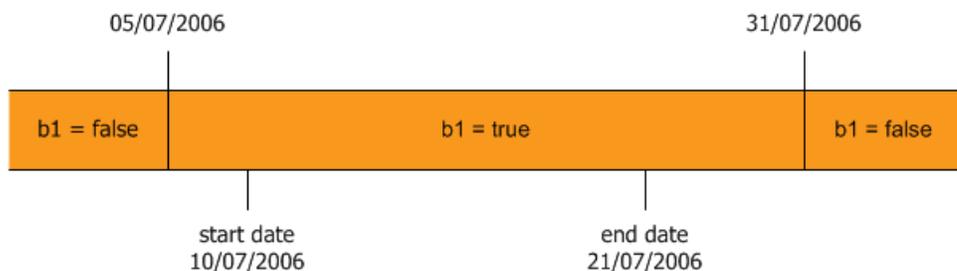
- IntervalMinimumIf(<start date>,<end date>,<variable>,<boolean filter>)

For example, to determine the minimum rate of benefit calculated between 5 July 2006 and 31 July 2006 (inclusive) for days where the client is also eligible for the benefit, you would write the following rule in Word:

**the minimum rate of benefit payable during the assessment period = IntervalMinimumIf(2006-07-05,2006-08-01,the minimum daily rate of benefit,the client is eligible for the benefit)**



The function returns a value of true for 'the client was in jail at all times during the assessment period' for the following data where b1 is 'the client was in jail':



### Check if a condition is ever true in the time period

The Interval Sometimes function returns true if and only if the attribute is ever true in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

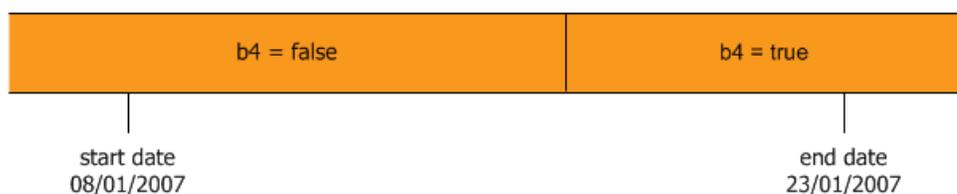
- IntervalSometimes(<start date>,<end date>,<boolean>)

For example, the Interval Sometimes function could be used to determine whether the client was in Australia at any time between 8 January 2007 and 22 January 2007 (inclusive). In Word you would write this rule as:

**the client has been in Australia if**

IntervalSometimes(2007-01-08,2007-01-23,the client was in Australia)

This function returns a value of true for 'the client has been in Australia' for the following data where b4 is 'the client was in Australia':



### Check if a condition is true for at least the specified number of days in the time period

The Interval At Least Days function returns true if and only if the attribute is true for at least the specified number of days (not necessarily consecutive) in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

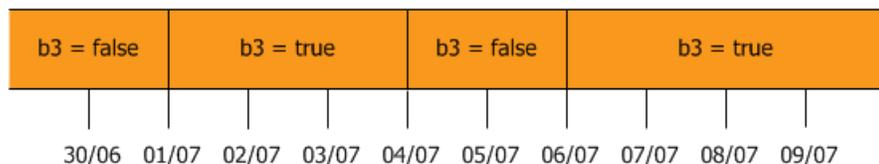
- IntervalAtLeastDays(<start date>,<end date>,<number>,<boolean>)

For example, the Interval At Least Days function could be used to determine whether an employee has been at work for at least 5 days during the assessment period. The assessment period begins on 1/7/07 and ends on 7/7/07 (inclusive). In Word you would write this rule as:

**the employee has been at work for at least 5 days during the assessment period if**

IntervalAtLeastDays(2007-07-01,2007-07-08,5,the employee was working)

The function returns a value of true for 'the employee has been at work for at least 5 days during the assessment period' for the following data where b3 is 'the employee was working':



Check if a condition is true for at least the specified number of consecutive days in the time period

The Interval Consecutive Days function returns true if and only if the attribute is true for at least the specified number of consecutive days in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

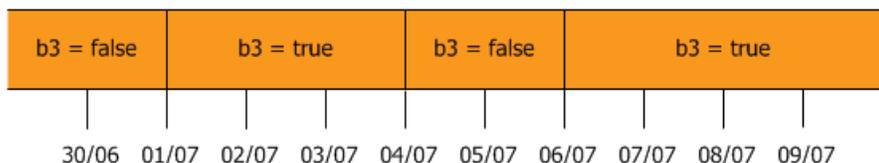
- IntervalConsecutiveDays(<start date>,<end date>,<number>,<boolean>)

For example, the Interval Consecutive Days function could be used to determine whether an employee has been at work for at least 5 consecutive days during the assessment period. The assessment period begins on 1/7/07 and ends on 7/7/07 (inclusive). In Word you would write this rule as:

**the employee has been at work for at least 5 consecutive days during the assessment period if**

IntervalConsecutiveDays(2007-07-01,2007-07-08,5,the employee was working)

This function returns a value of false for 'the employee has been at work for at least 5 consecutive days during the assessment period' for the following data where b3 is 'the employee was working':



Check if a condition is true for all of a specified number of preceding days

The Temporal Always Days function returns a boolean attribute that varies over time and is true if and only if the given boolean attribute is true for all of a specified number of preceding days, not including the current day. The syntax for this function is:

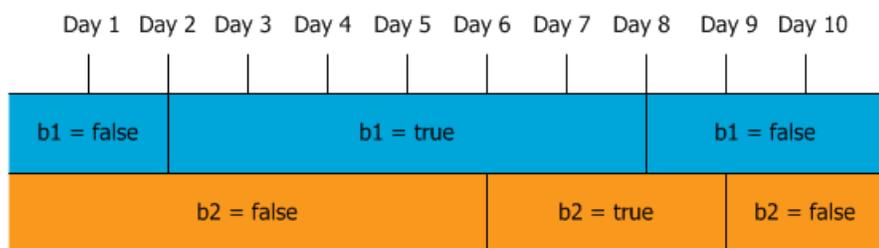
- TemporalAlwaysDays(<number>,<boolean>)

For example, the Temporal Always Days function could be used to determine whether an employee has been at work for the last 4 days. In Word you would write this rule as:

**the employee has been at work for the last 4 days if**

TemporalAlwaysDays(4,the employee was working)

If this rule is applied to the sample data below where b1 is 'the employee was working', b2 'the employee has been at work for the last 4 days' would take the following temporal result: {false, true from Day 6, false from Day 9}.



NOTE: If <number> is defined as zero, the result will always be true regardless of the value of the <boolean> parameter.

TIP: To see an example of a complete rulebase using this function, open and run the [Aged Care Approval rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### Check if a condition is true for at least the specified number of consecutive preceding days

The Temporal Consecutive Days function returns a boolean attribute that varies over time and is true if and only if the given boolean attribute is true for at least a specified number of consecutive days at any time within the preceding specified number of days, not including the current day. The syntax for this function is:

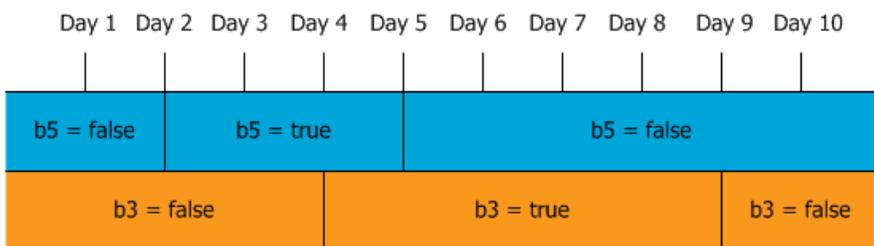
- TemporalConsecutiveDays(<mindays number>,<daycount number>,<boolean>)

For example, the Temporal Consecutive Days function could be used to determine whether a customer's bank account balance has exceeded \$50 for at least 2 consecutive days at any time in the last 5 days. In Word you would write this rule as:

**the customer's bank balance has exceeded \$50 for at least 2 consecutive days in the last 5 days if**

TemporalConsecutiveDays(2,5,the customer's bank balance exceeds \$50)

If this rule is applied to the sample data below, b3 'the customer's bank balance has exceeded \$50 for at least 2 consecutive days in the last 5 days' would take the following temporal result: {false, true from Day 4, false from Day 9}. b5 is 'the customer's bank balance exceeds \$50'.



### NOTES:

- If <mindays number> is defined as zero, the result will always be true regardless of the values of <daycount number> and the <boolean> parameter.

- b. If the <daycount number> is defined as zero, the result will be false if <mindays number> is known and is not equal to zero.
- c. If <mindays number> is greater than <daycount number>, the result will always be false regardless of the value of the <boolean> parameter.

### Check if a condition is ever true within a specified number of preceding days

The Temporal Sometimes Days function returns a boolean attribute that varies over time and is true if and only if the given boolean attribute is ever true within a specified number of preceding days, not including the current day. The syntax for this function is:

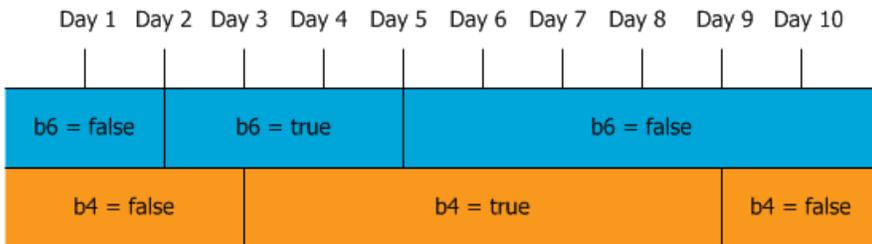
- TemporalSometimesDays(<number>,<boolean>)

For example, the Temporal Sometimes Days function could be used to determine whether a customer's bank account balance has exceeded \$100 at any time in the last 4 days. In Word you would write this rule as:

**the customer's bank balance has exceeded \$100 in the last 4 days if**

TemporalSometimesDays(4,the customer's bank balance exceeds \$100)

If this rule is applied to the sample data below, 'the customer's bank balance has exceeded \$100 in the last 4 days' (b4) would take the following temporal result: {false, true from Day 3, false from Day 9}. b6 is 'the customer's bank balance exceeds \$100'.



NOTE: If <number> is defined as a zero, the result will always be false regardless of the value of the <boolean> parameter.

### Build a temporal value from entity instances

You can convert entities into temporal attributes using the Temporal From Start Date, Temporal From End Date, and Temporal From Range functions. The entities contain a value attribute and either a start date, an end date, or both. The functions also take a default value that is applied to any uncovered periods. The default value must be a constant or expression of the same type as the value attribute, or uncertain or unknown.

### What do you want to do?

Get a temporal attribute from entity instances with values from the start date

Get a temporal attribute from entity instances with values up until the end date

Get a temporal attribute from entity instances with values from the start date until the end date

## Get a temporal attribute from entity instances with values from the start date

The Temporal From Start Date function takes a relationship, a start date attribute on the entities in the relationship, and a value attribute on the entities, and returns a single temporal attribute (at the source entity level) with values that take effect from the start date. Care should be taken that the start dates are unique, because the result will be 'uncertain' if two entities have the same start date.

The syntax for this function is:

- TemporalFromStartDate(<relationship>,<start date>,<target entity-level attribute>)

For example, if a person has various jobs over the years, you could use the Temporal From Start Date function to determine the person's employer at a given time. To do this you would write the following rule in Word:

**the person's most recent employer = TemporalFromStartDate(the person's jobs, the job's start date, the job's employer)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the person' , an entity 'the job' and a one-to-many relationship 'the person's jobs'.)

If this rule is applied to the sample data below, 'the person's most recent employer' would take the following temporal result: {uncertain, Big Bank from 12/05/1995, Superannuation Company from 24/12/2002, Insurance Agency from 03/10/2007}.

employer	start date
Big Bank	12/05/1995
Superannuation Company	24/12/2002
Insurance Agency	03/10/2007

## Get a temporal attribute from entity instances with values up until the end date

The Temporal From End Date function takes a relationship, an end date attribute on the entities in the relationship, and a value attribute on the entities, and returns a single temporal attribute (at the source entity level) with values that take effect up until the end date.

NOTE:

- If no entity has an uncertain end date then the value is uncertain after the last end date (see Example 1 below)
- If one entity has an uncertain end date, the value for that entity holds from the last specified end date onwards (see Example 2 below)
- If two entities have the same end date, the result will be 'uncertain' for a period (see Example 3 below)

The syntax for this function is:

- TemporalFromEndDate(<relationship>,<end date>,<target entity-level attribute>)

### Example 1: All end dates are specified

A person has a first aid certificate that must be renewed every year. Each certificate has an ID number that has been recorded with its expiry date. The Temporal From End Date function is used to determine the ID number that was current at any time. In Word you would write this rule as:

**the person's effective first aid certificate ID = TemporalFromEndDate(the person's first aid certificates, the first aid certificate's expiry date, the first aid certificate ID)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the person' , an entity 'the first aid certificate' and a one-to-many relationship 'the person's first aid certificates'.)

If this rule is applied to the sample data below, 'the person's effective first aid certificate ID' would take the following temporal result: {4534545A, 4943234E from 15/08/2005, 3404329F from 20/08/2006, uncertain from 12/08/2007}.

certificate ID	expiry date
4534545A	15/08/2005
4943234E	20/08/2006
3404329F	12/08/2007

### Example 2: One end date is open

In this example, one of the certificates has an open-ended expiry date.

certificate ID	expiry date
4534545A	15/08/2005
4943234E	20/08/2006
3404329F	uncertain

The intention is that this record remains in effect until further notice. When the data is collected, the expiry date for this record remains unanswered so it is initially uncertain. A rule table is required to convert the uncertain date to the latest possible date:

the first aid certificate's expiry date allowing for open ended entries	
the first aid certificate's expiry date	the first aid certificate's expiry date is certain
latest()	otherwise

The original rule then uses the expiry date allowing for open ended entries:

**the person's effective first aid certificate ID = TemporalFromEndDate(the person's first aid certificates, the first aid certificate's expiry date allowing for open ended entries, the first aid certificate ID)**

The result is that 'the person's effective first aid certificate ID' is {4534545A, 4943234E from 15/08/2005, 3404329F from 20/08/2006}.

### Example 3: Two end dates are equal (an error)

In this example, an error has been made resulting in two end dates that are equal:

certificate ID	expiry date
4534545A	15/08/2005
9984993B	20/08/2006
4943234E	20/08/2006
3404329F	12/08/2007

Using the same rules as in the previous example, the result is that 'the person's effective first aid certificate ID' is {4534545A, uncertain from 15/08/2005, 3404329F from 20/08/2006, uncertain from 12/08/2007}.

Get a temporal attribute from entity instances with values from the start date until the end date

The Temporal From Range function takes a relationship, an effective start date attribute and an expiry date on the entities in the relationship, and a value attribute on the entities, and returns a single temporal attribute (at the source entity level) with values that takes effect from the start date (inclusive) until the end date (exclusive). The value is uncertain if it expires before the next start date. Care should be taken that the start dates are unique, because the result will be 'uncertain' for a while, if two entities have the same start dates.

The syntax for this function is:

- TemporalFromRange(<relationship>,<start date>,<end date>,<target entity-level attribute>)

For example, if a person working for the Government has a certain level of security clearance (valid for a specified period), you could use the Temporal From Range function to determine the person's security clearance at a point in time. To do this you would write the following rule in Word:

**the person's effective security clearance = TemporalFromRange(the person's security clearances, the security clearance's start date, the security clearance's expiry date, the security clearance)**

(For this rule to compile the following entities and relationship must be included in a properties files in the project: an entity 'the person' , an entity 'the security clearance' and a one-to-many relationship 'the person's security clearances'.)

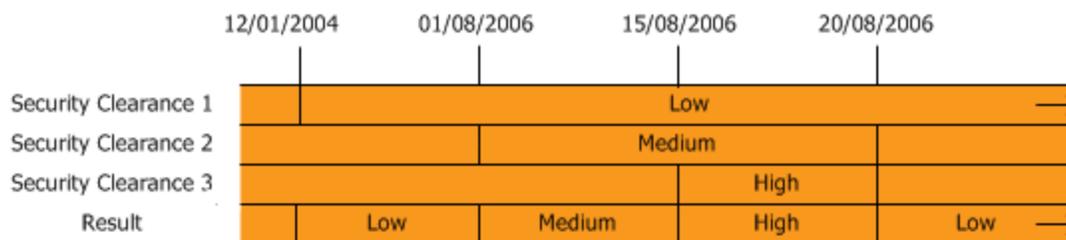
If this rule is applied to the sample data below, the result is that 'the person's effective security clearance' is {uncertain, Low from 01/07/2005, Medium from 01/07/2006, High from 01/07/2007, uncertain from 01/07/2008}.

Security Clearance	Start Date	Expiry Date
Low	01/07/2005	30/06/2006
Medium	01/07/2006	30/06/2007
High	01/07/2007	30/06/2008

In reality, a security clearance may be open-ended, and security clearances may overlap. For example:

Security Clearance	Start Date	Expiry Date
Low	12/01/2004	uncertain
Medium	01/08/2006	20/08/2006
High	15/08/2006	20/08/2006

If more than one security clearance is valid at a particular time, the one with the most recent start date applies, as seen with this example:



When this data is collected, the expiry date for the open-ended record remains unanswered, so it is initially uncertain. A rule table is required to convert the uncertain date to the latest possible date:

the security clearance's expiry date allowing for open ended clearances	
the security clearance's expiry date	the security clearance's expiry date is certain
latest()	otherwise

The original rule then uses the expiry date allowing for open ended entries:

**the person's effective security clearance = TemporalFromRange(the person's security clearances, the security clearance's start date, the security clearance's expiry date allowing for open ended clearances, the security clearance)**

The result is that 'the person's effective security clearance' is {uncertain, Low from 12/01/2004, Medium from 01/08/2006, High from 15/08/2006, Low from 20/8/2006}.

TIP: To see an example of a complete rulebase using this function, open and run the [Aged Care Approval rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

## Set the time period to use for calculations

To set the time period to use in calculations you can use the Earliest and Latest functions as start and end dates respectively. These functions allow extension of the time period under consideration, to the beginning or end of time. (Note that attempting to calculate any date differences with these values will result in uncertain.)

## What do you want to do?

[Get a date value equivalent to the earliest possible date](#)

[Get a date value equivalent to the latest possible date](#)

[Get a date value equivalent to the earliest possible date](#)

To get a date value equivalent to the earliest possible date, you use the Earliest function. This function will return a date guaranteed to be earlier than the value of any date attribute or expression.

The syntax for this function is:

- Earliest()

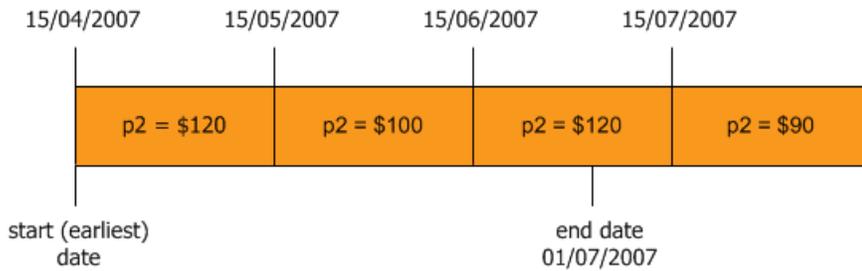
For example, an applicant is paid a benefit monthly, and the amount of the payment is a temporal value with values on specific days as follows (and 0 elsewhere):

15/4/2007: \$120, 15/5/2007: \$100, 15/6/2007: \$120, 15/7/2007: \$90

To calculate the amount of the benefit paid to the applicant up to 1/7/2007, the Earliest function is used as the start date in an [Interval Daily Sum function](#). In Word you would write this rule as:

**the amount of benefit paid to the applicant up until 1/7/2007 = IntervalDailySum(Earliest(),2007-07-01,the amount of the monthly payment)**

The result for 'the amount of benefit paid to the applicant up until 1/7/2007' would be \$340 (ie \$120+\$100+\$120) where p2 is 'the amount of the monthly payment'.



### Get a date value equivalent to the latest possible date

To get a date value equivalent to the latest possible date, you use the Latest function. This function will return a date guaranteed to be later than the value of any date attribute or expression.

The syntax for this function is:

- Latest()

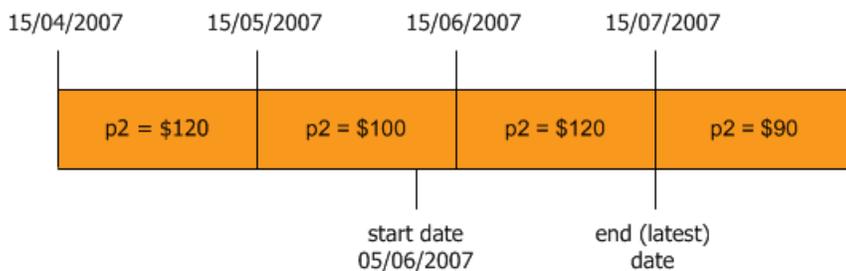
For example, an applicant is paid a benefit monthly, and the amount of the payment is a temporal value with values on specific days as follows (and 0 elsewhere):

15/4/2007: \$120, 15/5/2007: \$100, 15/6/2007: \$120, 15/7/2007: \$90

To calculate the amount of the benefit paid to the applicant since 6/5/2007, the Latest function is used as the end date in an [Interval Daily Sum function](#). In Word you would write this rule as:

**the amount of benefit paid to the applicant since 1/7/2007 = IntervalDailySum(2007-07-01, Latest(), the amount of the monthly payment)**

The result for 'the amount of benefit paid to the applicant since 1/7/2007' would be \$210 (ie \$120+\$90) where p2 is 'the amount of the monthly payment'.



### Determine a rule attribute on a given date

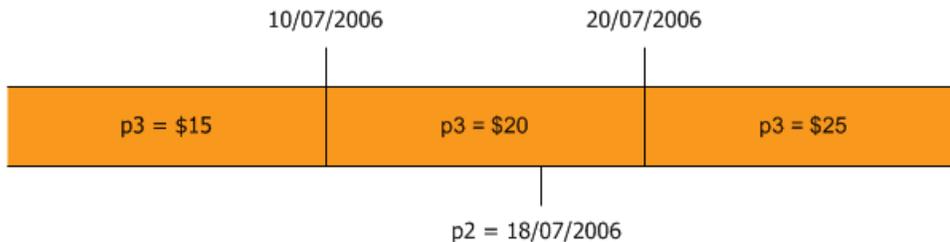
To determine the value of an attribute at a particular date you use the Value At function. This function has the following syntax:

- ValueAt(<date>, <attribute>)

For example, to determine that rate of benefit on the date of claim, you would write the following rule in Word:

**the rate of benefit payable on the date of claim = ValueAt(the date of claim,the rate of benefit)**

Using the sample data below, where 'the date of claim' (p2) is 18 July 2007, the function would return \$20 for 'the rate of benefit payable on the date of claim' (p3):



### Find the closest date when an attribute was true

To find the closest date when an attribute was true you use the When Last and When Next functions. These functions look forwards or backwards from a reference date and return a date when a specified boolean attribute is true.

### Find the date on which a boolean attribute was last true

To return the date on which a boolean attribute was last true, looking backwards from a reference date (including the reference date), you use the When Last function. This function has the following syntax:

- WhenLast(<date>,<boolean>)

For example, to determine when a customer's bank balance was last over \$100, you would write the following rule in Word:

**the date the customer's bank balance was last over \$100 = WhenLast(the current date,the customer's bank balance > 100)**

### Find the date on which a boolean attribute will next be true

To return the date on which a boolean attribute will next be true, looking forwards from a reference date (including the reference date), you use the When Next function. This function has the following syntax:

- WhenNext(<date>,<boolean>)

For example, to determine when was the first time in 2007 that a customer's bank balance was over \$100, you would write the following rule in Word:

**the date the customer's bank balance was over \$100 for the first time in 2007= WhenNext(2007-01-01,the customer's bank balance > 100)**

TIP: To see an example of a complete rulebase using this function, open and run the [Aged Care Approval rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

## Calculate the number of days/weeks/months/years since a given date

To calculate the number of days/weeks/months/years since a given date you use the following Temporal Since functions: Temporal Days Since, Temporal Weeks Since, Temporal Months Since, and Temporal Years Since.

The calculation stops by a given (exclusive) end date.

### What do you want to do?

Calculate the number of days since a given date

Calculate the number of weeks since a given date

Calculate the number of months since a given date

Calculate the number of years since a given date

Calculate the weekdays in a given time period

Calculate a specific day in a month for a given time period

### Calculate the number of days since a given date

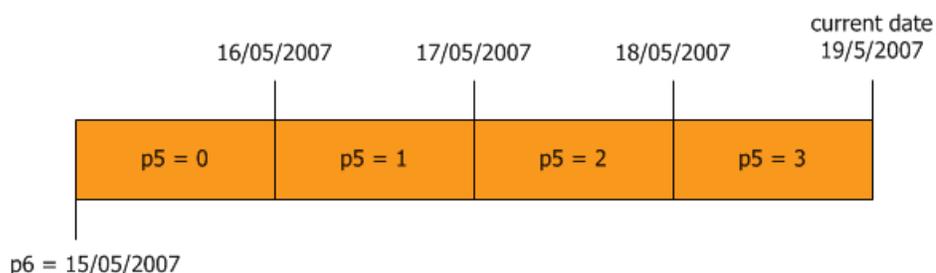
To calculate the number of full days since a given date, you use the Temporal Days Since function. Note that this function will return a number variable that varies every day. The function has the following syntax:

- TemporalDaysSince(<start date>, <end date>)

For example, to determine the number of days since it has rained, you would write this rule in Word:

**the number of days since it has rained = TemporalDaysSince(the date of the most recent rainfall, the current date)**

The function returns a temporal value with the number of days incrementing on the date of each daily change point. Where 'the date of the most recent rainfall' (p6) is 15 May 2007, the calculation of 'the number of days since it has rained' (p5) is shown in the diagram below:



### Calculate the number of weeks since a given date

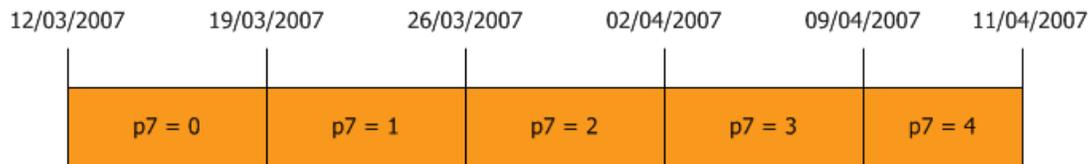
To calculate the number of full weeks since a given date, you use the Temporal Weeks Since function. Note that this function will return a number variable that varies every week. The function has the following syntax:

- TemporalWeeksSince(<start date>, <end date>)

For example, to determine the number of weeks in the assessment period where the start date is 12 March 2007 and the end date is 11 April 2007 you would write this rule in Word:

**the number of weeks in the assessment period = TemporalWeeksSince(2007-03-12,2007-04-11)**

The function returns a temporal value with the number of weeks incrementing on the date of each weekly change point. This is shown in the diagram below (p7 is 'the number of weeks in the assessment period'):



Calculate the number of months since a given date

To calculate the number of full months since a given date, you use the Temporal Months Since function. Note that this function will return a number variable that varies every month. The function has the following syntax:

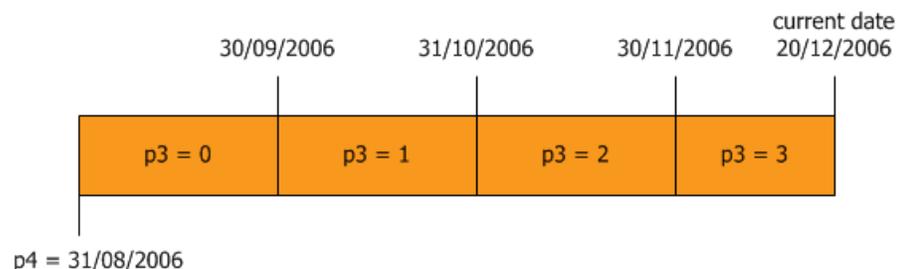
- TemporalMonthsSince(<start date>,<end date>)

For example, to determine the number of months a mobile phone contract has been in effect, you would write this rule in Word:

**the number of months the mobile phone contract has been in effect = TemporalMonthsSince(the start date of the mobile phone contract,the current date)**

The function returns a temporal value with the number of months incrementing on the date of each monthly change point. NOTE: Where the supplied date is after the 28th day of the month, and a subsequent month has fewer days than the supplied month, the change point for the anniversary month will be created on the last day of that month. For example, if the supplied date is 28, 29, 30 or 31 January 2007, the first change point will be 28 February 2007.

The earlier example is shown in the diagram below where 'the start date of the mobile phone contract' (p4) is 31 August 2006, and the current date is 20 December 2006 (p3 is 'the number of months the mobile phone contract has been in effect'):



Calculate the number of years since a given date

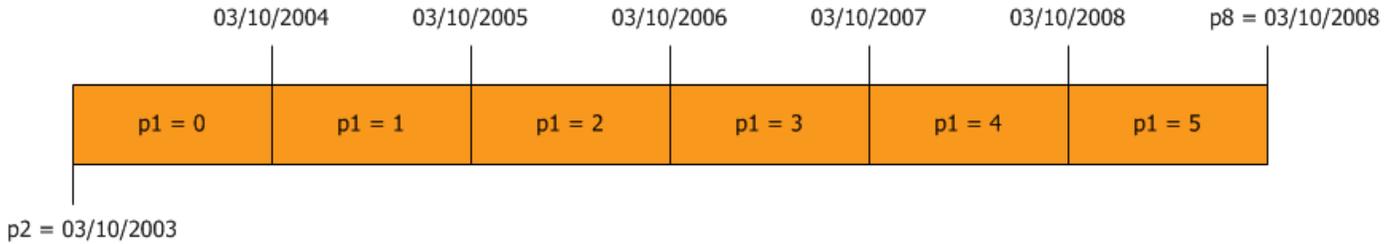
To calculate the number of full years since a given date, you use the Temporal Years Since function. Note that this function will return a number variable that varies every year. The function has the following syntax:

- TemporalYearsSince(<start date>,<end date>)

For example, to determine the child's age up to the child's fifth birthday, you would write this rule in Word:

**the child's age = TemporalYearsSince(the child's date of birth,the child's fifth birthday)**

The function returns a temporal value with the number of years incrementing on the date of each annual change point. This is shown in the diagram below where 'the child's date of birth' (p2) is 03 October 2003 (p1 is 'the child's age', and p8 is 'the child's fifth birthday'):



Calculate the weekdays in a given time period

The Temporal Is Weekday function returns true on dates that are weekdays and false on dates that are weekends from the specified start date (inclusive) to the end date (exclusive). Note that this function will return uncertain outside of the date range. The syntax for this function is:

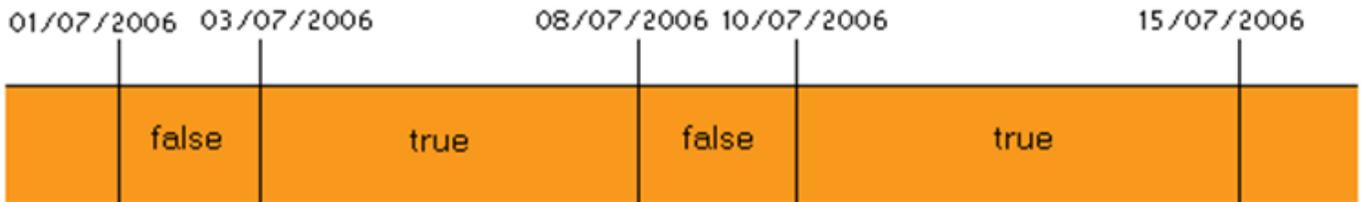
- TemporalIsWeekday(<start date>, <end date>)

For example, the Temporal Is Weekday function could be used to determine if an applicant is receiving money on a given day when that person is receiving money each weekday between 1 July 2006 and 15 July 2006. In Word you would write this rule as:

**the applicant receives money if**

TemporalIsWeekday(2006-07-01, 2006-07-15)

The function returns a value of true for dates that are weekdays and false for the dates that are weekends:



Calculate a specific day in a month for a given time period

The Temporal Once Per Month function returns true if the day is equal to the day-of-month parameter and false on all other days of the month from the specified start date (inclusive) to the end date (exclusive). Note that this function will return uncertain outside of the date range. When the day-of-month exceeds the number of days in the current month, the value is true on the last day of that month. Therefore the function returns a value that is true exactly one day per month. The syntax for this function is:

- TemporalOncePerMonth(<start date>, <end date>, <day-of-month>)

For example, the Temporal Once Per Month function could be used to calculate the allowance given to an applicant who is receiving an allowance on the 15th of every month between 1 July 2006 and August 31, 2006. In Word you would write this rule as:

**the applicant receives an allowance if**

`TemporalOncePerMonth(2006-07-01, 2006-08-31, 15)`

The function returns a value of true for dates that are equal to the specified day of the month and false for all other dates:



## Check if a condition is true relative to a given date

To check if a boolean attribute is true relative to a given date you use the following Temporal Date functions: Temporal Before, Temporal After, Temporal On, Temporal On Or Before, Temporal On Or After.

### What do you want to do?

Check if a condition is true before a given date and false on and afterwards

Check if a condition is true after a given date and false on and before

Check if a condition is true on a given date and false before and afterwards

Check if a condition is true on and before a given date and false afterwards

Check if a condition is true on or after a given date and false before

### Check if a condition is true before a given date and false on and afterwards

The Temporal Before function returns a boolean attribute that varies over time and is true before a given date and false on and afterwards. The syntax for this function is:

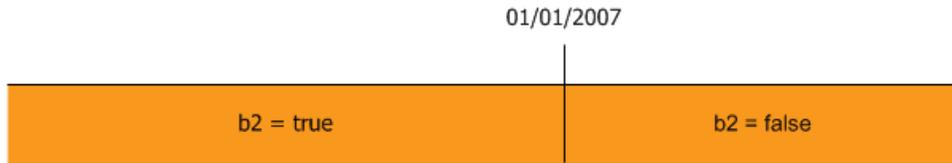
- `TemporalBefore(<date>)`

For example, the Temporal Before function could be used to determine if the pre-2007 Ministerial Determination is in force (this was in force before 1/1/2007). In Word you would write this rule as:

**the pre-2007 Ministerial Determination is in force if**

`TemporalBefore(2007-01-01)`

As the diagram below illustrates, 'the pre-2007 Ministerial Determination is in force' (b2) is true before the given date (1/1/2007) and false on and after that date.



TIP: To see an example of a complete rulebase using this function in combination with the Temporal On Or After function, open and run the [Aged Care Approval rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

Check if a condition is true after a given date and false on and before

The Temporal After function returns a boolean attribute that varies over time and is true after a given date and false on and before. The syntax for this function is:

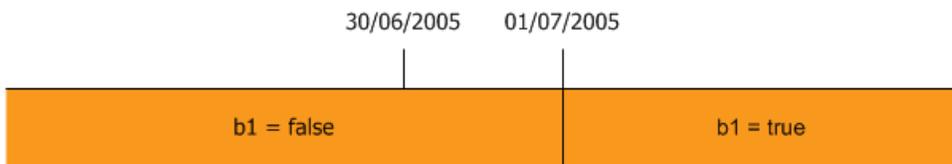
- TemporalAfter(<date>)

For example, the Temporal After function could be used to determine if the July 2005 rate changes apply (these rates take effect after 30/6/2005). In Word you would write this rule as:

**the July 2005 rate changes apply if**

TemporalAfter(2005-06-30)

As the diagram below illustrates, 'the July 2005 rate changes apply' (b1) is false up to and on the given date (30/6/2005) and true after that date (ie from 1/7/2005 onwards).



Check if a condition is true on a given date and false before and afterwards

The Temporal On function returns a boolean attribute that varies over time and is true on a given date and false before and afterwards. The syntax for this function is:

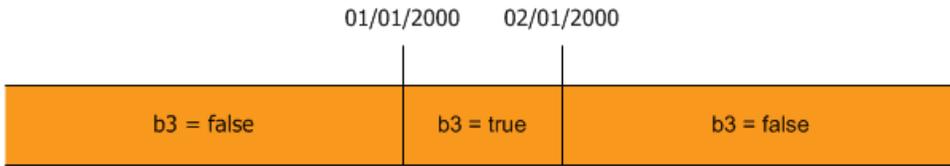
- TemporalOn(<date>)

For example, the Temporal On function could be used to determine if the New Millennium Promotion is available to customers (this promotion is only offered on 1/1/2000). In Word you would write this rule as:

**the New Millennium Promotion is available to customers if**

TemporalOn(2000-01-01)

As the diagram below illustrates, 'the New Millennium Promotion is available to customers' (b3) is only true on the given date (1/1/2000) and false before and after that date.



Check if a condition is true on and before a given date and false afterwards

The Temporal On Or Before function returns a boolean attribute that varies over time and is true on and before a given date and false afterwards. The syntax for this function is:

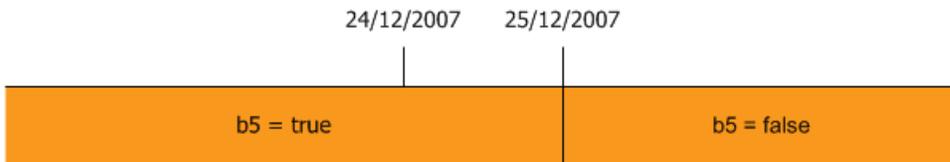
- TemporalOnOrBefore(<date>)

For example, the Temporal On Or Before function could be used to determine if the pre-Christmas price list applies (it applies up to 24/12/2007). In Word you would write this rule as:

**the pre-Christmas price list applies if**

TemporalOnOrBefore(2007-12-24)

As the diagram below illustrates, 'the pre-Christmas price list applies' (b5) is true up to and including the given date (24/12/2007) and false after that date (ie from 25/12/2007 onwards).



Check if a condition is true on or after a given date and false before

The Temporal On Or After function returns a boolean attribute that varies over time and is true on or after a given date and false before. The syntax for this function is:

- TemporalOnOrAfter(<date>)

For example, the Temporal On Or After function could be used to determine if the 2007 Ministerial Determination is in force (in force from 1/1/2007). In Word you would write this rule as:

**the 2007 Ministerial Determination is in force if**

TemporalOnOrAfter(2007-01-01)

As the diagram below illustrates, 'the 2007 Ministerial Determination is in force' (b4) is false before the given date (1/1/2007) and true on and after that date.

01/01/2007



TIP: To see an example of a complete rulebase using this function in combination with the Temporal Before function, open and run the [Aged Care Approval rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

# Interviews and flows

## Topics in "Interviews and flows"

- [Design an interview](#)
- [Create, modify or delete a screens file](#)
- [Create, modify or delete a question screen](#)
- [Collect information about entity instances](#)
- [Customize interview user input options](#)
- [Decide whether to allow uncertainty in user answers](#)
- [Hide, display and disable an interview screen element](#)
- [Tutorial: Hiding and displaying summary screen elements](#)
- [Change the text of an interview question or sentence](#)
- [Change the layout or appearance of interview screens](#)
- [Customize Oracle Web Determinations](#)
- [Define interview screen order](#)
- [Define interview screen flow](#)
- [Change how interview data is summarized and reviewed](#)
- [Check attribute inclusion on interview screens](#)
- [Create, update or delete interview help](#)
- [Overview: The process of creating an interview document](#)
- [Create, update or delete an interview document](#)
- [Develop a template for an interview document](#)
- [Test an interview or screen flow](#)

## See also:

- [Validate user input using errors and warnings](#)
- [Deploy an interview to Web Determinations](#)
- [Deploy a rulebase or interview to Determinations Server](#)

## Design an interview

If your rules will be deployed in an interactive software application, defining an interview allows you to specify the way in which users will interact with the rulebase.

When the rulebase is run, the Oracle Determinations Engine collects information in order to find a value for the specified goal (see [Oracle Determinations Engine and the Inference Cycle](#) for more on this process). The interview you define specifies the user's experience while providing this information and reviewing the conclusions reached by the rules. The aim of your interview design is to provide users with a logical and easy-to-understand interface to your rulebase.

## Interview features

There are many options available that allow you to design a user's experience of your rulebase in the best possible way. Consider the options available and allow some time to identify the characteristics that your users would find most effective and usable in interacting with your rulebase.

Options you can use in the design of your interview include:

- [Grouping of questions into separate screens](#). This allows you to group logically related attributes so that values are entered for them at the same. For example, a group of attributes collecting income from different sources may be defined on a single "Income" screen.
- Controlling the order of the question screens shown in the interview. By default, the order in which question screens are displayed to the user will be driven by the question search. This will collect information in the most efficient manner, but may not provide your users with interview navigation that they find predictable and intuitive. For more control over the question screen navigation, you can specify the [order](#) that you would like screens to appear in, or you can also define a precise [screen flow](#), which provides even tighter control over the display of question screens and can be used to mirror application forms.
- Using the [known operator](#) in your rules to collect certain base data first to feed into the investigation of the substantive rule model. Certain attributes might play a "streaming" role, providing sufficient information to infer conclusions across the rule model quickly without asking redundant questions.
- Controlling what users see when they first start the interview, and how they are guided into investigating the rulebase, as well as [how the conclusions of the rulebase are displayed](#) to them.
- Options [controlling the answers that users can provide](#) for individual questions. For example, you may wish to provide a drop-down list of options that a user must select from, or to restrict the highest value that can be entered for a particular question.
- Options controlling the [layout or appearance of questions and screens](#). For example, you may wish to emphasize some parts of your screen text in bold font.
- Additional [integrated help text](#), linked to questions, screens or key terms, to help users clearly understand the interview and how to answer questions.
- Ability to produce a [printable report or form](#) with results and data from the interview.

Most interview elements are defined in a [screens file](#). This contains details of [questions screens](#) and [questions](#), as well as the [summary screen](#), which guides users through available goals to investigate and presents conclusions. Specific [screen orders](#) for your question screens, and [screen flows](#) are also defined in the screens file. Other interview features, such as [integrated help files](#) or [interview documents](#), are added as additional files and linked to from within the screens file.

To see an example of a complete rulebase with many interview features, open and [run](#) the [Social Services Screening example rulebase](#) project provided in the Examples folder in the Oracle Policy Modeling installation folder.

### See also:

- [Create, modify or delete a screens file](#)
- [Test an interview or screen flow](#)

## Create, modify or delete a screens file

Oracle Policy Modeling has integrated screen development tools which allow you to develop software applications around your rule model to produce efficient, streamlined interview paths.

The first step to building screens for your rulebase application is to add a screens file to your project. You can have one or more screens files as is convenient – on building all screens in the project will be brought together to form a single screen definition file.

## What do you want to do?

[Create a screens file](#)

[Modify a screens file](#)

[Delete a screens file](#)

### Create a screens file

To add a new screens file to your project:

1. In Oracle Policy Modeling, right-click the **Interviews** folder in the Project Explorer and select **Add New Screens File**.  
A new Screens file will be added to your project. The new file will be selected and highlighted in the list.
2. Type a name for your screens file, for example, "Screens".
3. Save your project by selecting **File | Save All**.

### Modify a screens file

To open your screens file for editing, double-click on the file in the Project Explorer.

### Organize a screens file

Folders can be added to your screens file to help organize your screens. (By default, the first screens file that is added to a project will contain a Questions Screens folder and a Documents folder.)

To add additional folders to your screens file:

1. Right-click the \*.xint filename, or another folder, in the screens view.
2. Select **New Folder** from the pop-up menu.
3. Enter an appropriate name for your screen folder.

### Edit a screens file

In your screens file you can define and edit:

- [Questions screens](#)
- [Summary screens](#)
- [Screen orders](#)
- [Screen flows](#)
- [Interview documents](#)

### Delete a screens file

To delete a question screens file:

1. In the Project Explorer in Oracle Policy Modeling, right-click the screens file and select **Delete**.
2. Click **OK** to confirm the permanent deletion.

TIP: To only remove the file from your Oracle Policy Modeling project (but not delete it from your file system as well), right-click it in Oracle Policy Modeling and select **Remove from Project**.

## Create, modify or delete a question screen

A question screen is a screen displayed to the user during an assessment to collect data. It contains question text and answer fields. Building question screens allows you to group attributes onto single views and provide the user with a sense of context by adding labels and headings to those screens. Used with Oracle Web Determinations, these screens can also display [integrated help](#) for each screen question. Question screens are created in a [screens file](#) in Oracle Policy Modeling.

### What do you want to do?

[Create a question screens folder](#)

[Create a question screen](#)

[Add questions to screens](#)

[Add labels to question screens](#)

[Create a screen attribute](#)

[Preview a question screen in Oracle Web Determinations](#)

[Modify a question screen](#)

[Find a question screen](#)

[Delete a question screen](#)

[Organize question screens within a folder](#)

### Create a question screens folder

By default, the first screens file that is added to a project will contain a Questions Screens folder. To add additional folders to your screens file:

1. Right-click the \*.xint filename, or another folder, in the screens view.
2. Select **New Folder** from the pop-up menu.
3. Enter an appropriate name for your screen folder.

TIP: Question screen folders in a [screen order](#) are used to define the 'stages' or groupings that are displayed at the top of an interview to indicate progress through the investigation.

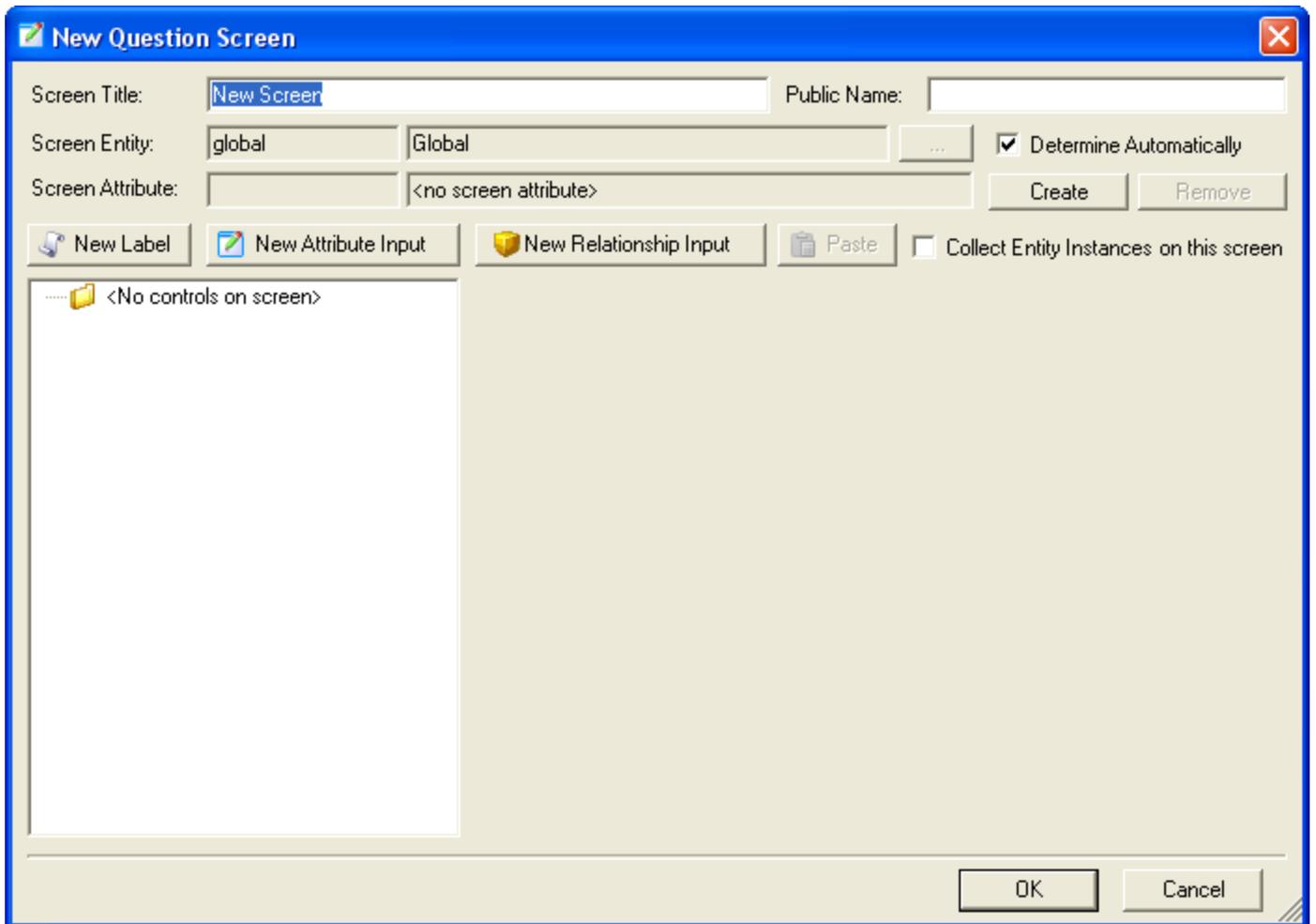
Once you have defined folders, you can add new screens to them.

### Create a question screen

To create a question screen:

1. Right-click the **Question Screens** folder in your screens file.
2. Select **New Question Screen** from the pop-up menu.

The following dialog will appear:



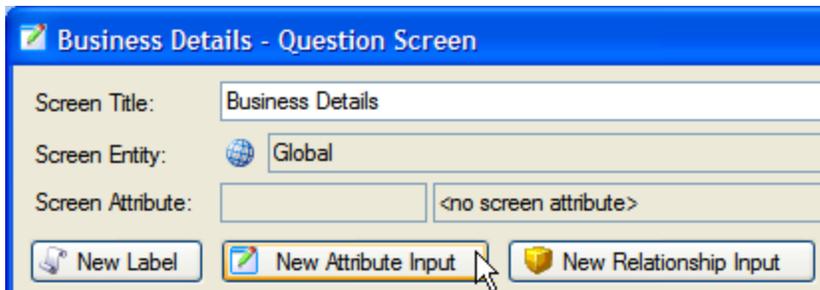
3. Enter an appropriate name for the screen in the **Screen Title** text box.  
TIP: Screen names are used as the first heading on a screen. If you use a clear screen name which makes it easy to understand the point of the current screen, the user will be much more receptive to the application. Screen names should be descriptive of the logical group or purpose of questions contained on that screen. You can use substitution in the screen name. For more information, see [Substitute an attribute value into the text on screens](#).
4. Click **OK** to apply the name change.
5. Save the screens file by selecting **File | Save <Screens\_File\_Name>.xint** from the main menu in Oracle Policy Modeling.

### Add questions to screens

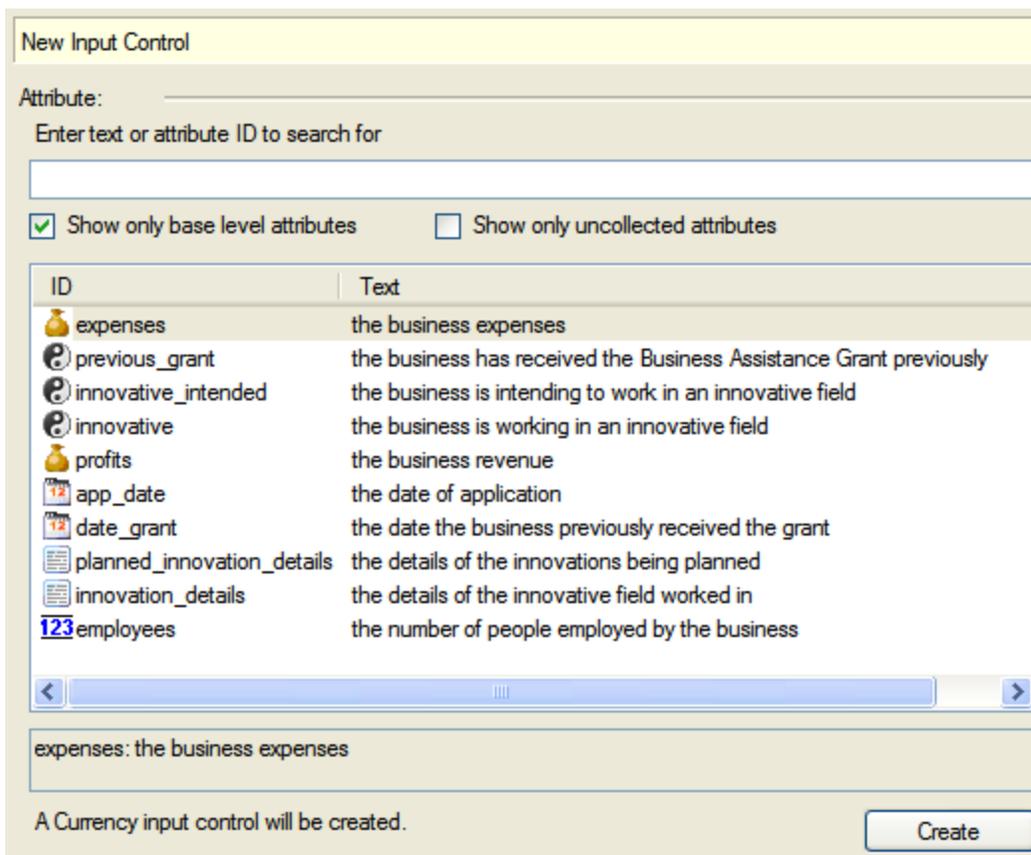
There are two steps to adding a question to your screen: adding a new input control and customizing the attribute input control.

To add a new input (question) control to your screen:

1. Double click your question screen in your screens file to open it for editing.
2. Click the **New Attribute Input** button in the question screen dialog.



The right hand side of the question screen dialog will show the following controls:



3. Select the attribute you want to place on the screen as a question. NOTE: You can filter the attribute list by typing in the text box above. You can also limit the list to only showing base level attributes and/or only showing attributes which are not already on a question screen by using the check boxes provided.
4. Click the **Create** button (or simply double-click the attribute in the list) to add it as a question on the screen.

To customize the attribute input control:

1. Select the attribute on the left hand side of the screen. The right side of the question screen edit dialog will be replaced with a set of edit controls for the question based on its data type.

Attribute Input Control

Down Up Cut Copy Delete

Attribute:

Appearance:

Question Text  Free Form Text

What is the date of application?

Is HTML

CSS Class:

CSS Style:

Input Type:

Style:

Input type:

Dynamic Default Value:

Attribute:

Value:

Default Value:

(yyyy-mm-dd)

Visibility:

Attribute:

Default State:  (if attribute is unknown or uncertain)

Read-Only:

Attribute:

Default State:  (if attribute is unknown or uncertain)

Mandatory:

Attribute:

Default State:  (if attribute is unknown or uncertain)

Here you can customize questions and customize user input options. You can also change the appearance of text and controls.

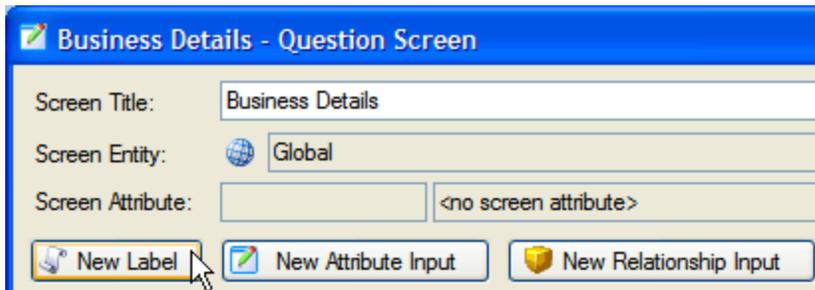
2. Click **OK** to apply any changes you have made to the screen. This will close the screen edit dialog. To save the changes permanently to your screen file, you will need to save it from the main menu in Oracle Policy Modeling.

## Add labels to question screens

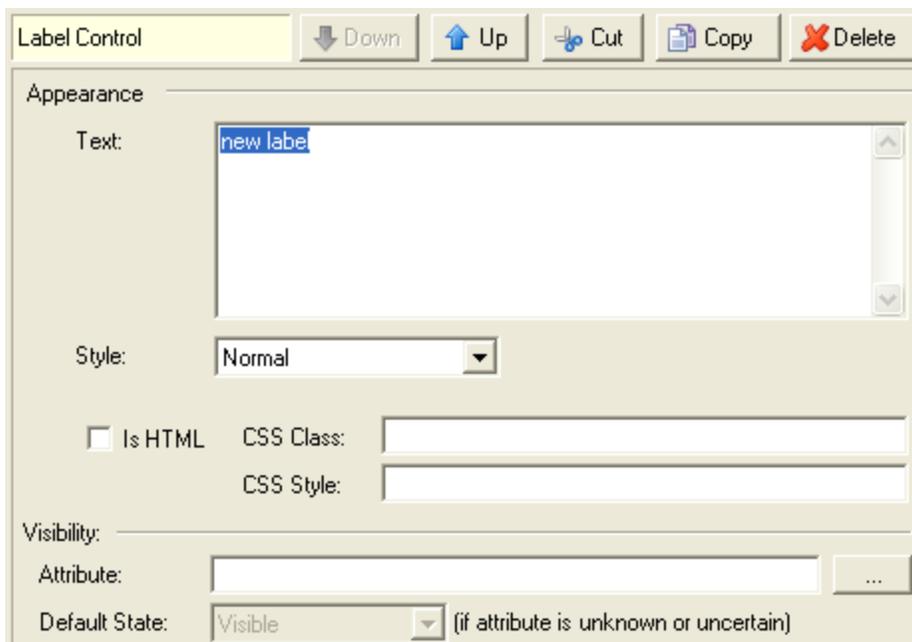
Labels provide a means for assisting the user to understand the context within which questions are being asked.

To add a new label to your screen:

1. Open your question screen in your screens file.
2. Click the **New Label** button in the question screen dialog.



The right hand side of the question screen dialog will display options for the label.



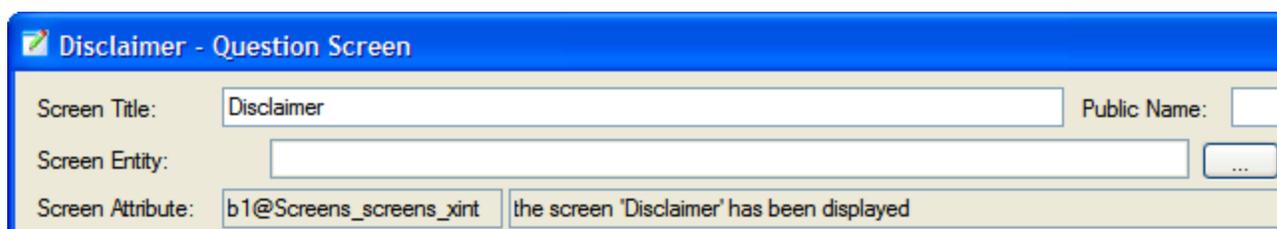
3. Enter the **Text** for the label.
4. Select a **Style** for the label from the drop-down list (eg Normal, Heading 1, Heading 2, Heading 3 etc).  
TIP: Generally, each screen should have a Heading 1 style label at the top. Additional heading styles may be used to break questions into sub-groups, and Normal style labels can be added to provide additional information and context.
5. If required, select the **Is HTML** checkbox. See [Change the appearance of text](#) for more information on this setting.
6. If required, enter a **CSS Class** and/or **CSS Style**. See [Change the appearance of a control](#) for more information on this setting.

## Create a screen attribute

A screen attribute is an attribute in the rulebase that is associated with a question screen and is used to prompt the display of the screen. Screen attributes are not displayed to the user but are given a value of 'true' automatically once the question screen has been displayed during an interview.

To create a screen attribute:

1. Open your question screen in your screens file.
2. Click the **Create** button next to the Screen Attribute field. The id and text of the automatically created screen attribute will be shown:



The screenshot shows a dialog box titled "Disclaimer - Question Screen". It has three main fields: "Screen Title" with the value "Disclaimer", "Screen Entity" which is empty, and "Screen Attribute" which contains "b1@Screens\_screens\_xint" and "the screen 'Disclaimer' has been displayed". There is a "Public Name" field which is also empty. A "..." button is visible next to the "Screen Entity" field.

3. Click **OK**.

You can now use this screen attribute in your rules. For example:

**the interview is complete if**

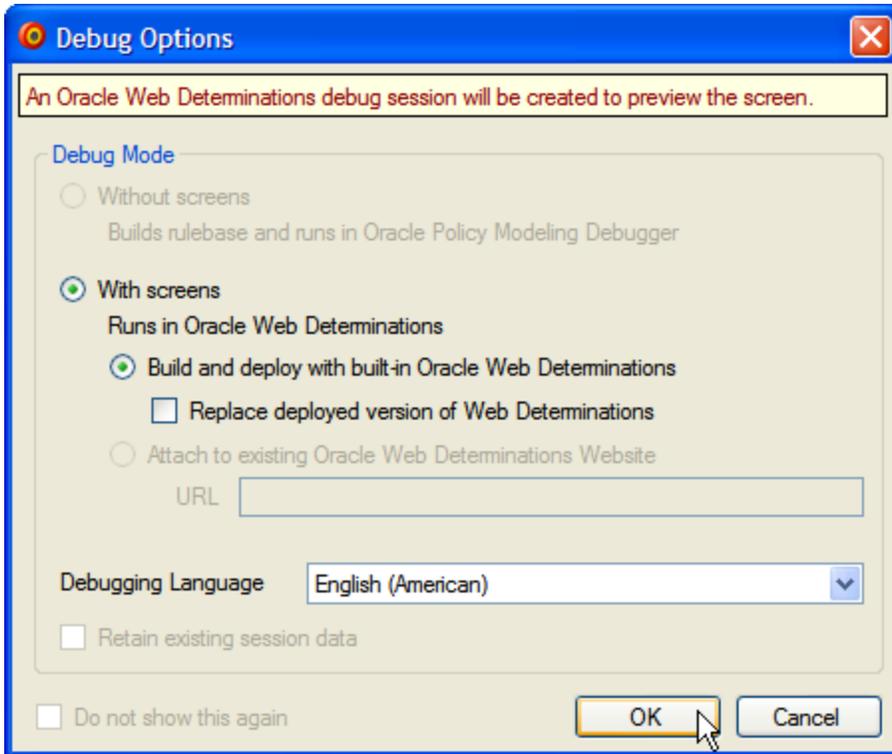
the screen 'Disclaimer' has been displayed and  
it is known whether or not the claimant is eligible for the benefit

NOTE: You need to ensure that the text of the screen attribute in your rule is identical to the text of the screen attribute in the screens file to ensure that the attribute created locally in your Word document links to the one in the screens file.

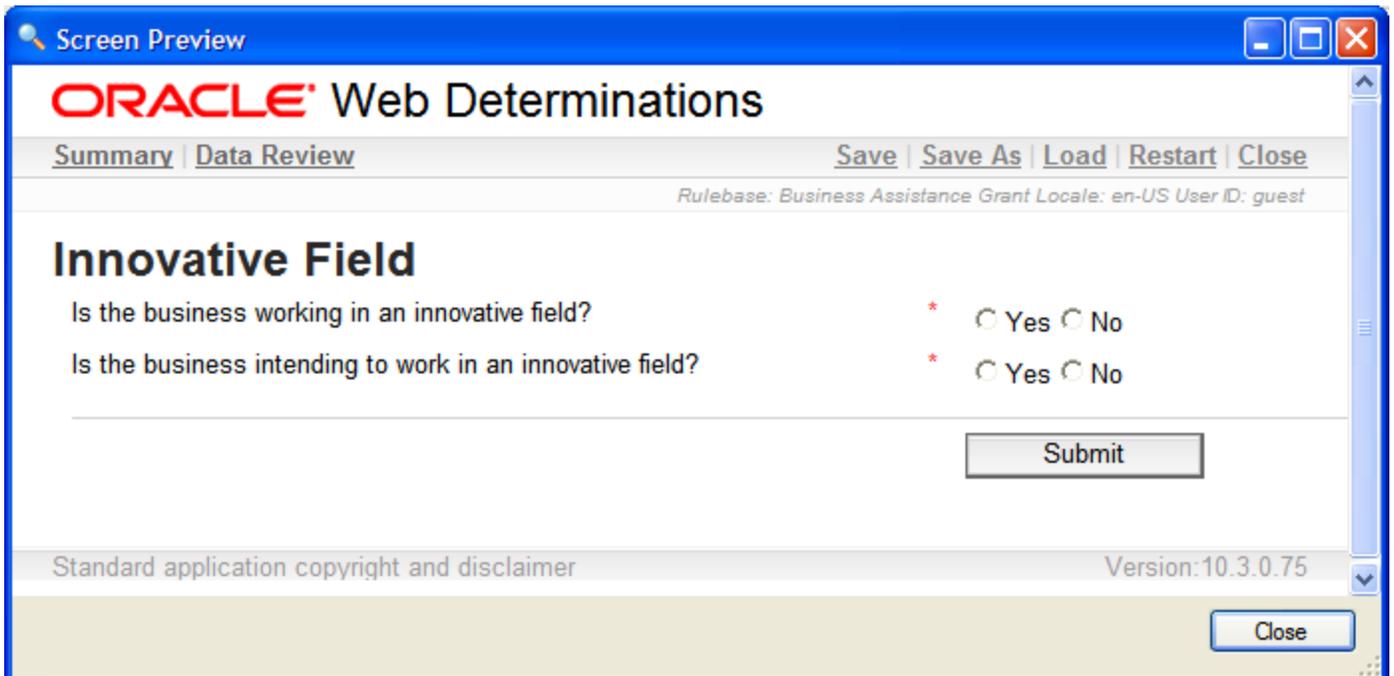
## Preview a question screen in Oracle Web Determinations

While you are creating and reviewing your question screens, you may wish to check the layout and formatting in the context of how they will appear in Oracle Web Determinations.

1. In your screens file, right click on the question screen you wish to preview and select **Preview in Web Determinations**. Alternatively, click the **Preview** button at the bottom left of the screen editor, if your screen is open.
2. The **Debug Options** dialog is displayed if no current debug session exists. Select the [debug options](#) you wish to use to preview the screen in Oracle Web Determinations and click **OK**.



- The **Screen Preview** window is displayed showing how your question screen will appear in Oracle Web Determinations, including any customizations that have been made to Oracle Web Determinations. The buttons and links are disabled, as this window is to preview your question screen only and is not a fully functional interview.



TIP: You can [import data from a debug session](#) prior to using the Preview option, and this data will be used in the screen preview and will allow you to check [substitution](#), [visibility attributes](#) and [other screen display elements](#) on your question screen.

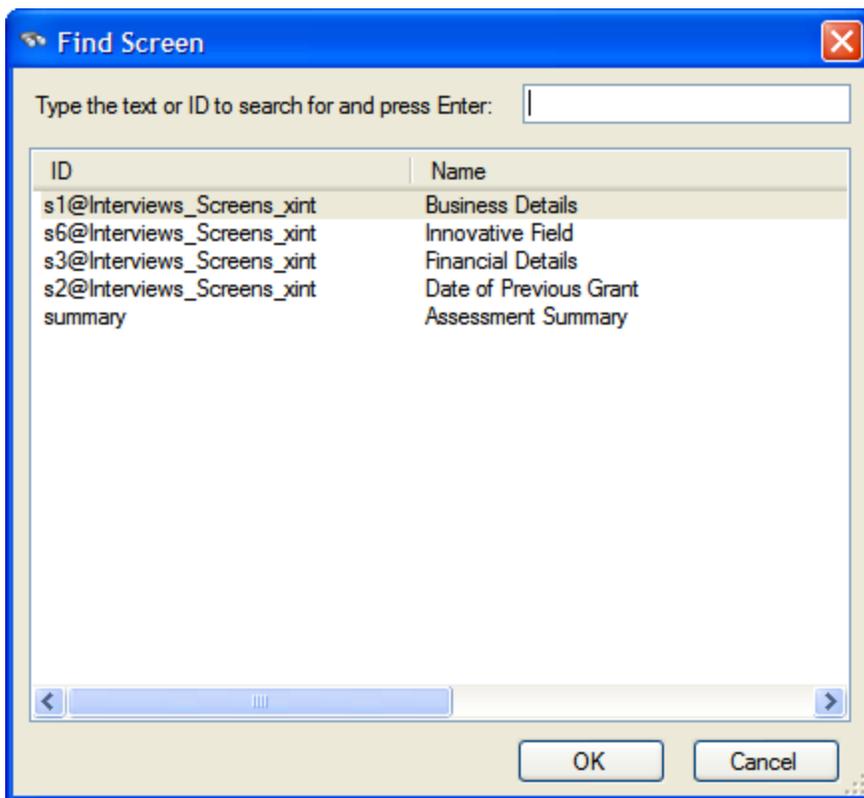
### Modify a question screen

To open the edit dialog for the screen again, double-click on the screen name in your screens file, or right-click and select **Open** from the pop-up menu. Make the necessary changes and then click **OK**.

### Find a question screen

To find a question screen:

1. In Oracle Policy Modeling, select **Edit | Find Screen...**  
This will open the **Find Screen** dialog box.



2. Enter the text or screen ID you want to search for in the text field provided. Only those screens that match the search criteria will be displayed in the list below.
3. Click **OK** to locate the screen in the screens file and open the screen for editing.

### Delete a question screen

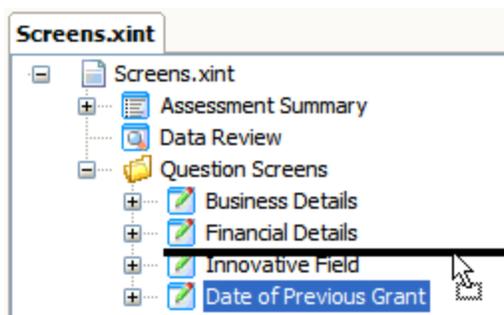
To delete a question screen, select the screen name in your screens file and press **Delete**, or right-click and select **Delete** from the pop-up menu.

## Organize question screens within a folder

By ordering the question screens in your Question Screens folder as you would like them to appear in an interview, you can simply use this folder to define your screen order. (See [Use the order of screens in the Question Screens folder to define the interview screen order](#) for more information.)

To change the order of question screens in a folder:

1. Open your question screens folder in your screens file.
2. Select the screen that you want to move and drag it to its new location.



3. Repeat for any additional screens that you want to reorder.
4. Click **OK**.

## Collect information about entity instances

An entity instance collection screen in Web Determinations is used to collect both a relationship and data about the related entity instances.

### What do you want to do?

Define a screen for collecting entity instances

Collect attributes for the entity

Create entity question screens

Use substitution on entity screens to identify the entity instance

Associate an entity instance with another set of entity instances via a reference relationship

### Define a screen for collecting entity instances

To set up a screen to collect entity instances:

1. Right-click the **Question Screens** folder in your screens file and select **New Question Screen**.
2. Enter an appropriate name for the screen in the **Screen Title** text box. TIP: The screen name should include the relationship text.
3. Select the **Collect Entity Instances on this screen** check box.

**Collect the children - Question Screen**

Screen Title:  Public Name:

Screen Entity:    Determine Automatically

Screen Attribute:

Collect Entity Instances on this screen

The Entity Control window will be displayed:

Collect Entity Instances on this screen

.....  <unknown entity>

**Entity Control**

Entity:

Behavior:

Add Instance Text:

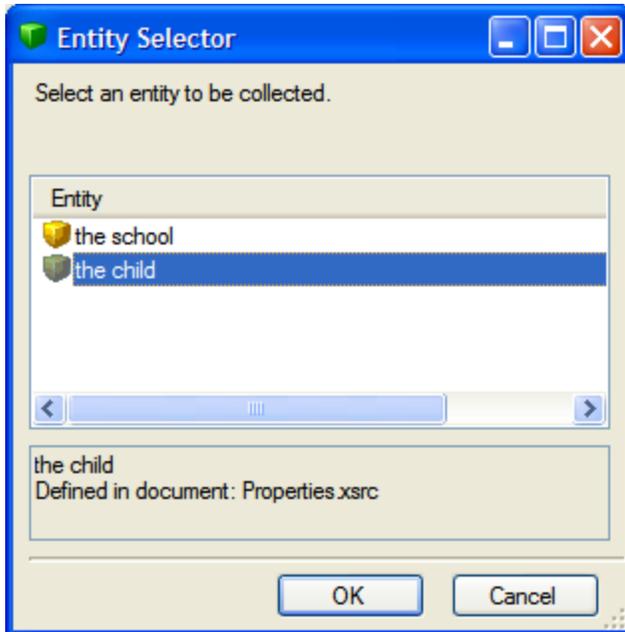
Remove Instance Text:

Base Value:  Entity instances will be named '#1', '#2', etc

Add default blank instance

Display style:

4. Click the browse button next to the **Entity** field to open the **Entity Selector** dialog. Select the appropriate entity.



5. Click **OK**. The entity name will now appear in the **Entity** text box in the Entity Control window. The Screen Entity is also shown, based on the containment relationship defined for the entity you have selected. TIP: The entity control is displayed as a cube in the screen control list on the left hand side of the question screen dialog. You can return to the entity control window at any point by clicking on this icon.
6. Specify the **Add Instance Text** and **Remove Instance Text**, if you wish to customize the text for the Add and Remove Instance buttons on the entity collect screen. If you leave these fields blank, the default text used for these buttons is "Add New Instance" and "Remove Instance(s)".
7. Specify the **Base Value** for the names of entity instances (eg "instance #", "child #", "pet #"). This is the base name used to generate the entity instance ID which is set behind the scenes. There are two circumstances in which you might see it on screens: either on automatic data review screens, or in a decision report that lists the entities associated with a relationship node where you have not [specified an identifying attribute](#) for the entity in question.
8. Select the **Add default blank instance** checkbox if you want a blank instance of the entity instance created by default when the screen is displayed.
9. Select the **Display style** from the drop-down list. This will determine how entity instances are displayed on the screen. The options are: **Portrait**, **Landscape**, **Tabular** and **Custom**. (If you select the Custom display style you will also need to specify the **Custom style**.)
10. Click **OK**.

### Collect attributes for the entity

On your entity collect screen, you should also collect some basic information about each entity instance in order to identify the entity instances at later points in the interview; for example, the [identifying attribute](#). To collect an attribute for the entity, do the following:

1. Double-click on the entity collect screen in your screens file to open it for editing.
2. Click on the **New Attribute Input** button.
3. In the **New Input Control** window, select the attribute you want to place on the screen as a question for each instance of the entity.

**New Input Control**

Attribute: \_\_\_\_\_

Enter text or attribute ID to search for

Show only base level attributes     Show only uncollected attributes

ID	Text
child_dob	the child's date of birth
person_salary	the person's annual salary
person_nickname	the person's nickname
<b>123</b> school_num_students	the school's number of students
school_type	the school's type

NOTE: You can only add attributes which belong to the target entity as questions to the screen. Adding attributes which do not belong to this entity will cause problems at runtime and will be reported as compilation errors.

4. Click **Create**.
5. Select the attribute on the left hand side of the screen. The right side of the question screen edit dialog will be replaced with a set of edit controls for the question based on its data type. Here you can [customize questions](#) and [customize user input options](#).
6. Click **OK**.

### Create entity question screens

Further question screens can be created to collect additional information about each entity instance. Each question screen can only collect attributes which belong to a particular entity, so for example, you cannot collect a global attribute and an entity attribute on the same screen. (Adding attributes from multiple entities to the same screen will cause problems at runtime and will be reported as compilation errors.)

To create an entity question screen:

1. Follow the steps outlined in [Create a question screen](#).
2. [Add questions to the screen](#). NOTE: All attributes need to be for the same entity.
3. Select the **Screen Entity**. If the **Determine Automatically** checkbox is selected for the screen, the entity to which the questions relate is automatically determined based on the entity of the attributes on the screen.

**New Question Screen**

Screen Title:     Public Name:

Screen Entity:      Determine Automatically

Screen Attribute:        

                Collect Entity Instances on this screen

school\_type: What is the school's type?

**123** school\_num\_students: What is the school's number of students?

Attribut

Attribute: \_\_\_\_\_

(If the **Determine Automatically** checkbox is not selected for the screen, you will need to manually select the screen entity. To do this, use the browse button next to the **Screen Entity** field to open the **Entity Selector** and then select the entity for the screen.)

4. Click **OK**.

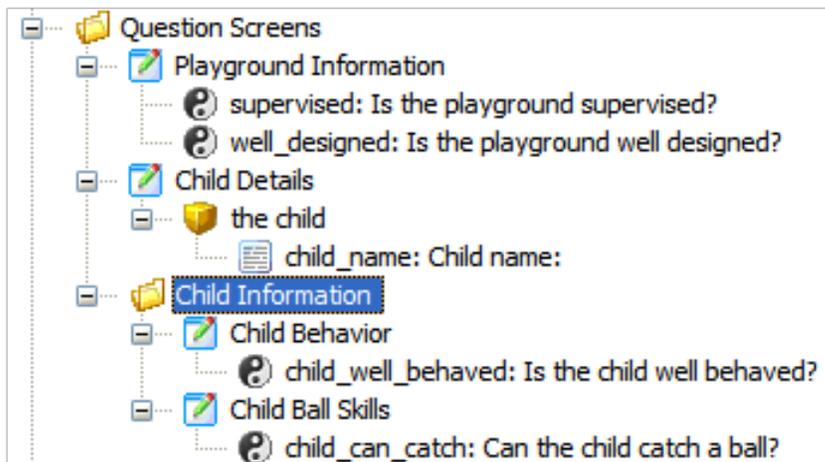
### Control the order of entity question screens

The default behavior of entity question screens is to show a screen for all instances of an entity before moving on to the next screen. For example, the following screen structure:



would ask "Is the child is well behaved?" for all children before asking the next question "Can the child catch a ball?" for all children.

To collect all information about an entity instance before moving on to the next instance of the entity, simply group the entity-level screens into a folder together. For example, the following screen structure:



would display all screens in the "Child Information" folder for one child (where relevant) before moving on to the next child.

The entity-level folder must only contain entity-level question screens for it to be identified as an entity-level folder and behave in this way.

Note that the entity collect screen "Child Details" is only shown once in the interview (to collect how many children there are) so should not be placed in the entity-level folder.

### Use substitution on entity screens to identify the entity instance

Where questions on a screen could be taken to relate to one of several instances of an entity, it is necessary to place the questions in context by clarifying the entity instance to which the questions relate. For example, a screen heading "The child's hobbies" will not provide enough information for the user to answer the questions if there is more than one child.

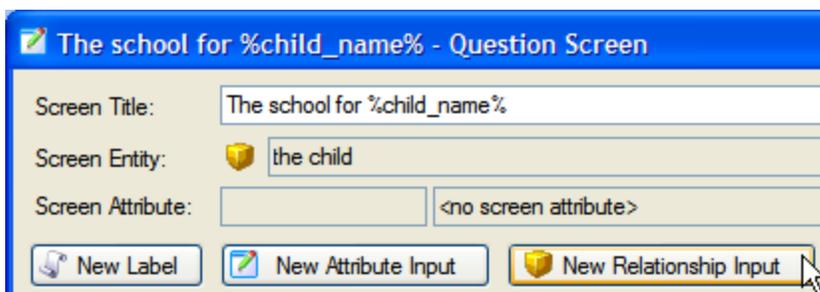
A variable should be substituted in all headings and controls where entity clarification is required. See [Substitute an attribute value into the text on screens](#) for how to do this.

TIP: The [concatenation function](#) can be used to create substitution variables such as "Bob (claimant, 32 years)", "Jane (daughter, 3 years)".

### Associate an entity instance with another set of entity instances via a reference relationship

On question screens, a relationship input control is used to set a relationship between entity instances that already exist in the interview. For example, if you wanted to collect a list of children (ie 'the child' entity instances) and collect a list of schools (ie 'the school' entity instances), then you would also want to find out which school (of the previously entered schools) each child goes to (ie to set the relationship 'the child's school'). To do this you would use a relationship input on a question screen as per the steps below.

1. After [defining your entity collect screens](#) for the two entities in the relationship, [create a new question screen](#). TIP: The screen title should include substitution for the name of the source entity (eg "The school for %child\_name%") so that it is clear to which entity instance the screen is referring.
2. Click on the **New Relationship Input** button.



3. In the **Relationship Input** window, click the browse button next to the **Relationship** field to open the **Relationship Selector** where you can pick the relationship that you want to set for the entities (eg 'the child's school').
4. Click the browse button next to the **Display Attribute** field to open the **Attribute Selector** where you can pick the attribute that is used to identify the target entity instances (eg 'the school'). This will be the attribute that is shown for each of the target entity instances, from which the user will select to set up the relationship. Note that you can control the list of target entity instances that is displayed - see [Filter the list of available target entities](#) below for more information.
5. Enter a **Caption**. This is the text that will appear next to the display attribute (eg "Pick the school that %child\_name% attends:"). TIP: You can use substitution of the name of the target entity in the caption to further clarify who you are setting the relationship for.

6. Click **OK**.

### Filter the list of available target entities

You can filter the list of entity instances that are able to be selected as a target of the relationship being collected. There are three options available to do this in the **List Options Filter** section for a relationship input control:

1. **Exclude source instance:** This option is useful where the source and target entities of the relationship you are collecting are the same. Selecting this option means that the source entity instance will not show up in the list of possible targets. (This option is selected by default.)  
For example, if you had an entity "the person" and a relationship "the person's spouse" where both the source and the target of the entity were the same. Since a person can not be their own spouse, selecting this option would remove the source instance from the list of target entity instances.
2. **Filter by Attribute:** This option lets you specify a boolean attribute in the relationship's target entity that (i) if true, will make that entity instance appear in the list of targets, and (ii) if false, will hide that entity instance in the list of targets. The **Default State** option can be used to specify what happens if the attribute is unknown or uncertain.
3. **Filter by Relationship:** This option lets you specify a relationship whose known targets are to be used as the set of possible targets for the relationship you are collecting. The source and target entities of the filter relationship must be the same as the relationship you are collecting. The **Default State** option is useful where the filter relationship is inferred and lets you control whether a target instance should be displayed in the list if it is unknown or uncertain whether it is a member of the relationship.

Note that the last two options are mutually exclusive.

## Customize interview user input options

A number of options are provided to customize user input options in a Web Determinations interview.

### What do you want to do?

Specify the type of input

Specify individual date and time edits

Specify the values for a restrictive input control

Source list contents from an external file

Specify a dynamic default for an input

Specify a default value for an input

Make an input mandatory

### Specify the type of input

For every attribute collected on a screen, you have several choices when defining your screen as to how the user can input their answer:

- Default (creates radio buttons for boolean attributes and a text box for all other attribute types)
- Checkbox (only selectable for booleans)
- Drop Down List
- List
- Radio Buttons (not selectable for booleans, use the Default option instead)

To specify the input type:

1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. Select the **Input type** from the drop-down list.
4. Click **OK**.

NOTE: For booleans collected with a checkbox, checked means true and unchecked means false. As a consequence, any boolean that is collected via a checkbox will always get a value as soon as the screen is submitted. It also means that the concept of **mandatory** is meaningless, so all boolean checkbox inputs will always be rendered without the mandatory flag.

### Specify individual date and time edits

For inputs for date, time and date/time variables, you can choose whether to display a single text entry edit, or individual edits for the components of the attribute (date/time/year/month/day/hour/minute/second components, depending on the attribute).

1. In your screens file, double click to open the screen containing the date, time or date/time question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. The **Input type** must be set to **Default** to use individual edits.
4. Under the **Style** section, select the appropriate **Input Type** option. All controls have the **Single Edit** option, which produces a single text entry box for the user to type freely into. Additional options are:

- Date variables have an option for individual **Year, Month and Day Edits**.
  - Time variables have the option of either individual **Hour, Minute and Second Edits** or individual **Hour and Minute Edits**.
  - Date/time variables have options for individual **Date and Time Edits**, or individual **Year, Month, Day, Hour, Minute and Second Edits** or individual **Year, Month, Day, Hour and Minute Edits**.
5. Additionally, you may specify the minimum **Minute increment** and **Second increment** for time and date/time variables, in common increments. (Note that **Second increment** is disabled if the input type chosen does not include seconds.)

Input Type:

---

Style:

Input type:

Minute increment:

Second Increment:

6. Most of these individual edits are displayed as drop down lists, allowing users to select from valid options. Year/date/time edits are displayed as text entry fields.

## ORACLE Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: HealthyEating Locale: en-GB User ID: guest

### General Information

Your name: \*

Assessment date and time: \*

Submit

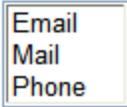
Note that if the **Single Edit** option is chosen for a date/time or time variable, the user can omit seconds and these will be set to 00 by default.

### Specify the values for a restrictive input control

When collecting data, particularly variable data, you may want to limit the set of possible answers a user may provide using a restrictive input control (ie list boxes, drop down lists or radio buttons). For example,

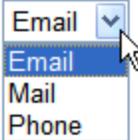
- List box:

What is your preferred contact method?

\* 

- Drop down list:

What is your preferred contact method?

\* 

- Radio buttons:

What is your preferred contact method?

\*  Email  
 Mail  
 Phone

NOTE: By default drop down lists are searchable and configurable. For more information, see [Change the appearance of a drop down list in Oracle Web Determinations](#).

TIP: Where a variable can only take on one of a fixed list of values, use a drop down list to collect the value from the user. Use this variable in interpretive rules to infer a number of separate boolean attributes, one for each possible value. Then use these boolean attributes in source rules. Do not use the variable value directly in source rules.

Once you have defined the restrictive input control (see [Specify the type of input](#) above), you need to specify the range of possible values for it. To do this:

1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. In the **Values** section, select the **Specify selection items** option.
4. Click the **New** button.
5. Change the values in the **Value** and **Display Text** text boxes from "new" to the values you want in the control. Ensure the values you enter are in [the correct format](#).

Values:

Specify list name

Specify selection items

Email	Value: <input type="text" value="Email"/>
Mail	Display Text: <input type="text" value="Email"/>
Phone	

NOTE: **Value** is the actual value passed to the Oracle Determinations Engine, whilst **Display Text** is what the user sees on the screen. Sometimes it is appropriate for these two values to be different. Display Text is a text value and will always be shown in the form you specify here; it will not be formatted according to the [rulebase regional settings](#).

TIP: You can use attribute substitution in the Display Text. For example:

Values:

Specify list name

Specify selection items

Yes, %claimant\_name%  No, %claimant\_name%

Value:

Display Text:

Selected by default

This would appear in Oracle Web Determinations as:

## Permission

Please confirm the following:

- \*  Yes, Phoebe gives permission for Mark to enquire about the claim  
 No, Phoebe does not give permission for Mark to enquire about the claim

For more information on using attribute substitution, see [Substitute an attribute value into the text on screens](#).

### Source list contents from an external file

For attributes of input type "List Box", "Drop Down List" or "Radio Buttons", it is possible to source attribute values from an external XML file.

To source list contents from an external file, do the following:

1. Create a new folder `\Development\include\lists\<session locale>`.
2. Create an XML file in this directory.  
**Note:** The name of the XML file should be related to the list it will be used for (for example, `JobIndustry.xml`).
3. In the XML file, add XML nodes for the various list options. For example, to create list options for a Job Industry list input, with Finance and IT as the list options:

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <option text="Finance" value="Finance" />
  <option text="IT" value="IT" />
</list>
```

4. Save the XML file using UTF-8 encoding.
5. In Oracle Policy Modeling, open your screens file and double click to open the screen containing the question.
6. Click on the question in the left hand pane to open it for editing in the right hand pane.
7. In the **Values** section, select the **Specify list name** option.
8. In the text field, enter the name of the XML file (eg JobIndustry).
9. Click **OK**.

If there are any problems with this file at runtime (for example, the file doesn't exist or uses the wrong XML structure), the static list options are used (if they exist for the list control).

NOTE: This xml list implementation does not currently support visibility or default values, but it does support the use of attribute substitution in **option text**.

### Specify a dynamic default for an input

It is possible for controls to have a dynamic default value, that is, a default value that is derived from the value of another attribute. During a Web Determinations interview, the default value for the control will be set to the value of the attribute that it is based on if it is known, or it will default to the user specified value if the dynamic default value is unknown.

To specify a dynamic default value for an attribute control:

1. In your screens file, double click to open the screen containing the question that you want to add the dynamic default to.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. In the **Dynamic Default Value** section, click the browse button next to the **Attribute** field.
4. In the **Attribute Selector**, select the attribute to base the default value on, then click **OK**.
5. Click **OK**.

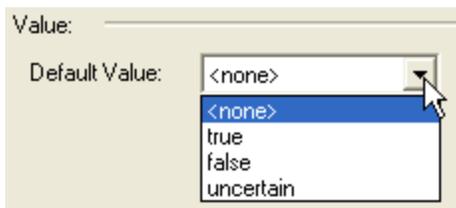
NOTE: A dynamic default attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect, it may be in the screen entity.

### Specify a default value for an input

Specifying default values for inputs speeds data collection and assessment processing. The default value is the value that will be selected/shown when the input control is displayed.

To specify a default value for an input:

1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. If the attribute has an input type of Default, then in the **Value** section select or specify a default value for the input.



Value: \_\_\_\_\_

Default Value:

- If the attribute has an input type of List Box, Drop Down List or Radio Buttons, then in the **Value** section select the value in the list that you want as the default, and then click the **Selected by default** checkbox.

Values: \_\_\_\_\_

Specify list name

Specify selection items

Email
Mail
* Phone

Value:

Display Text:

Selected by default

### Make an input mandatory

The rulebase will not necessarily need answers other than uncertain to any or all the questions on a screen. When designing screens you should therefore consider which answers should be mandatory and which ones should allow uncertainty. For more information, see [Decide whether to allow uncertainty in user answers](#).

To specify that an input control can allow uncertainty (ie allow a user to enter an uncertain value for the input control):

- In your screens file, double click to open the screen containing the question.
- Click on the question in the left hand pane to open it for editing in the right hand pane.
- In the **Mandatory** section, select the **Allow Uncertain** option from the drop-down list for **Default State**.

Mandatory: \_\_\_\_\_

Attribute:

Default State:  (if attribute is unknown or uncertain)

- Click **OK**.

This will mean that the input will always allow an uncertain option.

To specify that an input control can only allow uncertainty in some situations, then you need to create an attribute (and associated rules) to control in which circumstances the question will be mandatory. For example,

**the pregnancy question should be mandatory if**

the pregnancy question should be displayed

### the pregnancy question should not be displayed if

there is not a female in the household who is aged 10 to 60 years (inclusive)

To add this attribute to the input control:

1. In the **Mandatory** section, click the browse button next to the **Attribute** field.
2. In the **Attribute Selector**, select the attribute to base the value on, then click **OK**.  
TIP: The mandatory control attribute is expected to be boolean – a value of true means mandatory, a value of false means not mandatory (ie optional) so a wording for the control attribute such as "x should be mandatory" is recommended.  
NOTE: A mandatory attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect, it may be in the screen entity. The value of the mandatory attribute can not be **temporal**.
3. Select the **Default State** for the input control (ie **Mandatory** or **Allow Uncertain**). This is the state that the attribute will appear in if the mandatory control attribute is unknown or uncertain.
4. Click **OK**.

The rules relating to the use of **Allow Uncertain** are as follows:

1. For inputs where **Allow Uncertain** is not selected (that is, mandatory inputs), a blank string will be treated as not having supplied a value for this input and will therefore trigger an error.
2. For inputs where **Allow Uncertain** is selected, a blank string:
  - i. Will be treated as the value **Uncertain** for all inputs except text inputs.
  - ii. In the case of text inputs, a blank string will be treated as blank string and the text '*uncertain*' will be treated as setting the input to the string value **Uncertain**.
3. For boolean checkbox inputs, the concept of mandatory is meaningless, so these inputs will always be rendered without the mandatory flag.

See also:

- [Hide, display and disable an interview screen element](#)

### Decide whether to allow uncertainty in user answers

During an interview, it is possible for attributes to all have a value of "uncertain". "Uncertain" is considered a valid attribute value, allowing the assessment to continue without requiring the user to answer every question.

An assessment can continue until the goal attribute has a value, which may itself be uncertain, but which may be able to return true or false based on other information provided by the user. If a definite answer to the investigated goal can be proved by some means, then it may be that the user does not have to obtain the missing information (eg where one option in a OR rule is uncertain, but another returns true).

In contrast, if the user could answer only 'yes' or 'no' or select a fixed value to every question, the assessment would stop until the user was able to answer the question.

## Uncertainty and rulebase inheritance

In its most straightforward application, uncertainty reasoning operates to infer attributes to uncertain when the conditions that prove the attribute are uncertain. Consider the following hierarchy of rules:

**attribute 1 is true if**

attribute 2 is true

attribute 3 is true

attribute 4 is true

In an interview, if attribute 4 is set to uncertain, then that propagates through the rulebase in the following way:

attribute 3 is inferred to uncertain;

then attribute 2 is inferred to uncertain;

then attribute 1 is inferred to uncertain.

Higher-level attributes inherit uncertainty through inferencing. Unhandled, uncertainty can introduce unintended consequences in your applications. This idea applies equally to the 'unknown' operator.

See also:

- [Make an input mandatory](#)

## Hide, display and disable an interview screen element

There are frequent situations where you will want to hide and make visible input controls (questions and reference relationships), input control values (for restricted input controls) and labels based on user data. You may also wish to control summary screen elements in the same way, presenting different options on the screen at different stages of the interview.

You can do so by defining an attribute specifically to control the visibility of a screen element, and then writing rules to set the respective value of that attribute to control its state.

For example,

**the maternity leave question should be displayed if**

the applicant is female

In this example, we only want to show the maternity leave question if the applicant is female. If the applicant is male we do not want to ask about maternity leave.

NOTE: Visibility can only be based on an attribute that is known before a screen is displayed in Oracle Web Determinations. It cannot be based on the value of another attribute on the same screen. This is because the input values on a screen are not submitted by Oracle Web Determinations to the engine immediately after each value is entered - only once the screen is submitted. If you are using a custom user interface, however, then it is possible to submit data after each question and have the screen behave dynamically as values are entered.

There are also times when you may want to make an input control read-only thereby preventing users from altering data on that control.

## What do you want to do?

Control the visibility of questions, labels and relationships

Control the visibility of restricted input options

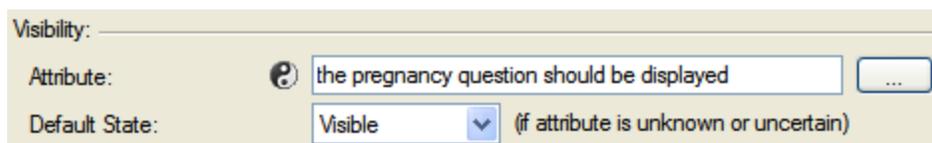
Control the visibility of summary screen elements

Make an input read-only

### Control the visibility of questions, labels and relationships

Once you have written rules to control the logic of when a question, label or relationship should be displayed, you then need to connect it to the related input control. To do this:

1. In your screens file, double click to open the screen containing the question, label or relationship input.
2. Click on the question, label or relationship in the left hand pane to open it for editing in the right hand pane.
3. In the **Visibility** section, click the browse button next to the **Attribute** field.
4. In the **Attribute Selector** dialog, select the visibility attribute from the list.
5. Click **OK**. Select the **Default State** for the input control (ie **Visible** or **Hidden**). This is the state that the attribute will appear in if the visibility control attribute is unknown or uncertain.



Visibility: \_\_\_\_\_

Attribute:  the pregnancy question should be displayed

Default State:  (if attribute is unknown or uncertain)

6. Click **OK**.

**TIP:** You can use the **Preview** button in the Screen Editor, assuming you have a debug session running containing the requisite data, to quickly check if the visibility attribute is working as expected.

#### NOTES:

- i. The Visibility control attribute is expected to be **boolean** - a value of true means visible, false means not visible (ie hidden). Generally, you should use an existing rulebase attribute as the visibility attribute if possible. If there is more complexity in the logic then you should create a rule proving the attribute (eg "question xxx should be displayed") which will become the visibility attribute.
- ii. A visibility attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect it may be in the screen entity.
- iii. The value of the visibility attribute can not be **temporal**.
- iv. If all the relevant questions that need a value in order to continue through the investigation are hidden on the screen, then the interview will not be able to progress. This will be reported as a ScreenLoopingException in your logs.

### Control the visibility of restricted input options

The method for controlling restricted input control values (ie members of a drop-down list, list box or radio button group) is identical to that for controlling individual screen controls.

1. Write a rule to control the visibility of the control value:  
**the Long Service Leave option should be displayed if**  
the employee has worked for the company for more than 10 years
2. In your screens file, double click to open the screen containing the restricted input control.
3. Click on the restricted input control in the left hand pane to open it for editing in the right hand pane.
4. In the **Values** section, select the control value in the list, then click the browse button next to the **Visibility Attribute** field.
5. In the **Attribute Selector** dialog, select the visibility attribute from the list, then click **OK**.

6. Select the **Default State** for the input control (ie **Visible** or **Hidden**).
7. Click **OK**.

Visibility of the value will depend on the truth value of the visibility attribute - visible if the value is true and hidden if the value is false.

NOTE: A visibility attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect it may be in the screen entity. The value of the visibility attribute can not be [temporal](#).

TIP: You can use the **Preview** button in the Screen Editor, assuming you have a debug session running containing the requisite data, to quickly check if the visibility attribute is working as expected.

### Control the visibility of summary screen elements

Summary screen elements may be controlled using the same method as screen controls and list elements, however three states are available: hidden, enabled (actionable) and disabled (displayed but not actionable). The visibility attribute for summary screen elements can be either a boolean or a text attribute. Use **boolean** logic to switch between two of these states, or a **text variable** to switch between the three states (setting the text variable to "hidden", "enabled" and/or "disabled" as required).

1. Write a rule to control the visibility of the summary screen element:  
**the end of interview items should be displayed on the summary screen if**  
the health interview is complete
2. In your screens file, double click to open the summary screen.

3. Click on the element in the left hand pane to open it for editing in the right hand pane.
4. In the **Visibility** section, click the browse button next to the **Attribute** field.
5. In the **Attribute Selector** dialog, select the visibility attribute from the list, then click **OK**.
6. Select the **Default State** for the input control (ie **Enabled**, **Disabled** or **Hidden**). This is the state that the element will appear in if the visibility control attribute is unknown or uncertain.

Visibility:

Attribute:

Default State:  (if attribute is unknown or uncertain)

7. Click **OK**.

If you need to switch between the three states (enabled, disabled and hidden), create a text variable to control the visibility and write a rule (eg a rule table) accordingly.

NOTE: A summary screen visibility attribute may be in the global entity or in the same entity as the summary screen element it is attached to. The value of the visibility attribute can not be [temporal](#).

TIP: You can use the **Preview** button in the Screen Editor, assuming you have a debug session running containing the requisite data, to quickly check if the visibility attribute is working as expected.

### Make an input read-only

By making an attribute read-only, you can prevent users from altering data on the input control. During a Web Determinations interview, the control will be rendered read-only if the attribute that it is based on is known, or it will default to the user specified value (editable or read only) if the attribute is unknown or uncertain.

To make an input read-only:

1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. In the **Read-Only** section, click the browse button next to the **Attribute** field.
4. In the **Attribute Selector** dialog, select the attribute to base the read-only status on, then click **OK**. TIP: The Read-only control attribute is expected to be boolean – a value of true means read-only, a value of false means not read-only (ie editable) so a wording for the control attribute such as "x should be read-only" is recommended.
5. Select the **Default State** for the control (ie whether the control should be **Editable** or **Read Only** if the attribute that it is based on is unknown or uncertain).
6. Click **OK**.

NOTE: A read-only control attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect it may be in the screen entity. The value of the read-only attribute can not be [temporal](#).

TIP: You can use the **Preview** button in the Screen Editor, assuming you have a debug session running containing the requisite data, to quickly check if the visibility attribute is working as expected.

## Tutorial: Hiding and displaying summary screen elements

To control what the user sees on the summary screen at different stages of an interview, you apply visibility attributes to the associated controls in the summary screen definition. Rules set the values of these attribute to control their state.

Common uses of visibility attributes on the summary screen are:

- to display a goal to investigate at the start of the interview, but then to hide it at the end, and
- to display additional labels at the end of the interview, but have them hidden initially

### Example

Let's say you have a procedural rule such as this:

**the interview is complete if**

the claimant is known and

it is known whether or not the claimant is eligible for low income allowance

You want to use this as your goal on the summary screen at the start of the interview, but at the end of the interview you instead want to display whether or not the claimant is eligible for the allowance.

To do this you would follow the process described below:

#### Step 1. Write the rules to control the display of the summary screen elements

In Microsoft Word, add the following rules:

**the goal for completing the interview should not be displayed on the summary screen if**

the interview is complete

**the eligibility goal should be displayed on the summary screen if**

the interview is complete

These rule conclusions are the attributes you will use as your visibility attributes.

#### Step 2. Add the procedural goal and visibility attribute to the summary screen

1. Open the summary screen in your screens file.
2. [Add a new goal](#) for the attribute "the interview is complete".
3. Change the **Unknown** caption to "Click here to determine eligibility for low income allowance".
4. [Add the visibility attribute](#) "the goal for completing the interview should be displayed on the summary screen" with the default state of **Enabled**.

#### Step 3. Add the eligibility goal and visibility attribute to the summary screen

1. On your summary screen, add a new goal for the attribute "the claimant is eligible for low income allowance".
2. Add the **Visibility attribute** "the eligibility goal should be displayed on the summary screen" with the default state of **Hidden**.

## Change the text of an interview question or sentence

Oracle Policy Modeling allows you to override the sentence and question forms generated by the inbuilt parser. You can also set up substitution which allows the text of a variable to be substituted with its actual value when it is used in another boolean attribute. Substitution can also be used to replace variables with pronouns rather than the text of the variable's value, and to substitute an attribute's value into headings and screen names on interview screens.

### What do you want to do?

[Customize sentence text](#)

[Customize question text](#)

[Substitute the actual value of a variable for its text](#)

[Substitute a gender pronoun for a text variable](#)

[Set up substitution](#)

[Collect the gender of a person](#)

[Substitute an attribute value into the text on screens](#)

[Display interview questions in second person form](#)

### Customize sentence text

Oracle Policy Modeling provides an alternate mechanism for sentence generation. This can be used for "canned text" (where a fixed statement, question, negation and uncertain form is used for each attribute). Overriding the text in this way should only be used:

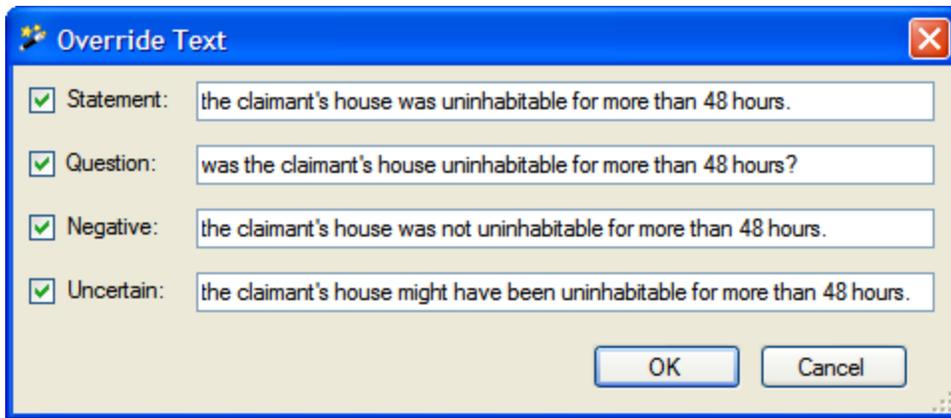
- a. when you need to simplify a complex sentence, or
- b. in a Rapid Language Support rulebase where the generic form defined in the RLS parser does not create the correct sentence text. For more information on using an RLS parser, and changing individual sentence forms in such a project, see the Help available in the Rapid Language Support Tool.

Text overriding should never:

- i. change the meaning of the sentence, or
- ii. be used instead of alternate parsing. If the sentence forms for an attribute are not what you expect, you should first consider an alternate parse for the attribute. See [Select an alternate parse](#) for more information.

To customize the sentence text for an attribute follow these steps:

1. In Oracle Policy Modeling, open the properties file for your project.
2. Double-click the attribute whose sentence text you want to change. This will open the **Attribute Editor**.
3. Select the **Override** button. This will open the **Override Text** dialog.
4. Select the check box next to the sentence form you want to change (statement, question, negative, uncertain). This will activate the text field for that form and you can then change the sentence text as required.



5. Select **OK** in the Override Text dialog box and then **OK** in the Attribute Editor dialog box.

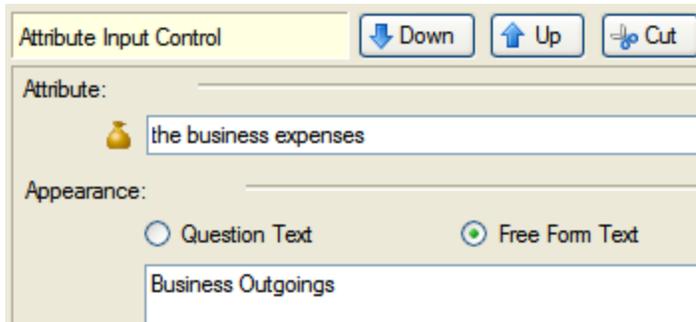
### Customize question text

Not all question forms generated by Oracle Policy Modeling's automatic parsing of attributes are suitable for the required purpose. In many cases, rather than asking a question in question form, you might want to abbreviate this to a label for the text field. For example, instead of asking "Who is the person?" in question form, it might be better to express this more simply as "Name:"

Overriding the default question text in this way is used when you need to abbreviate or shorten the text. The change in text should NEVER change the meaning of the sentence. If you want to simplify the sentence concept you should not use this mechanism – you should use the **Override** feature available in the properties file in Oracle Policy Modeling. For more information, see [Customize sentence text](#) above.

To override the default question text obtained from the automatic attribute parse:

1. In the **Attribute Input Control** pane, select the **Free Form Text** radio button.
2. Enter the override text which you would like displayed.



### Tips for question wording

General principles to bear in mind when editing question text are:

- use plain English rather than legal language
- use terminology the target audience (ie assessors) are familiar with
- do not assume the user is familiar with the source material unless instructed otherwise

- use substitution (see below) to make questions easier to understand
- keep questions short, cover additional detail in interview help text
- avoid superfluous phrases like "in respect of", "in the UK or elsewhere", etc.
- simplify questions so that users find it easier to determine what they are being asked (as long as you can do so without compromising the meaning of the underlying legislation/business rules/etc)

### Substitute the actual value of a variable for its text

It is possible to substitute the text of a variable with its actual value when it is used in another attribute in the rulebase.

For example, having the following attribute:

the claimant lives in the country of residence

and the variable:

the country of residence

allows substitution of the words "the country of residence" for the value of the variable.

So if the country of residence is "France" then

the claimant lives in the country of residence

becomes:

the claimant lives in France

This sort of substitution can occur at a more complex level, for example:

the claimant's sibling lives in the claimant's sibling's country with the claimant

can become:

Charlene lives in Morocco with Anne

where "the claimant's sibling", "the claimant's sibling's country" and "the claimant" are all substituting variables.

Using name substitution is particularly important when using entities. For example, asking:

"Is the child a full-time student?"

is not helpful if there are multiple children in the family. It is more useful to ask:

"Is Bart a full-time student?", "Is Lisa a full-time student?" etc.

How to [set up substitution](#) is explained below. For more information on how variable substitution operates, see [Text substitution principles](#).

### Substitute a gender pronoun for a text variable

Variables can be replaced with pronouns rather than the text of the variable's value. The pronoun substitution is based on the default gender specified for an attribute. This applies only to text variables which have substitution allowed (see [Set up substitution](#)

below).

For example, if we had the following attribute which uses the substituting text variable "the claimant":

the claimant lodged the claimant's form

we would not want this to become:

Tom lodged Tom's form

Rather we would want this to become:

Tom lodged his form

To set the default gender for a text variable:

1. In Oracle Policy Modeling, double click the properties file in the Project Explorer to open it for editing.
2. On the Attributes tab, double-click the variable to open it in the **Attribute Editor**.
3. Select an option from the **Default Gender** drop-down list (see below for an explanation of these choices), then click **OK**.

There are four default gender options to choose from:

Default Gender	When to use	Example
Impersonal (it)	This is typically used for things which don't have a gender like company names or inanimate objects.	The client company expended more than thirty percent of <b>the client company's</b> income in the relevant tax year. becomes Parker Incorporated expended more than thirty percent of <b>its</b> income in the relevant tax year.
Generic (he/she)	This is the most commonly used gender option for variables specifying people. The substitution pronoun will be automatically determined based on what gender the variable is given at runtime. NOTE: This setting should be used in conjunction with a gender attribute which is a means for collecting whether or not a person is male or female at runtime. See <a href="#">Collect the gender of a person</a> below.	The client expended more than thirty percent of <b>the client's income</b> in the relevant tax year. becomes He expended more than thirty percent of <b>his</b> income in the relevant tax year.
Male (he)	This is used where the gender is known to be masculine (eg "the man"). In this case the sex of the phrase need not be set at runtime - it will permanently be set to male.	The man has had surgery to <b>the man's</b> knee. becomes Bob has had surgery to

Default Gender	When to use	Example
Female (she)	This is used where the gender is known to be feminine (eg "the girl"). In this case the sex of the phrase need not be set at runtime - it will permanently be set to female.	<b>his</b> knee.  The client is in the second trimester of <b>the client's</b> pregnancy. becomes Gillian is in the second trimester of <b>her</b> pregnancy.

### Set up substitution

Values can only be substituted into other attributes if the variable is set to allow substitution. To specify this, follow the steps below.

1. Once you have created your variable in your rules document and compiled your rules, open the properties file for the project.
2. [Create a public name](#) for your attribute.
3. Select your variable in the attribute list, and double-click to open the **Attribute Editor**.
4. Check the **Allow Substitution** check box. If the variable is a text variable select the **Default Gender** from the drop-down box. This is to ensure that the correct pronoun is substituted for subsequent occurrences of the variable in the attribute. For more information, see [Substitute a gender pronoun for a text variable](#) below.
5. Click **OK** and save your document to apply these changes.

### TIPS:

- i. To prevent unwanted substitution occurring in your rules, when specifying substitution in a variable you should ensure that the same text is only used in other attributes in which substitution is appropriate.
- ii. Use the string concatenation function if you have collected a person's first name and last name separately but you want to combine them for the purpose of name substitution. See [Combine multiple text strings into a single text variable for more information](#).
- iii. When substitution is enabled, any existing translations will need to be manually updated. For more information, see [Update a translation file](#).

**NOTE:** Most attributes will not use the substitution option as it can lead to nonsensical attributes being produced during interviews.

For example, if you had an attribute:

the claimant's weekly rent doubled is more than one half of the claimant's weekly pay

and the variables "the claimant's weekly rent" and "the claimant's weekly pay" had substitution turned on, an attribute like the following would be generated:

\$200 doubled is more than one half of \$1,500

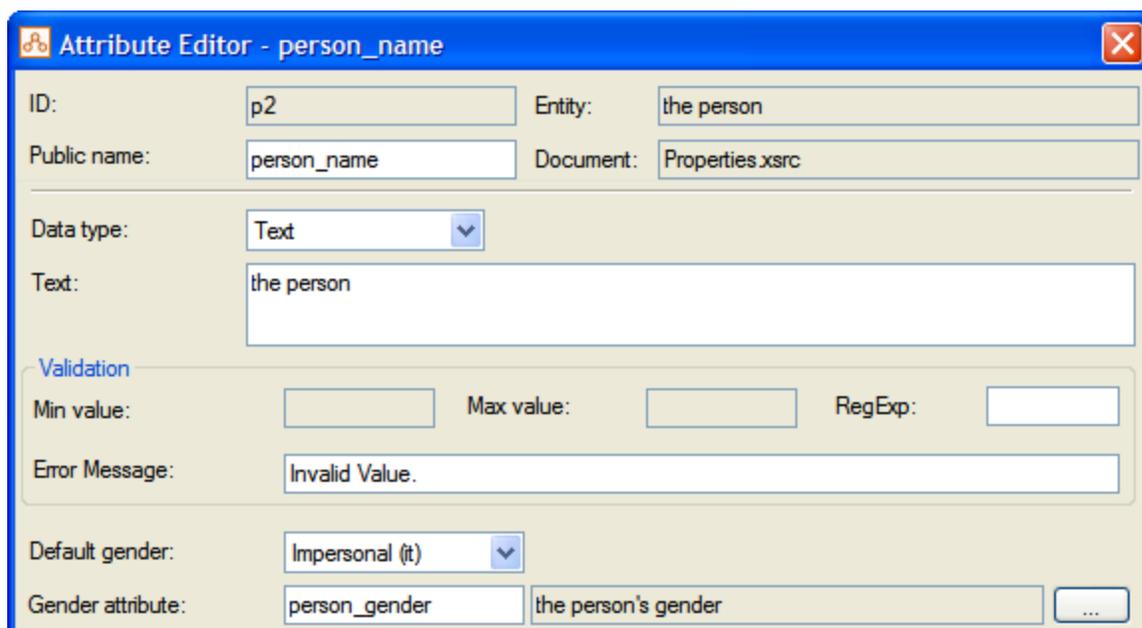
## Collect the gender of a person

A gender collect control is used to provide a means for collecting whether or not a person is male or female. This is done by adding a gender attribute as a control on a screen. Essentially, the gender attribute is a text type variable, and the control will set it to a value of either "male" or "female".

This is important for determining the correct [pronoun to substitute](#) when the default gender of a variable is generic (he/she), otherwise we will end up with attributes like "Tom lodged his/her form".

To set up a gender attribute:

1. In your properties file, create a new text variable to be used as the gender attribute (eg "the person's gender").
2. In the Attribute view for the properties file, right-click on the gender attribute and select **Copy Attribute ID**.
3. Also in the Attribute view, double-click on the attribute with which the gender attribute is to be associated (eg "the person") to open the **Attribute Editor** dialog box for that attribute.
4. In the **Gender Attribute** text box paste the attribute ID for your gender attribute.



The screenshot shows the 'Attribute Editor - person\_name' dialog box. It has a blue title bar with a close button. The dialog is divided into several sections:

- ID:** p2
- Entity:** the person
- Public name:** person\_name
- Document:** Properties.xsrc
- Data type:** Text (dropdown menu)
- Text:** the person
- Validation:** A section with 'Min value:', 'Max value:', and 'RegExp:' labels, each followed by an empty text box.
- Error Message:** Invalid Value.
- Default gender:** Impersonal (it) (dropdown menu)
- Gender attribute:** person\_gender (text box) | the person's gender (text box) | ... (button)

5. Click **OK** and then compile your document.

You also need to create a new input control for the gender attribute in your screen file. To do this:

1. In Oracle Policy Modeling, open your screen file and select the screen where you want to collect the gender.
2. Create a new input control on the screen and select the gender attribute. Click **Create**.
3. Set the control type to an appropriate restricted input control type (list box, drop-down list or radio button). You will notice that the values "male" and "female" will automatically be added to those control definitions. Click **OK**.

## Substitute an attribute value into the text on screens

It is possible to substitute an attribute value into free form question text, list display text, a heading or a screen name. For instance, we could substitute the person's name into the screen name, enabling the "Financial Details" screen to appear as "John's Financial Details". Substituting the person's name in this way can help personalize the interview process for the user.

To substitute an attribute value into the text on an interview screen:

1. In Oracle Policy Modeling, open your screen file and select the screen where you want the substitution to occur.
2. In the appropriate field (ie the **Screen Title** field for screen name, the **Display Text** field for list items, the **Text** field for the heading label, or the **Free Form Text** field for the question text) use the syntax "%<Attribute ID>%" or "%<Attribute ID>?%" where you would like the substitution to occur. (How these two different options operate is explained below.) NOTE: The attribute ID can be either the public name (eg %applicant\_name%) or the automatically assigned attribute ID (eg %p1@veterans\_doc%).
3. Click **OK**.

In the example above, you would get the Financial Details screen to appear as John's Financial Details by defining the screen name as either:

- %p1@veterans\_doc%'s Financial Details , or
- %applicant\_name%'s Financial Details

## Handling unknown and uncertain values

When an attribute is unknown or uncertain, the value substituted depends on the variation of the syntax used.

If the syntax "%<Attribute ID>%" is used, the formatted values for unknown and uncertain are used for the values unknown and uncertain respectively. For example, the caption "%applicant\_name%'s Financial Details" would be substituted as "unknown's Financial Details" and "uncertain's Financial Details" for unknown and uncertain respectively.

If the syntax "%<Attribute ID>?%" is used, the basic attribute text (eg the applicant's name) is substituted for both unknown and uncertain values. For example, "%applicant\_name?'s Financial Details" would be substituted as "The applicant's name's Financial Details" for both unknown and uncertain.

## Substituting gender pronoun attributes

If you wish to substitute a gender pronoun attribute onto screen text, you can use the syntax "%<Attribute ID>:his/her/its%", where <Attribute ID> is the main attribute to which the gender is applied (eg "the person"), rather than the gender attribute itself (eg "the person's gender").

For example, you may wish to define your screen title to read "Jane and her assets" (for an interview involving a person Jane who is female). You would do this with a screen title "%person% and %person:his/her/its% assets".

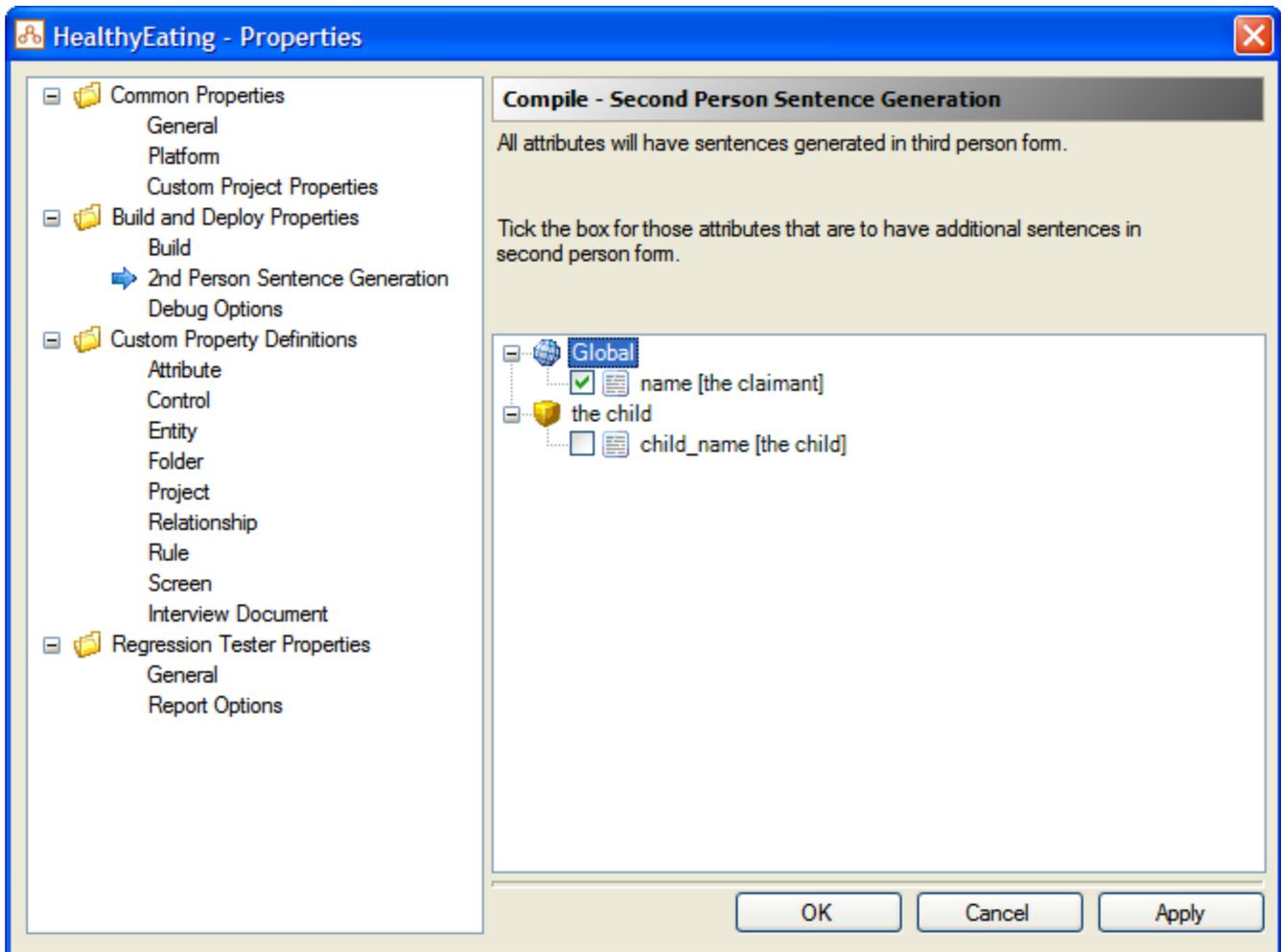
## Display interview questions in second person form

You can debug or run your rulebase with the questions being asked in second person form, rather than the more usual third person form. For example, "What is your taxable income?" instead of "What is Fred's taxable income?". This can be useful for self-service style interviews.

To have your rulebase questions asked in second-person form, you must first have a text variable [set up to substitute into other attributes](#), for example "the applicant".

1. **Open the Project Properties window from File | Project Properties.**
2. In the **Build and Deploy Properties**, select the **2nd Person Sentence Generation** option.

3. All text attributes in the rulebase which have substitution enabled are displayed. Tick the checkbox next to the attribute which you wish to designate as the second person attribute, and click **OK**.  
TIP: You would usually only tick one attribute for second person question forms in a rulebase.



4. The next time you debug or run your rulebase, any questions or statements using the attribute you ticked will be phrased in second person form.

## ORACLE Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#) |

Rulebase: HealthyEating L

### Further Information

Would you like to receive further information regarding healthy eating?

\*  Yes  No

NOTE: You may select entity attributes for second person sentence generation, however for this to operate in a meaningful way, you will require customizations to the application in which the rulebase runs. Entity instances by definition represent multiple items, so it is not meaningful for an interview to use second person form questions for every entity instance. However, application customizations may allow a particular entity instance to be designated as the single instance for which the second person questions should be shown, which would make this arrangement more meaningful to a user.

## Change the layout or appearance of interview screens

The default layout and appearance of interview screens can be changed by editing the screen definitions in Oracle Policy Modeling.

### What do you want to do?

[Change the appearance of text](#)

[Change the appearance of a control](#)

[Change the size of a text box](#)

[Add an image to a screen](#)

[Show/hide features used for debugging](#)

[Improve the appearance and layout of screens](#)

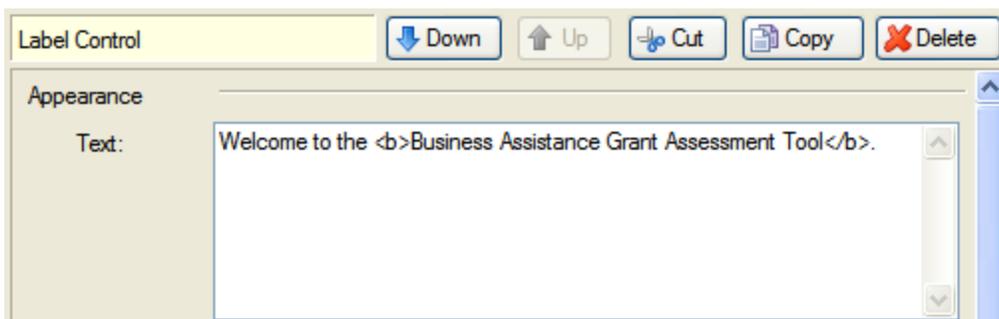
### Change the appearance of text

You can change the appearance of text (ie for a question, label or goal) on a question or summary screen using HTML tags defined in your screens file. For example, you may want to make some text bold, or add a hyperlink to a website on your screen.

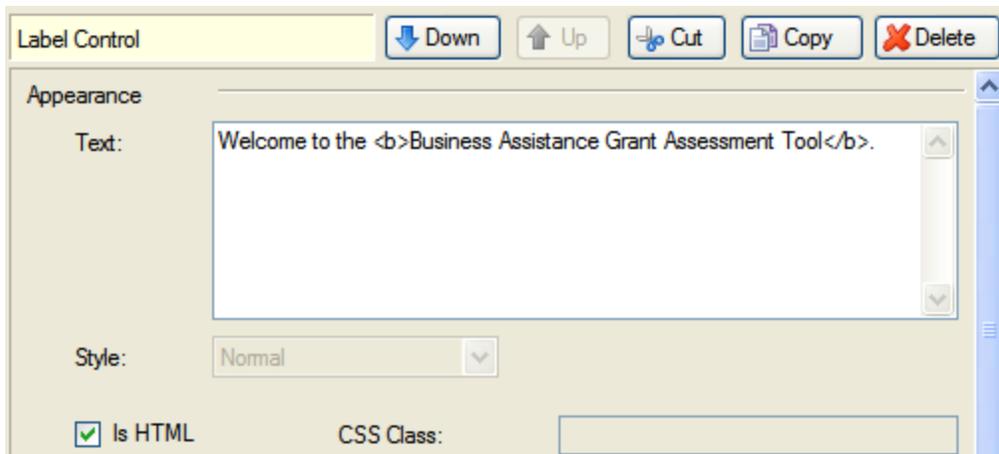
TIP: In order to use this feature you should have a basic working knowledge of HTML and web development. A good tutorial on HTML can be found at <http://www.w3schools.com/html/>.

To change the appearance of text using HTML tags:

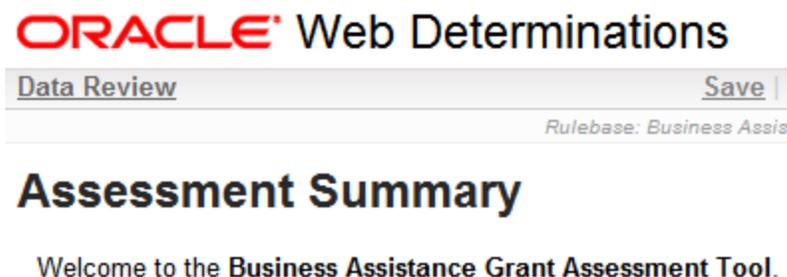
1. In Oracle Policy Modeling, open your screens file and select the relevant question or summary screen.
2. Select the label or control that you want to alter the appearance of.
3. Change the text of the label, question or goal to include the HTML tags.



4. Select the **Is HTML** checkbox.



5. Click **OK**. In Web Determinations this would be rendered as:



NOTE: When authoring a rulebase that is to be deployed in the Interview Portlet, relative links to static resources or screens can not be specified using IsHTML and `<a href="..." />`. This is because portlets require special URL rewriting, which is only available at runtime. Instead, the URL rewriter should be used. Refer to the [Oracle Policy Automation Developer's Guide](#) for more information.

### Allowable HTML tags

Entry of HTML tags in this manner has certain security implications and by default Oracle Policy Modeling and Oracle Web Determinations limit the set of HTML tags that can be entered. The default set of tags allowed are any of the following:

`b,i,del,s,div,p,span,pre,table,td,tr,ol,ul,li,blockquote,font,a,h1,h2,h3,h4,h5,h6,img,hr,br`

If extra HTML tags are required then they must be added to both Oracle Policy Modeling and Oracle Web Determinations as described below:

1. Oracle Policy Modeling - Go to **File | Project Properties | Common Properties | Platform**. In the **Web Determinations** section enter any additional HTML tags in the **HTML tags allowable in screen content** field. If a tag is used in a screens file that is not in this comma separated list, then a build error will occur.
2. Web Determinations - Open the **application.properties** file located in `\Release\web-determinations\WEB-INF\classes\configuration` for the rulebase project. In the **Rulebase Loading Properties** section, add any additional HTML tags to the **screens.html.tags.whitelist**.

`screens.html.tags.whitelist`

=b;i;del;s-

s;div;p;span;pre;table;td;tr;ol;ul;li;blockquote;font;a;h1;h2;h3;h4;h5;h6;img;hr;br;u

If a tag is used in a screens file that is not in this list, the rulebase will not load in Web Determinations. If necessary, this HTML validation can be turned off by setting the **screens.validate.html** setting in the file above to "false".

## Change the appearance of a control

The appearance of a control can be customized using Cascading Style Sheet (CSS) classes and styles. For example, you may want to make a control appear with a yellow background. TIP: In order to use this feature you should have a basic working knowledge of CSS. A good tutorial on CSS can be found at <http://www.w3schools.com/CSS/>.

To change the appearance of a control using CSS class and/or style:

1. In Oracle Policy Modeling, open your screens file and select the relevant question or summary screen.
2. Select the control that you want to alter the appearance of.
3. Enter a **CSS Class** and/or **CSS Style** in the fields provided. Style definitions must be entered in the format "property: value;" and all style definitions, including the last one, must finish with a semicolon. For example,

The screenshot shows a dialog box titled "Attribute Input Control". At the top, there are buttons for "Down", "Up", "Cut", "Copy", and "Delete". Below these, the "Attribute:" section contains a text field with "the date of application" and a small icon with the number "12". The "Appearance:" section has two radio buttons: "Question Text" (unselected) and "Free Form Text" (selected). Below the radio buttons is a large text area containing "Date of current application". At the bottom, there is a checkbox for "Is HTML" (unchecked), a "CSS Class:" field (empty), and a "CSS Style:" field containing "background-color:yellow;".

4. Click **OK**. In Web Determinations this would be rendered as:

## ORACLE Web Determinations

Summary | Data Review | Save | Save As | Load | Restart | Close

Rulebase: Business Assistance Grant Locale: en-US User ID: guest

### Business Details

*Please enter the following details about the business*

Date of current application

\*



NOTE: CSS Class definitions need support from a web developer in order to work. The web developer will need to modify the CSS and/or Velocity templates for that control type to give controls with that Class their distinct appearance. Bear in mind that the default Oracle Web Determinations (OWD) user interface has a number of accessibility features (for more information, see [Accessibility features in OWD](#)). You will need to perform your own checks to ensure that your modifications do not compromise the accessibility of your application. For more information on the development of custom controls, refer to the [Oracle Policy Automation Developer's Guide](#).

### Change the size of a text box

For text attributes that are being collected using a text box (ie by selecting an **Input Type** of **Default**), you can specify the number of lines for the text box. To do this:

1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. In the **Value** section, change the **No. of Lines** to the desired number of lines for the text box.

The screenshot shows the Oracle Policy Automation control editor interface. It is divided into several sections:

- Attribute:** A text field containing "the details of the innovative field worked in" and a button with three dots.
- Appearance:** Two radio buttons: "Question Text" (unselected) and "Free Form Text" (selected).
- Text Area:** A large text area containing the text "Please provide details:".
- Is HTML:** A checkbox that is currently unchecked.
- CSS Class:** An empty text field.
- CSS Style:** An empty text field.
- Input Type:** A dropdown menu currently showing "Default".
- Dynamic Default Value:** A section with an "Attribute:" label and an empty text field.
- Value:** A section with a "No. of Lines:" label and a spinner control set to the value "10".

### Add an image to a screen

To include an image on a screen you can use add it as a HTML link on a screen control as follows:

1. In Oracle Policy Modeling, open your screens file and select the relevant question or summary screen.
2. Double click to open the screen for editing.
3. Add a new control (ie an attribute, label or goal control - the type does not matter for this purpose) to the screen.
4. In the text field for the control, add the HTML reference to the image file in the following format:
  - i. `` (the format for adding images to the summary screen)  
For example, ``

- ii. `` (the format for adding images to question screens)  
For example, ``

5. Select the **Is HTML** checkbox.
6. Click **OK**.
7. Copy the image file to `\Release\web-determinations\WEB-INF\classes\images`.

### Show/hide features used for debugging

There are several features that may be useful to display in Web Determinations while debugging. These settings are made in the **appearance.properties** file that is located in `\Release\web-determinations\WEB-INF\classes\configuration` for the project. This file can be opened, modified and saved using Notepad.

TIP: Properties in this file can be overridden on a per-locale basis by creating an `appearance.<locale>.properties` file (eg `appearance.en-GB.properties`) file.

### Attribute question identifiers

By default, attribute ids are not shown in Web Determinations. To enable this feature, change the **show-attribute-question-identifiers** setting to **true**. Attribute ids (either the automatically generated id or the public name) will then be displayed before the question text:

## Business Details

*Please enter the following details about the business*

[app\_date] Date of current application

[employees] Number of employees

[previous\_grant] Has the business received the Business Assistance Grant previously?

### Status bar

By default, a status bar is shown in the top right area of Web Determinations that shows the name of the rulebase, the locale, the user id, and the case id (for saved cases) for the current session:



- To hide this status bar, change the **show-status-bar** setting to false.
- To hide the name of the rulebase in the status bar, change the **show-rulebase** setting to false.
- To hide the name of the locale in the status bar, change the **show-locale** setting to false.
- To hide the user id in the status bar, change the **show-user-id** setting to false.
- To hide the case id in the status bar, change the **show-case-id** setting to false.

### Improve the appearance and layout of screens

To improve the appearance and user experience of your screens, follow these tips:

### **Limit the number of questions per screen**

Placing a large number of attributes on one screen, particularly where the user is required to scroll down the screen before continuing, can make system confusing and difficult to use. Generally, each screen should contain a maximum of 4 questions. However, more questions per screen are acceptable where the questions are simple and take less than one line (eg Title, Forename, Surname, Date of Birth, Address), or if you are collecting a group of related questions which will all be defaulted (eg asking about a set of uncommon exceptions).

### **Use headings to convey meaning about the screen**

Web Determinations uses the screen name defined in Oracle Policy Modeling as the first heading on the screen. Additional headings are an important mechanism for conveying meaning about the screen. Variable substitution, eg %Claimant\_name\_age%, may be used in headings. Headings can be a useful tool to identify factors that will have a significant impact on the course of an investigation. The following are recommended uses for each of the heading levels:

- Heading 1 for major headings (eg "Assets of %Claimant\_name\_age% )
- Heading 2 for short labels (eg "If so...")
- Heading 3 for long labels (eg where the screen is effectively asking a head question followed by dot points)
- Heading 4 for sentences of explanatory text.

### **Group related questions together**

Generally, unrelated questions should not be placed on a screen together. It is much easier to focus on one issue at a time rather than on many different ones. Think about when a question will most naturally be asked in the screen flow. In particular, try to avoid splitting similar concepts.

### **Use visibility attributes on controls on the summary screen**

Visibility attributes should be used on the summary screen so that key outcomes, warnings and links to document generation are only displayed when an attribute proved by a system rule (such as "the investigation is complete") is true or partially true.

### **Use visibility attributes to hide mutually exclusive attributes**

Visibility attributes can be used on question screens to hide mutually exclusive attributes. For example, if you want one question to be asked if the person is single and another if they are a member of a couple, or if you want to hide a question where a person is not in a couple. For attributes which are mutually exclusive, you need to be sure that the visibility attribute will be known prior to hitting the screen.

### **Use visibility attributes to hide attributes proven by shortcut rules**

The general rule for attributes which might be proven with shortcut rules prior to hitting a screen is that the attribute should not be shown if it is already known. There will be some exceptions to this:

- where the attribute is on the screen largely to display an intermediate conclusion
- where it might be useful to the user to see the conclusion (eg to give context to other questions on the screen).

### **Default questions following an "if so..." or "if not..." label to uncertain**

You may want to group questions on screens where answers to some questions are only required if another question on screen has been answered in a particular way. For example you may only require information on the child's school if the child goes to school. Both "does the child go to school?" and "what is the name of the child's school?" can be collected on the same screen, separated by an "If so..." label.

Questions following an "if so..." or "if not..." label should always be defaulted to uncertain. A validation event rule should be used to force the user to answer the question when required.

### Put questions linked by shortcut rules on separate screens

Shortcut rules should only be used when you can prove a base level attribute before it is asked. If two attributes can be linked by a shortcut rule, then they should generally not be collected on the same screen. If you do have to collect them on the same screen and they are both mandatory the shortcut rule should be removed. You may need to add some validation rules (events) to ensure logical consistency.

See also:

- [Customize Oracle Web Determinations](#)

### Customize Oracle Web Determinations

The styling of Oracle Web Determinations can be customized to suit your needs. Some simple changes can be done by modifying the **messages.<locale>.properties** file and the **appearance.properties** file. The default location for these files in a rulebase project is: \Release\web-determinations\WEB-INF\classes\configuration.

For information on more advanced customizations of Web Determinations, see the [Oracle Policy Automation Developer's Guide](#).

Note: The default Oracle Web Determinations (OWD) user interface has a number of accessibility features (for more information, see [Accessibility features in OWD](#)). If you customize this user interface, you will need to perform your own checks to ensure that your modifications do not compromise the accessibility of your application.

### What do you want to do?

[Change the Oracle Web Determinations banner](#)

[Configure the Oracle Web Determinations labels](#)

[Change the appearance of a drop down list in Oracle Web Determinations](#)

[Change the locale list in Oracle Web Determinations](#)

[Change the Oracle Web Determinations banner](#)

The Web Determinations banner is made up of an Oracle graphic (oralogo\_small.gif) and the text "Web Determinations":

The image shows the Oracle Web Determinations banner. It features the Oracle logo in red, followed by the text "Web Determinations" in a dark blue, sans-serif font.

This banner can be modified to use a different logo and name.

### Change the image in the Oracle Web Determinations banner

To replace the Oracle graphic:

1. Save the new graphic as oralogo\_small.gif in \Release\web-determinations\WEB-INF\classes\images.
2. [Build and run the rulebase](#) to view the new image in the Web Determinations banner.

TIP: To hide the banner image completely, modify the 'show header image' setting in the appearance.properties file as follows:

```
show-header-image = false
```

## Change the text in the Oracle Web Determinations banner

The "Web Determinations" banner text is defined in the 'application-name' setting in the messages.<locale>.properties file:

To change the text in the banner:

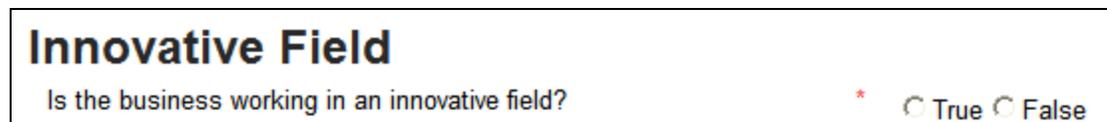
1. Open the messages.<locale>.properties file using Notepad.
2. Edit the 'application-name' configuration line, eg  
`application-name = Income Assistance`
3. Save the file.
4. [Build and run the rulebase](#) to view the new text in the Web Determinations banner.

TIP: To remove the banner text completely (eg if your image/logo contains the necessary text), modify the 'show application name' setting in the appearance.properties file as follows:

```
show-application-name = false
```

## Configure the Oracle Web Determinations labels

The label text of all links and buttons (eg Save As, Load, Restart, Close, Yes, No, Submit, Add Instance, [Why?]) in Oracle Web Determinations can be modified in the messages.<locale>.properties file. For example:



Here the out-of-the-box label text for the boolean answers "Yes" and "No", has been replaced with "True" and "False" respectively.

To change the text of a label:

1. Open the messages.<locale>.properties file using Notepad.
2. Edit the appropriate configuration line for the text, eg  
`boolean-true = True`  
`boolean-false = False`
3. Save the file.
4. [Build and run the rulebase](#) to view the new label text in Web Determinations.

## Change the appearance of a drop down list in Oracle Web Determinations

By default, drop down lists in Oracle Web Determinations are searchable. This means that the list of items is filtered based on the text that the user inputs.

Matches that take place at the start of the string will take priority over ones that occur somewhere else in the string and will be shown higher up in the list. Note that the text search is case insensitive.

## Your details

Name:  \*

Age:  \*

Location:  \* 

---

*Start typing here to see list filtered to only show text matches* 

- Nebraska
- Nevada
- New Hampshire
- New Jersey
- New Mexico
- New York
- North Carolina
- North Dakota
- Arizona
- Arkansas
- ...

To turn off searchable drop down lists in your project (making them render as normal HTML drop down lists):

1. Open the `appearance.properties` file using Notepad.
2. Modify the 'enable-searching-comboboxes' configuration line as follows:  
`enable-searching-comboboxes = false`
3. Save the file.
4. [Build and run the rulebase](#) to view the change in Web Determinations.

In searchable drop down lists you can configure the number of results displayed, the tooltip displayed for the trigger button, and the text displayed when the maximum number of results has been exceeded.

## Your details

Name: \* Steve

Age: \* 50

Location: \* Alabama

Alabama

Alaska

Arizona

Arkansas

California

Colorado

Connecticut

Delaware

District of Columbia

Florida

...

Show all items

"searching-combo-trigger-tooltip" setting

"max-search-results" setting

"searching-combo-more-results" setting

To change the maximum number of results to be displayed in the drop down list at any one time (the default is 10):

1. Open the `appearance.properties` file using Notepad.
2. Modify the 'max-search-results' configuration line as follows:  
`max-search-results =20`
3. Save the file.
4. [Build and run the rulebase](#) to view the change in Web Determinations.

Note that this number can have a substantial impact on performance - the lower the number is, the better the performance will be. Setting this number to 0 will cause all matching results to be displayed.

By default, the tooltip text that is displayed when the user hovers over the drop down trigger button is "Show all items". To change the tooltip text:

1. Open the `messages.<locale>.properties` file using Notepad.
2. Edit the 'searching-combo-trigger-tooltip' configuration line, eg  
`searching-combo-trigger-tooltip =Show list`
3. Save the file.
4. [Build and run the rulebase](#) to view the change in Web Determinations.

You can change the text that is displayed at the bottom of the drop down list when the maximum number of search results has been exceeded (by default it is "..."). To change this text:

1. Open the messages.<locale>.properties file using Notepad.
2. Edit the 'searching-combo-more-results' configuration line, eg  
`searching-combo-more-results =Start typing text to see more options`
3. Save the file.
4. [Build and run the rulebase](#) to view the change in Web Determinations.

## Change the locale list in Oracle Web Determinations

The locale list that is displayed when running a translated rulebase in Web Determinations can be modified in the appearance.properties file as follows:

1. Open the appearance.properties file using Notepad.
2. In the locale list (under 'Locale listings for localizing your language selection') change the locale name, eg  
`locale-en-NZ =English (New Zealand)`  
could be changed to  
`locale-en-NZ =English (NZ)`
3. Save the file.
4. [Build and run the rulebase](#) to view the new locale text in Web Determinations.

See also:

- [Change the layout or appearance of interview screens](#)

## Define interview screen order

By default, the screen order in an interview is primarily driven by the [question search](#). There are several limitations, however, to solely using the question search to drive question or screen order. For instance, using the question search alone to drive question or screen order:

- You cannot revisit screens in an investigation without resorting to the data review screen.
- Backtracking through an investigation is unreliable.
- It is difficult to control screen ordering. In order to control the screen order, screen order rules are intermixed with declarative logic which obscures the intent and source of the rules.

It is therefore beneficial to specify an explicit screen order in Oracle Policy Modeling that you would like your interview to follow. Using a defined screen order, the interview will follow the specified order of the screens only until enough information is known to make a decision, thereby avoiding making the user visit unnecessary screens. (This is in contrast to the functioning of a [Screen Flow](#) in which the interview will follow exactly the specified flow to its completion even if sufficient information is already known to make a determination.)

## What do you want to do?

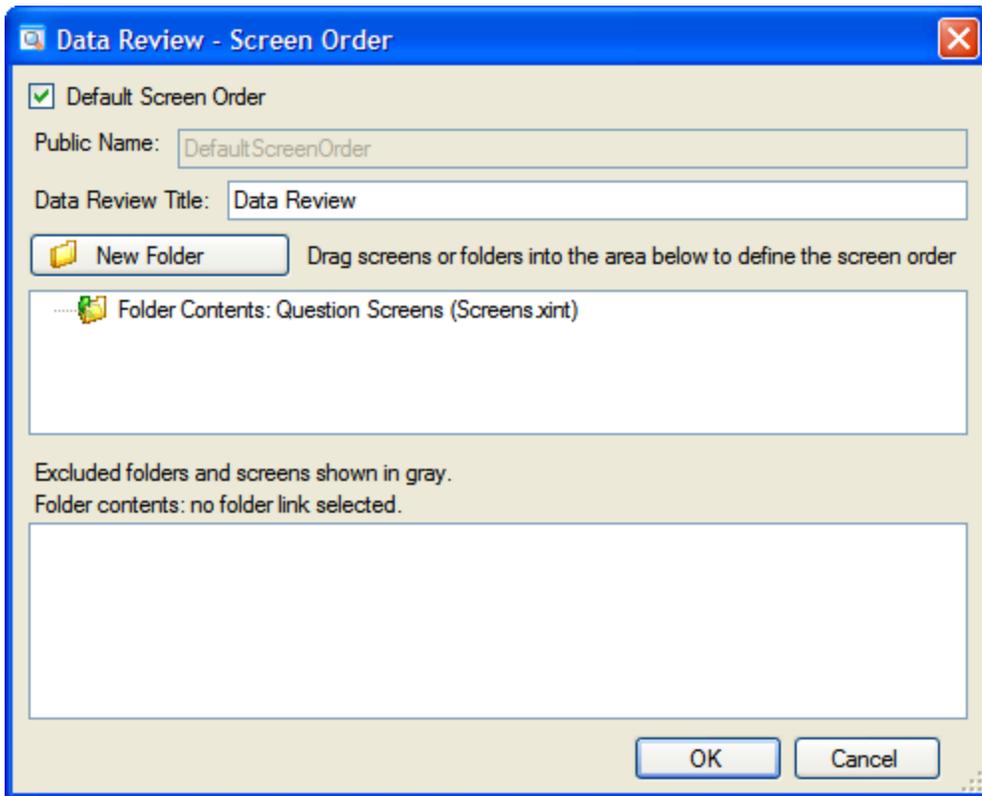
[Use the order of screens in the Question Screens folder to define the interview screen order](#)

[Create a new screen order](#)

[Edit a screen order](#)

[Use the order of screens in the Question Screens folder to define the interview screen order](#)

By default, the first screens file that is added to a project will contain a default screen order (labeled **Data Review** in the screens view). This screen order is automatically defined as being the order of the screens in the Question Screens folder.



This means that you just need to [order your screens in the Question Screens folder](#) as you would like them to appear in the interview, and this will automatically determine the screen order (and the order of the screens on the data review screen). That is, you do not need to manually add a new screen order to your screens file.

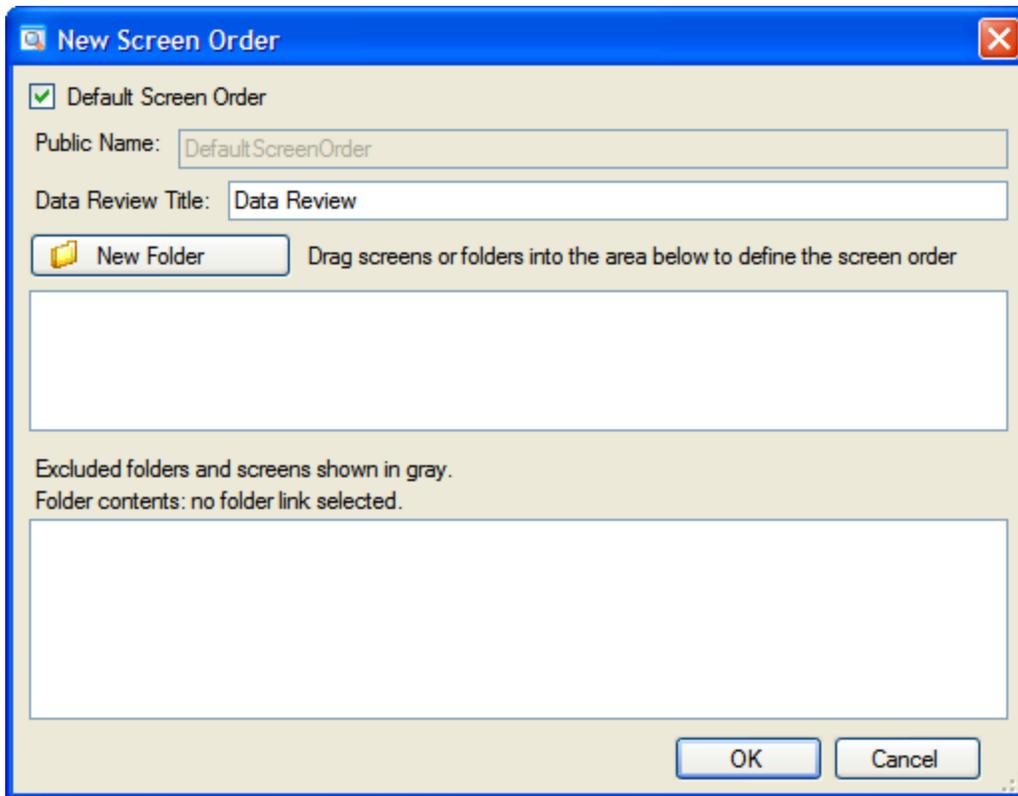
Note that this default behavior only applies to the first screens file added to a project. Any subsequent screens files added to the project will need to have a screen order added manually (see below).

### Create a new screen order

If you need to manually add a new screen order follow the steps below. (By default, the first screens file that is added to a project will already contain a default screen order - see above).

To create a new screen order:

1. Right-click the \*.xint filename, or another folder, in the screens view.
2. Select **New Screen Order** from the pop-up menu. The following dialog will appear:



3. Drag individual screens or folders from the screens file into the top pane to define the screen order.  
TIP: If you have your question screens in a separate folder in your screens file, you can order the screens as you would like them to appear in the interview in that folder. Then when creating your screen order here it is simply a matter of dragging that Question Screens folder into the top pane.
4. New folders can be added to the screen order by clicking on the **New Folder** button. (The folders in a screen order are used to group screens into the 'stages' that are displayed at the top of an interview to indicate progress through the investigation. If screens appear at the top level in the screen order (outside of any folder), they will also be used as stage names).
5. Click **OK**.

### Edit a screen order

To edit a screen order:

1. Double-click the screen order in the screens file to open it for editing.
2. Change the order of screens by dragging and dropping the screens into new locations. Alternatively, if your screen order is defined by the Question Screens folder, [reorganize the order of screens in that folder](#) in the screens view.
3. Click **OK**.

### Define interview screen flow

By default, the screen order in an interview is primarily driven by the [question search](#). There are times, however, when you may want the interview to follow a very precise screen flow (for example, if modeling a claim form). To enable this, you can define a

Screen Flow in Oracle Policy Modeling in which you draw the flow diagrams which represent the process or series of steps through your investigation. In this way you can have a screen flow which is completely independent of the rulebase.

## What do you want to do?

[Create a new screen flow](#)

[Edit a screen flow](#)

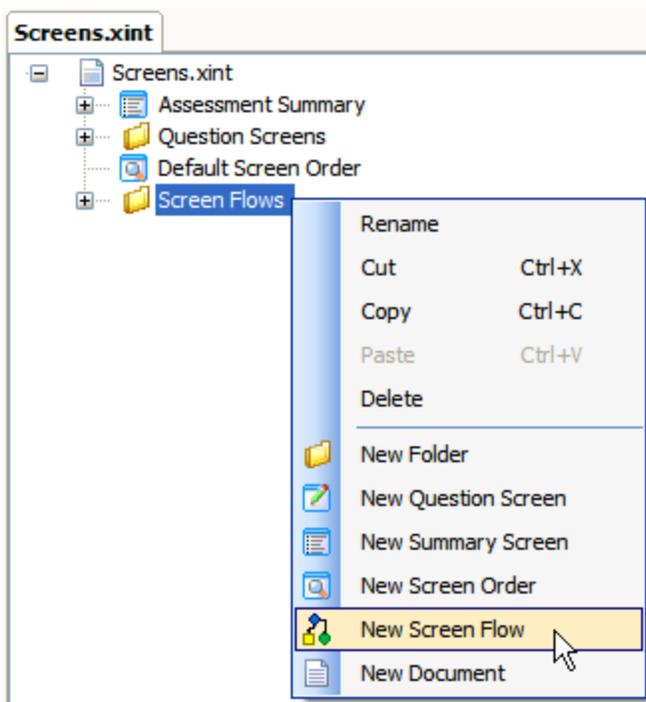
Create a new screen flow

Before you create a screen flow, first create a folder in which to store the flow:

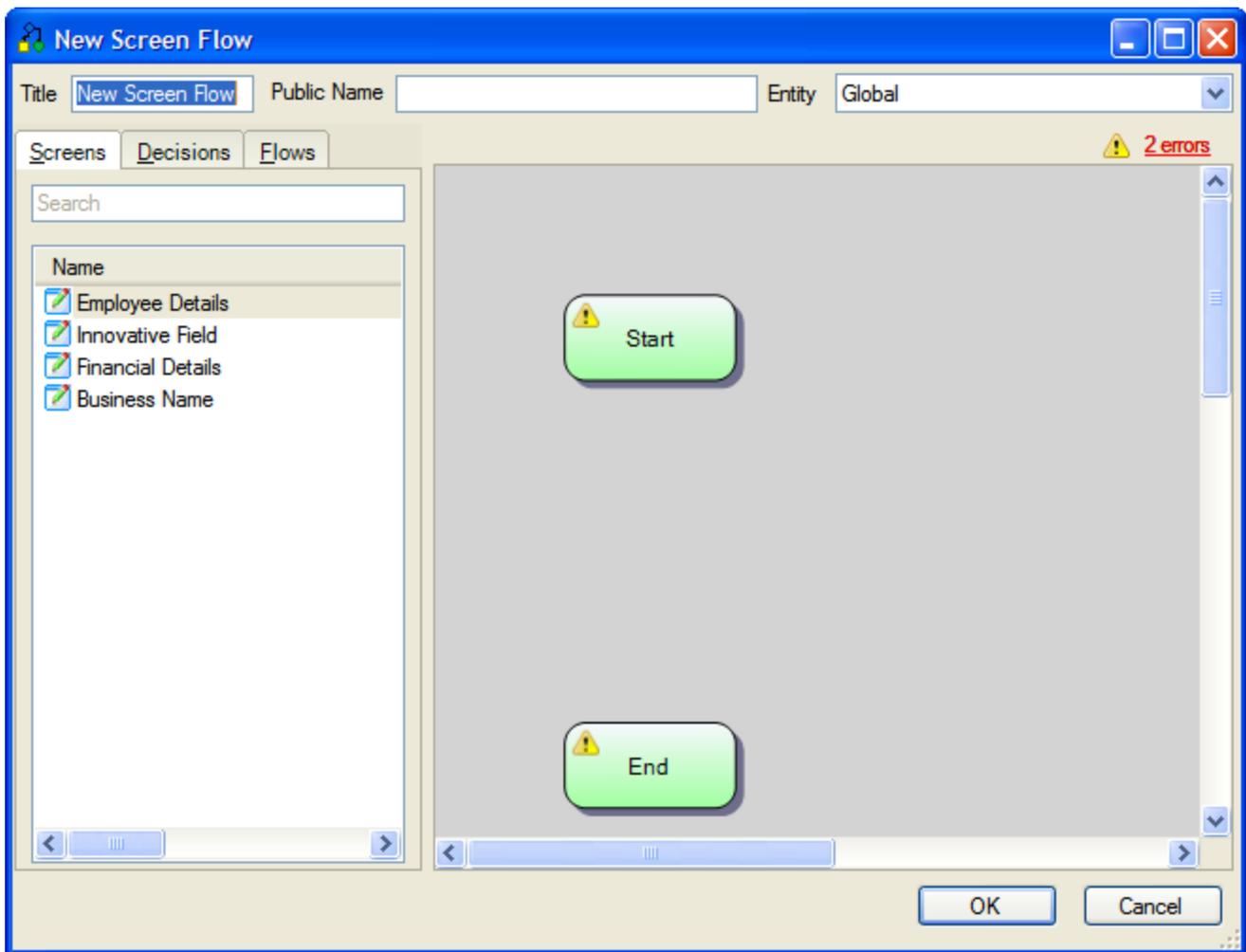
1. In your screens file, right-click the \*.xint filename in the screens view.
2. Select **New Folder** from the pop-up menu.
3. Enter an appropriate name for your folder (eg "Screen Flows").

To create a screen flow:

1. Right-click the screen flows folder in your screens file.
2. Select **New Screen Flow** from the pop-up menu.



The following dialog will appear:



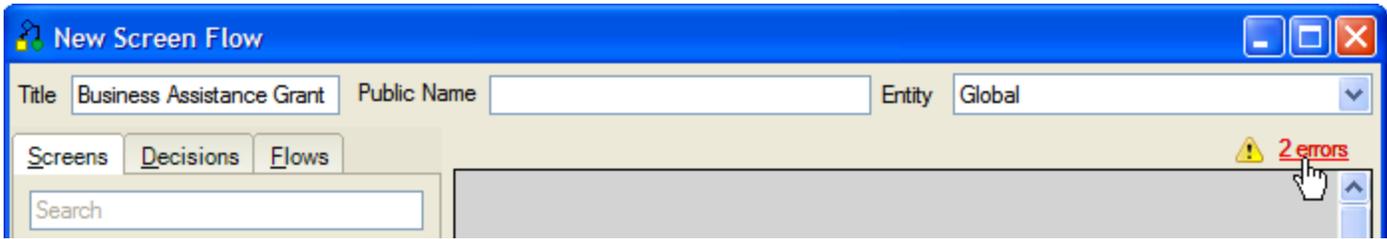
The left hand pane shows the various tabs that you use to select which shapes to add to the flow diagram in the right hand pane.

There are several steps involved in creating a typical flow as follows:

- Name the flow
- Add screens to the screen flow
- Add decision points to the screen flow
- Choose the entity that the screen flow operates within
- Add any subflows to the screen flow

A flow must start with a Start shape and end with an End shape.

Any errors that have been detected in the flow are shown by . To see a list of all errors, click on the link in the top right hand corner of the Screen Flow dialog.



After completing your flow, the flow needs to be added to the summary screen for the project so that it can be accessed in a Oracle Web Determinations investigation. For details on how to add the flow to the summary screen, see [Add a screen flow to the summary screen](#).

### Name the flow

The name of a flow is referenced by the summary screen and other flows. To change the flow name, enter a new name in the **Title** text field of the **New Screen Flow** dialog box.

NOTE: Flow names must be unique across all flows to allow references to [subflows](#).

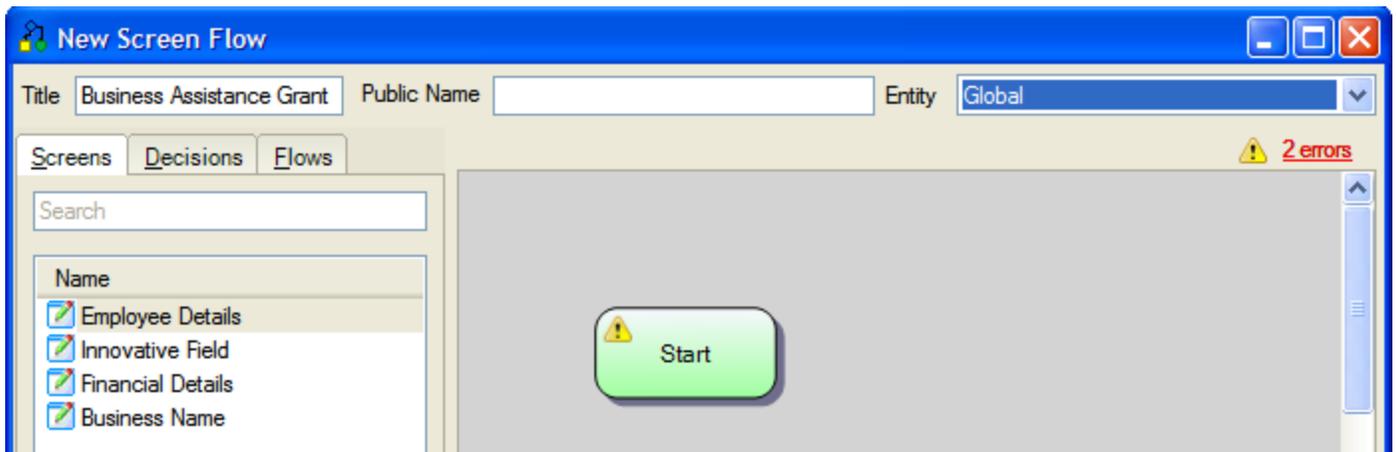
Add a public name for the flow if appropriate, in the **Public Name** field.

### Add screens to the screen flow

Interview screens can only be created using the standard interface within Oracle Policy Modeling. In your screen flow you can reference these screens.

To add an existing screen from Oracle Policy Modeling to your flow, follow these steps:

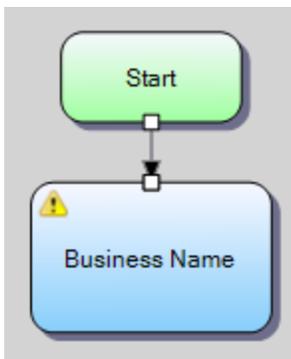
1. Click the **Screens** tab in the **Screen Flow** dialog box. This will show all the screens for the selected entity (the entity is shown in the top right hand of the Screen Flow dialog box).



2. Select the screen you want to add to your flow. You can search or filter the list by typing in the Search text box.
3. Drag the screen to the diagram and place it just under the shape that it should flow from.



4. Drag between shapes to join them.



### **Add decision points to the screen flow**

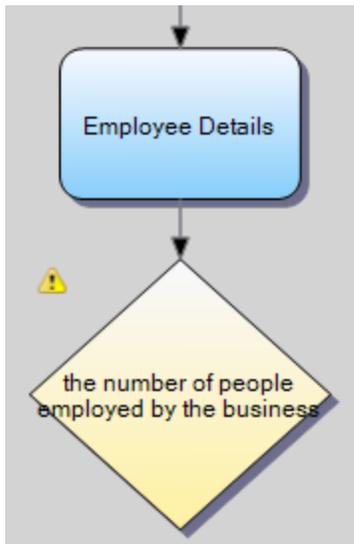
You can add conditions in your screen flow that control whether or not a path in the flow is taken. A condition is an attribute and the attribute value controls which path is taken. Each condition is called a decision point and is represented in the flow with a decision shape.

To add a decision point to your flow:

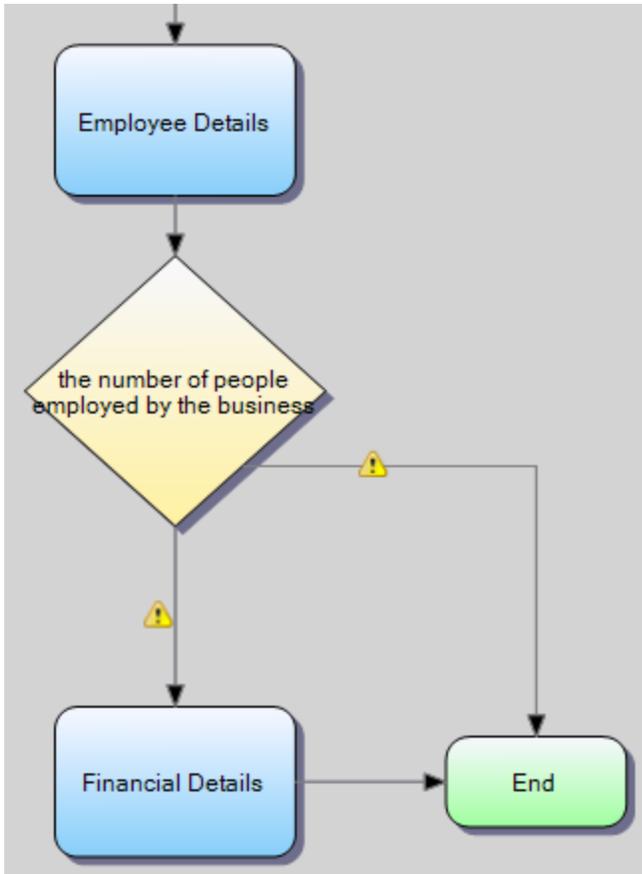
1. Click the **Decisions** tab in the **Screen Flow** dialog box.

ID	Text
business_name	the business
expenses	the business expenses
b6@Rules_Rules_doc	the business expenses are greater than the business revenue
b2@Rules_Rules_doc	the business has between 4 and 12 employees
eligible	the business is eligible for a Business Assistance Grant
innovative_intended	the business is intending to work in an innovative field
b5@Rules_Rules_doc	the business is struggling financially
innovative	the business is working in an innovative field
revenue	the business revenue
grant_received_date	the date the business previously received the grant
b1@Rules_ProceduralRules_doc	the interview is complete
123employees	the number of people employed by the business

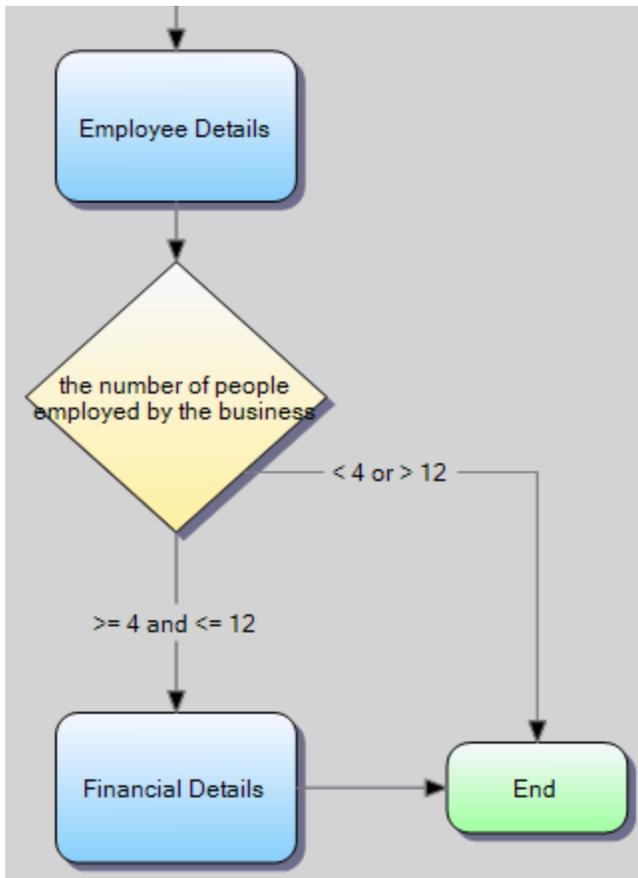
2. Select the attribute you want to add to your flow. You can search or filter the list by entering part of the attribute text or ID in the Search text box.
3. Drag the attribute to just under the shape that it should flow from in the diagram.



4. Drag from the decision shape to its related shapes to connect them.



5. For each connection that flows from a decision shape, double-click the connection and type a value condition. Then press **Enter**.



A condition can just be a straight value (eg true|false|"cat"). For number, date, date and time and time of day attributes, a condition can also be preceded by a comparison. For more information on what value conditions can be applied to different attribute types, see [Value conditions for screen flow connections](#).

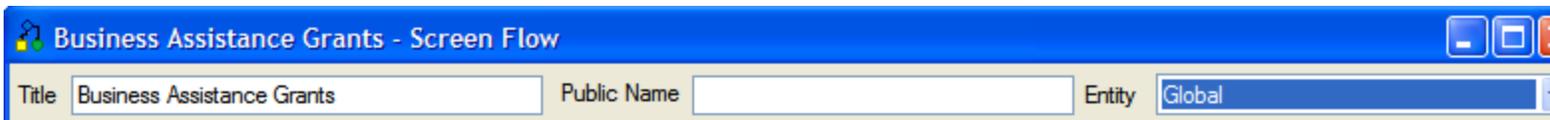
If no conditions are provided for a decision point, the attribute will be investigated and then the flow will continue on from that point.

#### NOTES:

- When a screen flow is executed, if it reaches a decision shape where the attribute is unknown, it will trigger an investigation of that goal using the default [screen order](#). If an attribute is required that doesn't appear in the screen order, then it will use the question search to investigate the attribute. It is best practice to have the screen flow for a rulebase fully specified, therefore avoiding the need for the question search to drive the screens.
- You can also have a decision shape with the condition "unknown" and it will be taken when the value is unknown instead of investigating the goal. If you don't have any such condition, the goal is investigated as described above.
- If the text of a condition is blank or "else" or "otherwise", then this is the 'catch all' connection that is used if none of the other conditions are fulfilled.

#### Choose the entity that the screen flow operates within

Screen flows all belong to a specific entity, and all screens in the flow must belong to that entity. The entity for the flow is shown in the top right hand corner of the **Screen Flow** dialog box.



If you want to show a screen for another entity, you must call a subflow (see below) and specify the relationship for that flow. When the (primary) flow is executed, the sub-flow will be executed once for each target of the relationship.

### Add subflows to the screen flow

Subflows are separate flows that can be added to the primary screen flow. A subflow is created in exactly the same way as a normal screen flow.

If you want to have a different entity in your screen flow you must use a subflow to access that entity. When the (primary) flow is executed, the subflow will be executed once for each target of the relationship.

To include a subflow in the screen flow:

1. Click the **Flows** tab in the **Screen Flow** dialog box.
2. Select the **Relationship** for the subflow. The list of flows will change to reflect the flows available for that particular relationship.
3. Select the (sub) flow that you want to add to your (parent) flow. You can search or filter the list by typing in the **Search** text box.
4. Drag the flow to just under the shape that it should flow from in the diagram.
5. Drag from the flow shape to its related shapes to connect them.

NOTE: There can be no cyclical references in shapes, ie A requires B and B requires A.

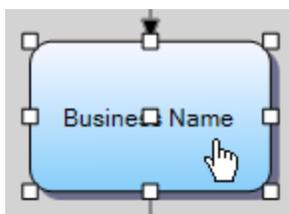
### Edit a screen flow

Once you have started creating a screen flow you will want to move and change the appearance of shapes (screens, decisions, connections etc) in the editing pane to make the flow easier to understand.

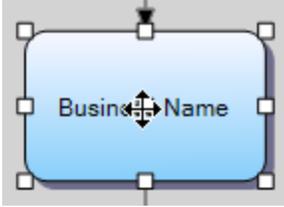
### Move a shape

To move a shape in a screen flow:

1. In the flow diagram, click anywhere on the shape to select it.



2. Select the centre white square (the hand will change to a multi-directional arrow).



3. Drag the shape to a new location in the editing pane.

### **Change the size of a shape**

To change the size of a shape in a screen flow:

1. In the flow diagram, click anywhere on the shape to select it.
2. Use one of the white squares on the edges of the shape to drag that edge of the shape inwards or outwards.

### **Delete a shape**

To delete a shape in a screen flow:

1. In the flow diagram, click anywhere on the shape to select it.
2. Press **Delete**.

## **Change how interview data is summarized and reviewed**

The summary screen is the central task page for out-of-the-box Oracle Web Determinations applications. Essentially a simple list of labels, goals, document generation triggers and flows, the summary screen provides an interface to your rulebase and screens. The summary screen also provides access to additional session management tools, including session saving and clearing, data review screens, and decision reports for completed rulebase goals.

The data review screen in Oracle Web Determinations provides a list of all the questions answered during an interview. This allows the user to revisit any question in the interview.

### **What do you want to do?**

[Create a summary screen](#)

[Add a label to the summary screen](#)

[Add a goal to the summary screen](#)

[Add a screen flow to the summary screen](#)

[Add entity-level items to the summary screen](#)

[Add a document link to the summary screen](#)

[Change the order of screens on the data review screen](#)

[Change the title of the data review screen](#)

### **Create a summary screen**

Each project will normally have a single summary screen that appears at the start and at the end of the interview, and that can also be viewed during an interview.

### **Explain this further**

At the start of an interview, the summary screen will typically:

- provide an explanation of the purpose of the rulebase
- allow the user to commence an interview of one or more rulebase goals
- provide links to additional supporting documentation

At the end of an interview, the summary screen will typically:

- display the outcome(s) of the assessment and provide links to the decision report for important goals and sub-goals
- display any warning or message text
- allow the user to generate documentation based on the information provided or inferred outcomes
- allow the user to return to all or part of the interview
- provide links to additional supporting documentation

At any point in the interview, returning to the summary screen will typically allow the user to:

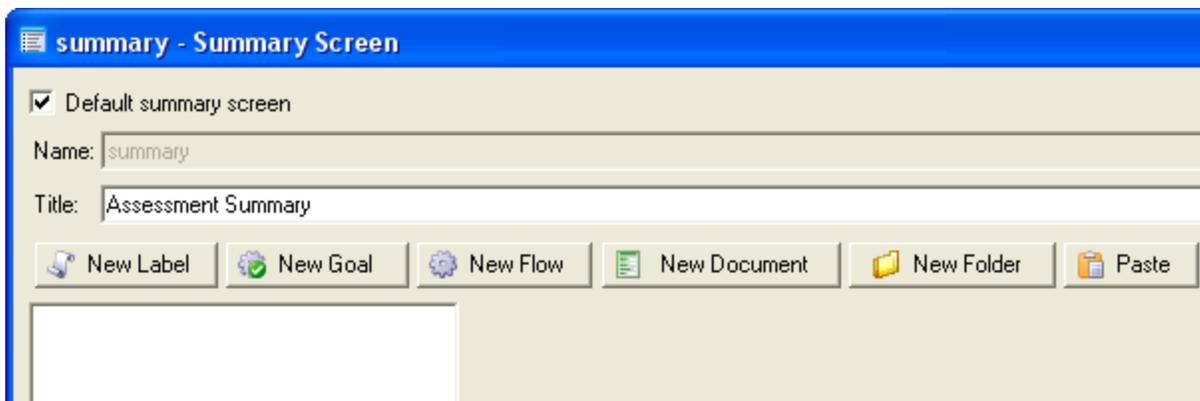
- review data already collected
- clear the session and start again
- save a copy of the assessment
- close the assessment

By default, the first screens file that is added to a project will contain an empty summary screen. If you need to manually add a new summary screen follow the steps below.

To create a new summary screen:

1. Right-click the \*.xint filename, or another folder, in the screens view and select the **New Summary Screen** menu option.

The **New Summary Screen** dialog will be displayed:



For out-of-the-box Oracle Web Determinations users, you should only define a single summary screen for your project, and leave it named "summary", which is the default value for summary screens.

2. Click **OK** and save your screens file to keep the new screen.

You can add folders to your summary screen in Oracle Policy Modeling to sort your summary screen items into manageable units, and to [allow summary screen items to be added for entities](#). You can also [add visibility attributes](#) to summary screen folders, to control the display of all elements within the folder. The folders themselves will not be displayed in Oracle Web Determinations – the summary screen will still be displayed as a flat list based on the order of all controls as if no folder structures existed. To add a new folder, click the **New Folder** button at the top of the summary screen dialog.

### Add a label to the summary screen

Labels are used to provide headings, plain text and HTML paragraphs on the summary screen.

To add and edit a new label:

1. Open the summary screen editor dialog by double-clicking on the summary screen entry in the main list of screens for your screens file.
2. Click the **New Label** button at the top of the summary screen dialog. You can then edit the label control by selecting it in the list of summary screen items in the left hand pane. The details for the label control will appear on the right side of the screen edit dialog.

The screenshot shows the 'Label Control' dialog box. At the top, there is a title bar with the text 'Label Control' and five buttons: 'Down', 'Up', 'Cut', 'Copy', and 'Delete'. Below the title bar, the dialog is divided into two main sections: 'Appearance' and 'Visibility'.  
In the 'Appearance' section:  
- There is a 'Text' label followed by a text input field containing the text 'new label'.  
- Below the text field is a 'Style' dropdown menu currently set to 'Normal'.  
- There is an 'Is HTML' checkbox, which is currently unchecked.  
- Below the checkbox are two text input fields labeled 'CSS Class' and 'CSS Style'.  
In the 'Visibility' section:  
- There is a 'Visibility:' label followed by an 'Attribute' text input field and a small button with three dots (...).  
- Below the attribute field is a 'Default State' dropdown menu set to 'Visible', with a note '(if attribute is unknown or uncertain)' to its right.

3. Specify the **Text** for the label.
4. Select the **Style** of the label from the drop-down list. Suggestions on the use of heading styles can be found in [Use headings to convey meaning about the screen](#).
5. If required, select the **Is HTML** checkbox. See [Change the appearance of text](#) for more information on this setting.
6. If required, enter a **CSS Class** and/or **CSS Style**. See [Change the appearance of a control](#) for more information on this setting.
7. If required, specify a **Visibility attribute**. See [Control the visibility of summary screen elements](#) for more details.
8. Click **OK**.

## Add a goal to the summary screen

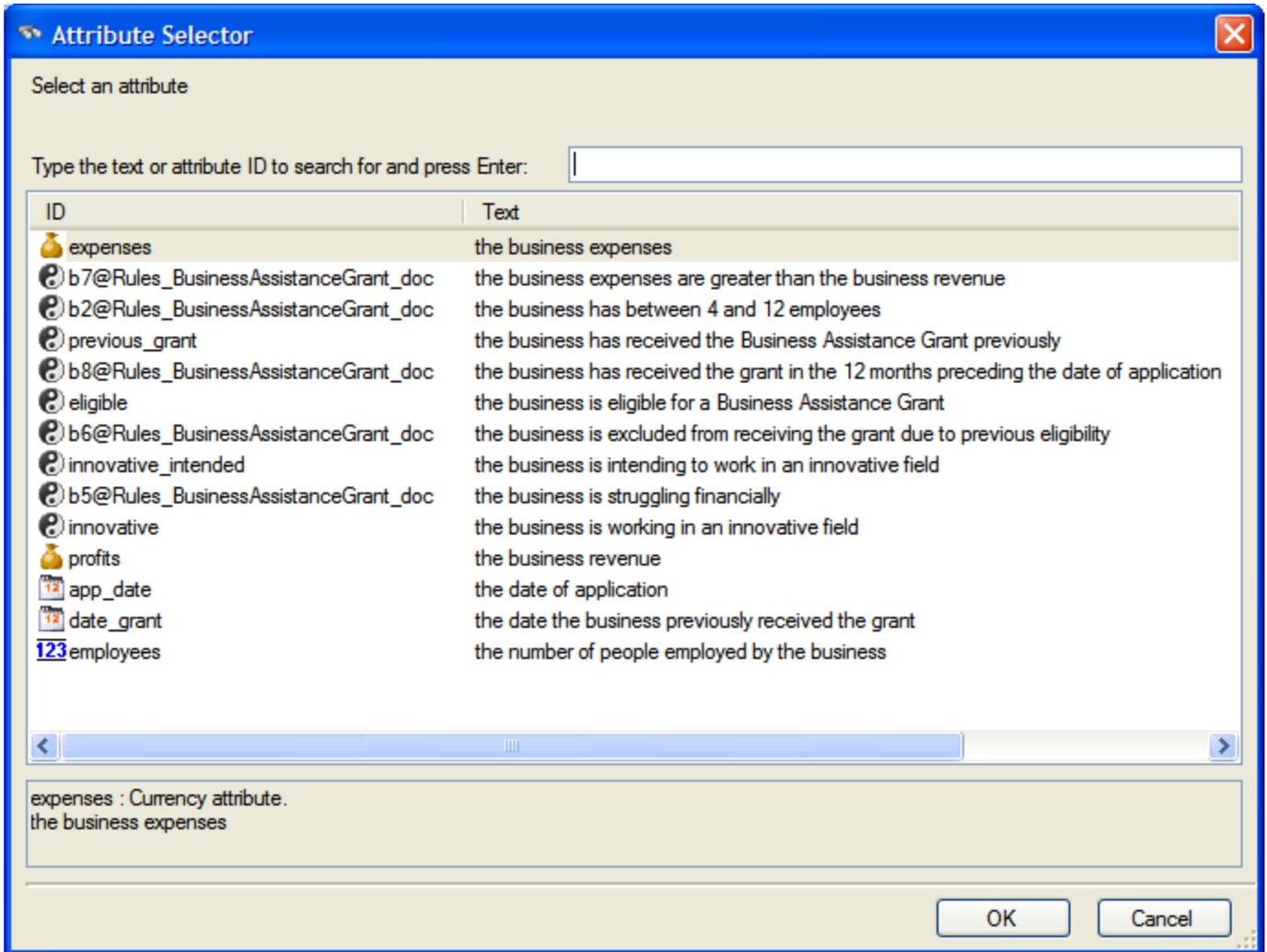
Goals are used to provide an entry point into an assessment or an entry point into a decision report. To add and edit a new goal:

1. Open the summary screen editor dialog by double-clicking on the summary screen entry in the main list of screens for your screens file.
2. Click the **New Goal** button at the top of the summary screen dialog. You can then edit the action control by selecting it in the list of summary screen items in the left hand pane. The details for the action control will appear on the right side of the screen edit dialog.

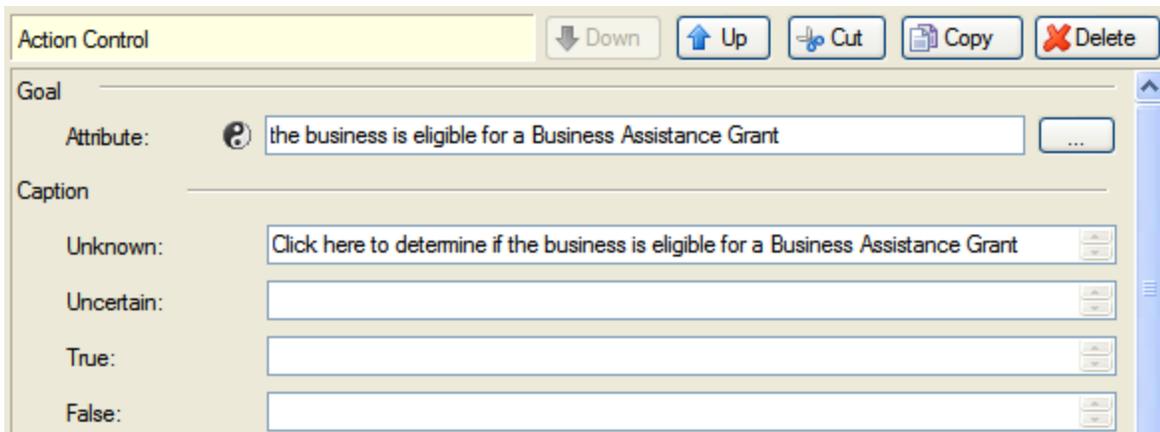
The screenshot shows the 'Action Control' dialog box. At the top, there is a toolbar with buttons for 'Down', 'Up', 'Cut', 'Copy', and 'Delete'. Below the toolbar, the dialog is organized into several sections:

- Goal:** Contains an 'Attribute' text field with a browse button (...).
- Caption:** Contains four text fields labeled 'Unknown:', 'Uncertain:', 'True:', and 'False:', each with a browse button (...).
- Appearance:** Contains a checkbox 'Is HTML', a 'CSS Class' text field, and a 'CSS Style' text field.
- Visibility:** Contains an 'Attribute' text field with a browse button (...), and a 'Default State' dropdown menu set to 'Enabled' with a note '(if attribute is unknown or uncertain)'.

3. Click the browse button next to the **Attribute** text field. This will open the **Attribute Selector** dialog box.



4. Select the attribute you want as your goal attribute from the list of attributes. Click **OK**. In the **Action Control** pane, the attribute you selected now appears in the **Attribute** field.



5. Enter **Captions** to define the text that will be displayed when the goal attribute has uncertain, unknown, true and false values. (The **Unknown** caption text is defaulted for you based on the attribute name.)
6. Select the **Is HTML** checkbox if your label/control is HTML. See [Change the appearance of text](#) for more information on this setting.
7. Specify a **CSS Class** and/or **CSS Style** if required. See [Change the appearance of a control](#) for more information on this setting.
8. If required, specify a **Visibility attribute**. See [Control the visibility of summary screen elements](#) for more details.
9. Click **OK**.

### Add a screen flow to the summary screen

If an [interview screen flow has been defined](#), it needs to be added to the summary screen so that it can be accessed in a Oracle Web Determinations investigation.

To add a screen flow to the summary screen:

1. Open your screens file and open your summary screen. Select the **New Flow** button at the top of the summary screen dialog. A Flow Control will be created.

The screenshot shows a 'Flow Control' dialog box. At the top, there is a title bar with the text 'Flow Control' and five buttons: 'Down', 'Up', 'Cut', 'Copy', and 'Delete'. Below the title bar, the dialog is divided into sections. The first section is titled 'Flow' and contains a 'Flow name' text field with the value 'New Flow' and a small '...' button to its right. Below this is a 'Caption' text field. The next section contains a checkbox labeled 'Is HTML', followed by 'CSS Class' and 'CSS Style' text fields. The 'Visibility' section includes an 'Attribute' text field with a '...' button and a 'Default State' dropdown menu currently set to 'Enabled', with the text '(if attribute is unknown or uncertain)' to its right.

2. Select the **Browse** button next to the **Flow name** text field and select your flow from the list of pre-existing flows in the **Flow Browser** dialog box. Click **OK**.  
In the **Flow Control** pane, the attribute you selected now appears in the **Flow name** field.
3. Modify the **Caption** if necessary. (By default this is set to the flow name.) This is the text that will appear for the link to the flow on the summary screen.
4. Select the **Is HTML** checkbox if your label/control is HTML. See [Change the appearance of text](#) for more information on this setting.
5. Specify a **CSS Class** and/or **CSS Style** if required. See [Change the appearance of a control](#) for more information on this setting.
6. If required, specify a **Visibility attribute**. See [Control the visibility of summary screen elements](#) for more details.
7. Click **OK**.

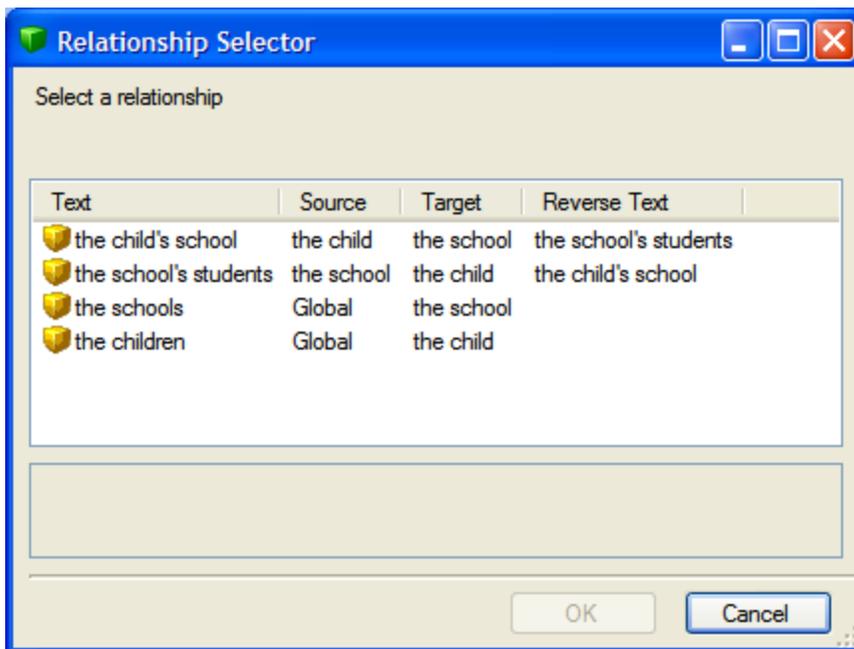
## Add entity-level items to the summary screen

Labels, goals and screen flows can also be added to the summary screen that operate within an entity. To do this, entity-level items must be grouped within a summary screen folder which is associated with the entity.

To create a summary screen folder and associate it with an entity:

1. Open the summary screen editor dialog by double-clicking on the summary screen entry in the main list of screens for your screens file.
2. Click **New Folder** and give the new folder an appropriate name.
3. In the **Summary Screen Folder** properties window, click on the **Browse** button to the right of **Relationship**.
4. In the **Relationship Selector**, select the relationship in which the entity-level items will function on the summary screen (most often this will be the entity's **containment relationship**, but it need not be), and click **OK**.

Note that the source entity for the selected relationship must be at the same entity level that the new summary screen folder is in. For instance, in our example the source entity for the relationship is the global entity, which is the same entity level as the base level of the summary screen, in which the new Children folder has been added.



5. The summary screen folder is now associated with the target entity for the relationship, and **labels**, **goals** or **screen flows** which operate at the level of that entity may now be added within the folder. Oracle Policy Modeling will now expect all goals or screen flows added within this folder to be within the entity, however labels may use **text substitution** using attribute values from the entity's parents (eg the source entity of the relationship you selected).

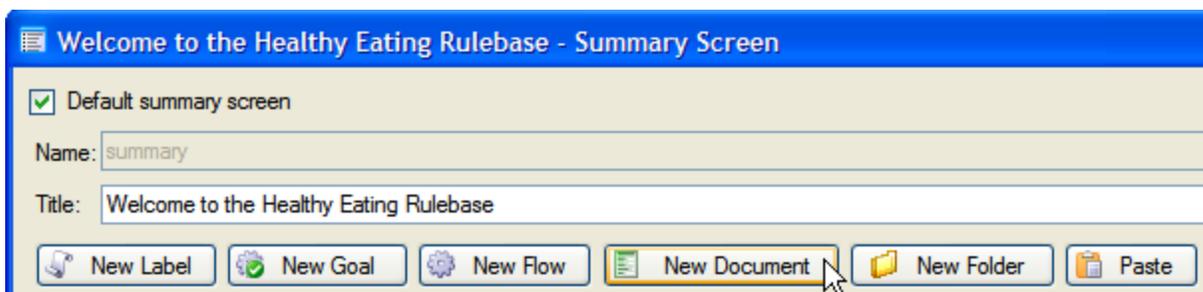
Note that summary screen folders with relationships may be nested, if this structure is reflected in the relevant entity relationships.

## Add a document link to the summary screen

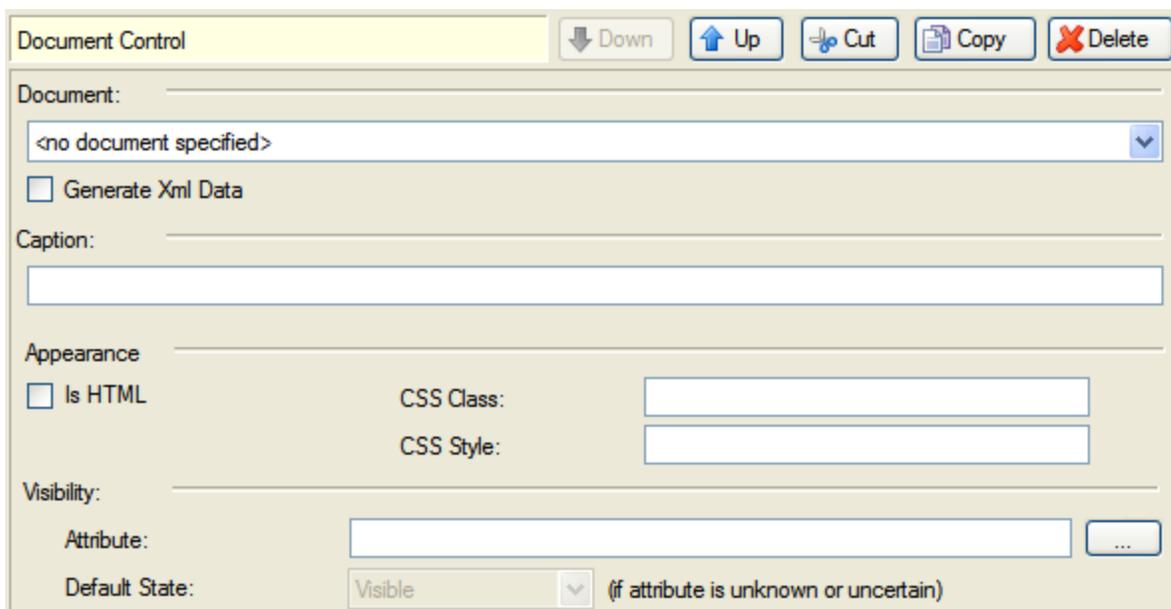
A document link needs to be added to the summary screen to allow the user to generate and view an interview document. (To do this you need to have already created the interview document that you want to link to. For more information on this, see [Create, update or delete an interview document.](#))

To add and edit a document link:

1. Open the summary screen editor dialog by double-clicking on the summary screen entry in the main list of screens for your screens file.
2. Click the **New Document** button at the top of the summary screen dialog.



3. Select the document control in the list of summary screen items in the left hand pane. The details for the document control will appear on the right side of the screen edit dialog.



4. Select the document you want to link to from the **Document** drop down list. (This list contains all of the documents in the DocGen folder in the Screens file.)
5. Specify a **Caption** for the document control. This is the text which will be displayed on the summary screen as a link for the generated document.
6. Click **OK**.

### Additional settings

In the Document Control window there are additional settings that you can specify for your document:

Setting	Description
Generate Xml Data	When checked, clicking the document link on the summary screen will generate a raw XML representation of the session that can be saved. This is useful not just for debugging purposes, but for importing into the BI Publisher tool in Word as <a href="#">sample data</a> .
Is HTML	When selected, indicates that the caption contains HTML tags. These are used to change the appearance of the document link on the summary screen. See <a href="#">Change the appearance of text</a> for more information on this setting.
CSS Class and CSS Style	Using these settings, the appearance of the document link can be customized using Cascading Style Sheet (CSS) classes and styles. See <a href="#">Change the appearance of a control</a> for more information.
Visibility Attribute and Default State	These settings are used to control the visibility of the document link on the summary screen. See <a href="#">Control the visibility of summary screen elements</a> for more details.

### Change the order of screens on the data review screen

The data review screen is the screen that is displayed when you click the "Data Review" link in Oracle Web Determinations. The order that the screens are listed on the data review screen in Web Determinations is determined by the order of screens defined in the screen order in the screens file (regardless of whether a screen order or screen flow is being used to drive the interview).

So to change the order of screens on the data review screen:

1. First check that you have a screen order defined in your screens file. By default, the first screens file that is added to a project will contain a default screen order (labeled **Data Review** in the screens view). This screen order is automatically defined as being the order of the screens in the **Question Screens** folder. If you don't have a screen order defined, Oracle Web Determinations will simply display the screens in a randomly ordered list which can make it difficult to find the attribute/screen you are interested in from the data review screen. See [Create a new screen order](#) for more information.
2. [Edit the screen order](#).

### Change the title of the data review screen

To change the title of the data review screen as it appears in Oracle Web Determinations:

1. Open your screens file in Oracle Policy Modeling, then double click the **Data Review** entry.
2. In the **Data Review - Screen Order** dialog, change the name in the **Data Review Title** field, then click **OK**.

### Check attribute inclusion on interview screens

In Oracle Policy Modeling you can see which attributes are collected on screens, and which attributes on screens have broken references. There are also occasions when you might want to collect an attribute on multiple screens.

### What do you want to do?

[View a list of attributes that are not collected on any interview screens](#)

[View a list of inferred attributes that are collected on interview screens](#)

[Find and fix any broken attribute references on screens](#)

[Collect an attribute on multiple screens](#)

### View a list of attributes that are not collected on any interview screens

An Uncollected Attributes Report lists all base level attributes not collected on a screen. This is useful for checking that your question screens include all base level attributes. (If not, and an uncollected attribute is required by the question search, that attribute will appear on an automatic screen in Oracle Web Determinations.)

To run an uncollected attributes report, in Oracle Policy Modeling select **Reports | Uncollected Attributes**.

### View a list of inferred attributes that are collected on interview screens

An Inferred Screen Attributes Report shows a list of inferred attributes which appear on screens. Inferred attributes are attributes which are proved by other attributes in the rulebase.

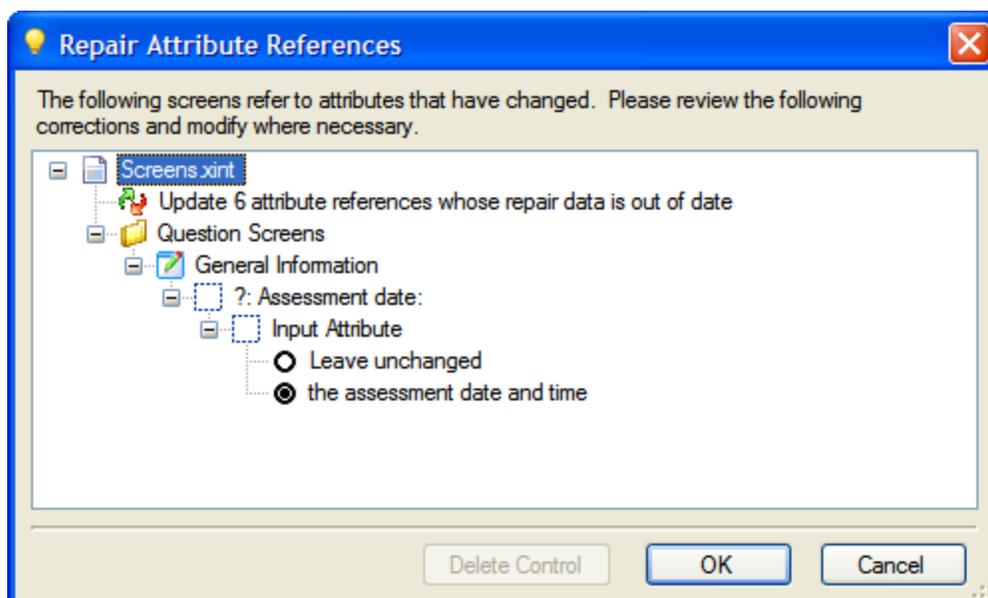
To run an inferred screen attributes report, in Oracle Policy Modeling select **Reports | Inferred Screen Attributes**. When generating this report you have the option to include base level attributes proven by shortcut rules.

### Find and fix any broken attribute references on screens

A broken attribute reference occurs on a screen when the attribute text has been changed in the source document (including properties files) without a corresponding update to files that referenced that attribute text.

To find and fix any broken attribute references on screens:

1. In Oracle Policy Modeling, select **Tools | Repair Attribute References...**  
The **Repair Attribute References** dialog box will open.



2. For each broken attribute reference, choose whether to leave it unchanged, or update the attribute on the screen to use the new text (default option). (You also have the option to delete selected controls using the **Delete Control** button.)
3. Click **OK**.

### Collect an attribute on multiple screens

When developing a rulebase application, there may be situations where it makes sense to have the same attribute collected on multiple screens.

For example, assume you are creating a rulebase and interview about work related travel. The interview needs to determine the number of nights that the participant is going to spend away, and the type of travel (domestic or international). If international travel is being undertaken, then the destination country must also be known.

You can see from a user interface point of view, it makes sense to develop two screens, one for collecting domestic travel details, and one for collecting international travel details:

#### Domestic Travel Details

- Number of nights away

#### International Travel Details

- Number of nights away
- Destination country

When an attribute is collected on multiple screens, the [screen order](#) defined in the screens file dictates which screen will be shown first, and any subsequent screens will only be shown if some other attribute on the screen is required and hasn't already been collected.

## Create, update or delete interview help

Integrated interview help text, also known as commentary, is provided in an Oracle Web Determinations application to help people understand the questions that they are being asked, and the screens they are being presented.

Oracle Web Determinations uses HTML documents for each question on the screen. In other words, for every question there is a corresponding HTML document. Clicking on a question will load the help for that question.

In addition to question-related help, help can also be provided for at the screen level or at the word-level. Clicking on the screen title or word, will open the help for that screen or word.

## What do you want to do?

[Generate commentary files for attributes and screens](#)

[Create commentary for a word in a label or question](#)

[Make the commentary open in a new window](#)

[Update a commentary file](#)

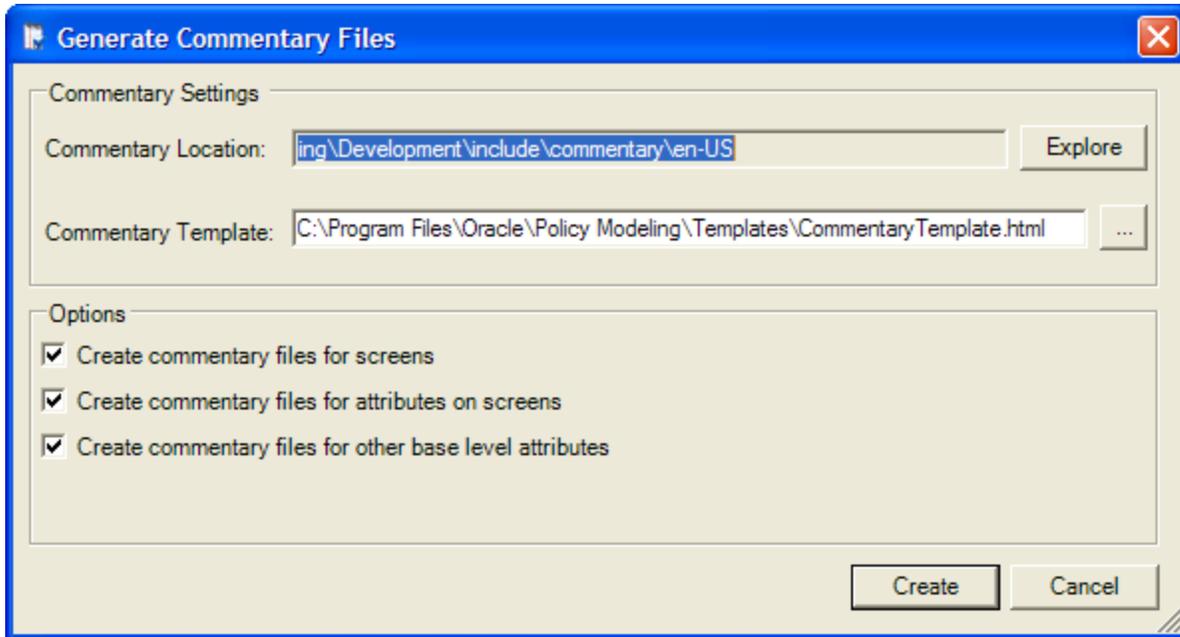
[Delete a commentary file](#)

[Localize a commentary file](#)

## Generate commentary files for attributes and screens

You can automatically generate commentary files for your project from Oracle Policy Modeling. To do this:

1. Select **Build | Generate Commentary Files...**  
This opens the **Generate Commentary Files** dialog box.



By default, the commentary files will be located in `\Development\include\commentary\<rulebase language>`. To view the files in this location, click the **Explore** button.

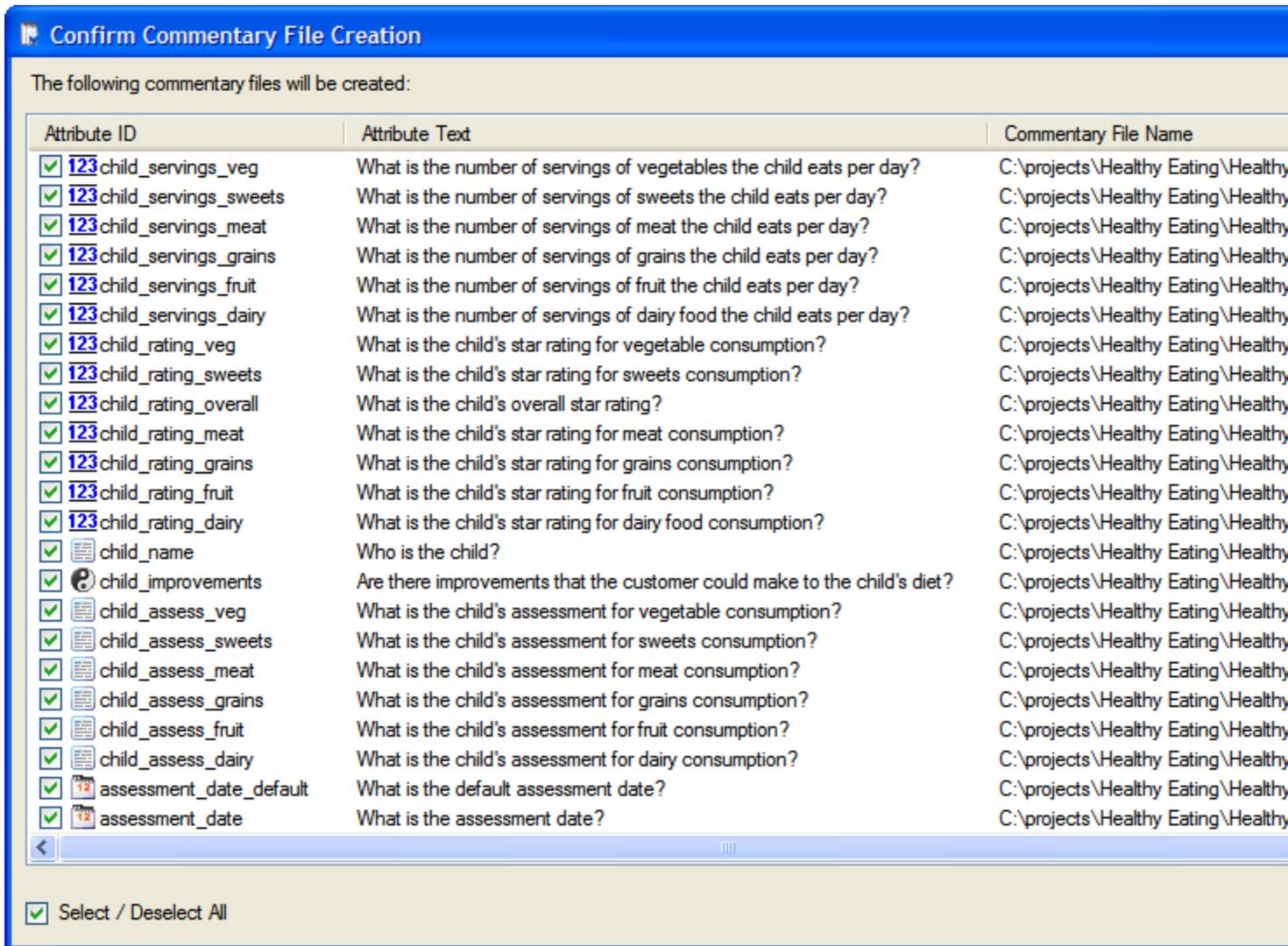
NOTE: The default commentary plug-in fetches HTML files out of the rulebase archive here. This default commentary plug-in can be replaced with a custom-developed one that returns commentary from some other location like a database or external web server. For more information on using another commentary plug-in, see the [Oracle Policy Automation Developer's Guide](#).

By default, the commentary template will be located in `C:\Program Files\Oracle\Policy Modeling\Templates`. This template should include any styles, headings and information that are to be reasonably common across all attributes. To select a different commentary template, use the browse button to locate and select another file.

2. Select whether you want to create commentary files:
  - \* for screens
  - \* for attributes on screens
  - \* for other base level attributes (including attributes for which automatic screens will be shown)

3. Click **Create**.

The **Confirm Commentary File Creation** dialog box will open. This displays a list of attributes and screens that meet your chosen criteria. The commentary file name is `<attribute id>.html`. (NOTE: If you change the commentary file name, the commentary will not be displayed for that attribute/screen.)



4. Ensure that the check box is ticked for any attributes and screens that you want to create commentary files for, then click **OK**. (The "default" (this rulebase) commentary file which is created will only appear on the locale selection screen. This screen is only displayed if translations have been added to the rulebase, ie if the rulebase can be run in more than one language.)
5. You will then be advised when the commentary files have been generated and asked if you want to view them. Click **OK** to open the folder containing the commentary file, or click **Cancel** to return to Oracle Policy Modeling.

#### Create commentary for a word in a label or question

You can also create "per-word" commentary for words in labels and questions. For example, the question "What is the claimant's weekly net pay?" could have the word "pay" as a link to commentary which provides a definition of the term.

## Further Information

Would the claimant like to receive further information regarding healthy eating? \*  Yes  No

To add commentary for a word in a question or label:

1. In Oracle Policy Modeling, open your screens file and select the relevant question or summary screen.
2. Select the label or question that you want to add per-word commentary to.
3. Change the text of the label or question to include the HTML tag for the commentary in the following format: `<a href="..\..\..\commentary/<project name>/<rulebase language>?target=<file name>"><link text></a>`  
For example, "What is the claimant's weekly net `<a href="..\..\..\commentary/Web Determinations/en-US?target=pay">pay</a>?"`.

NOTES:

(i) If the link is on the summary screen one less "../" is required (because the link text is relative to the screen URL path). For example, `<a href="..\..\..\commentary/Web Determinations/en-US?target=pay">pay</a>`.

(ii) To make the per-word commentary appear in a separate pop-up window, include "target="\_blank"" in the HTML, eg "What is the claimant's weekly net `<a href="..\..\..\commentary/Web Determinations/en-US?target=pay" target="_blank">pay</a>?"`.

4. Click **OK**.
5. Create a commentary file named exactly the same as the filename specified in the HTML text (eg "pay.html").
6. Put the commentary file in the commentary directory for the project (ie `\Development\include\commentary\<rulebase language>`).

NOTE: When using per-word commentary in question text (ie on an attribute input control), the standard commentary file for that attribute must not exist (that is, either delete it or do not create it when generating the commentary files). Otherwise the control will render somewhat unusably as a link-within-a-link.

### Make the commentary open in a new window

The default behaviour is for commentary to appear in the same window (that is, as a pane on the right hand side of the window) when a question or screen is clicked on. If you want to have the commentary open in a new window you need to make the following change to the `appearance.properties` file:

1. Open the **appearance.properties** file which is located in `\Release\web-determinations\WEB-INF\classes\configuration` for the project.
2. Change the **opa-commentary-type** setting from "frameset" to "popup".
3. Save and close the file. TIP: You will need to close and restart your Web Determinations investigation to see this change take effect.

NOTE: Setting the commentary to open in a new window doesn't work when running Web Determinations in the debugger; it only works when running Web Determinations in an external web browser.

## Update a commentary file

To update a commentary file:

1. Browse to the commentary file on your local drive: `\Development\include\commentary\<project language>`.  
(TIP: Alternatively, you can select **Build | Generate Commentary Files** and click the **Explore** button next to the **Commentary Location** field to open the directory containing the commentary files.)
2. Double click the file to open it.
3. Edit the file as required and then save it.

NOTE: Changes to commentary files will not appear until you re-build and start a new Web Determinations session.

## Delete a commentary file

To delete a commentary file:

1. Browse to the commentary file on your local drive: `\Development\include\commentary\<project language>`.  
(TIP: Alternatively, you can select **Build | Generate Commentary Files** and click the **Explore** button next to the **Commentary Location** field to open the directory containing the commentary files.)
2. Right-click and select **Delete**. Click **Yes** to confirm the file deletion.

NOTE: If there is no commentary file for a particular attribute or screen, the question or screen will not appear as a link in the application.

## Localize a commentary file

If you have added a language translation to your rulebase, you may also wish to [localize your rulebase commentary files](#) accordingly.

## Overview: The process of creating an interview document

An interview document is a document that can be generated from an interview session in Oracle Web Determinations. It provides the user with a record of the interview, including answers and conclusions, that they can view and download. Interview documents have many uses, including pre-populated claim forms and advice letters.

Oracle Policy Modeling supports the creation of HTML, RTF, PDF and Excel interview documents.

After you have authored your rulebase in Oracle Policy Modeling (including finalizing your data model) and you have tested it in Web Determinations, you can then create an interview document by following these steps:

1. [Add a new document definition to your screens file](#)  
This is where you associate a template with the document (see next step). This is also where you generate the XML schema containing all of the publicly-named attributes in the project, and specify any decision reports that you would like available to your interview document.
2. [Develop the template for your interview document](#)  
Using Microsoft Word you develop the RTF template for the interview document using the BI Publisher Template Builder and your XML Schema. Using sample data you can preview the document.
3. [Add a document link to your summary screen](#)  
You add a document link to the summary screen to enable the user to generate and view your interview document.

#### 4. [Test the generation of the document](#)

Using Web Determinations you can test that your document generates in the format and style with the content that you expected.

The Social Services Screening rulebase and the Healthy Eating rulebase that are installed with Oracle Policy Modeling (ie \Program Files\Oracle\Policy Modeling\examples\ ) are examples of complete rulebases containing interview documents.

### Create, update or delete an interview document

Customized interview documents can be generated from an Oracle Web Determinations application. Typical documents include an assessment notice and a personalized claim form.

Interview documents are defined in the screens file (associating them with [an RTF template](#)) and then integrated into Oracle Web Determinations applications using [document links on the summary screen](#).

### What do you want to do?

[Create a documents folder](#)

[Create a new interview document](#)

[Modify an interview document](#)

[Delete an interview document](#)

#### Create a documents folder

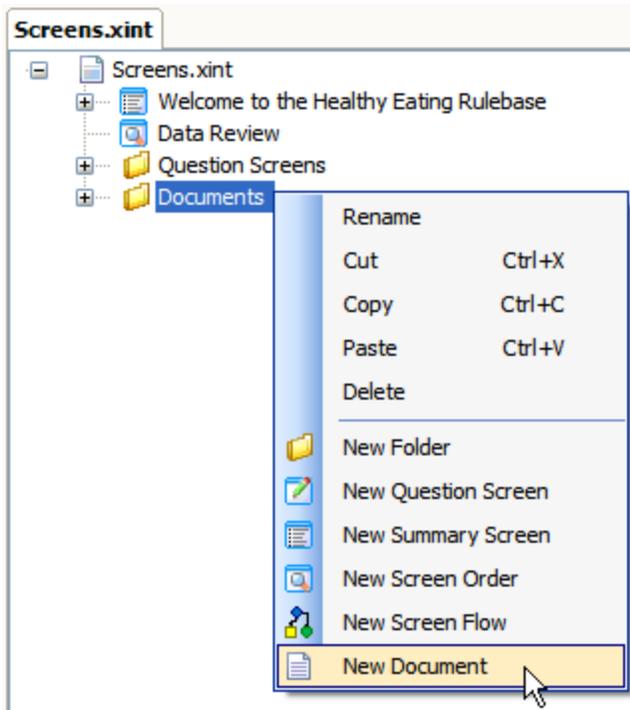
By default, the first screens file that is added to a project will contain a Documents folder. To add additional folders to your screens file:

1. Right-click the \*.xint filename, or another folder, in the screens view.
2. Select **New Folder** from the pop-up menu.
3. Enter an appropriate name for your screen folder.

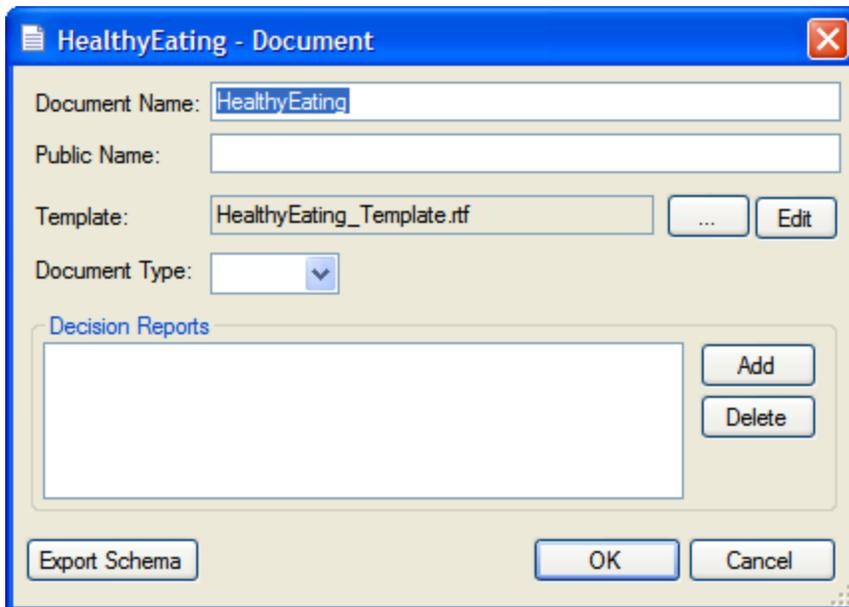
#### Create a new interview document

To create an interview document:

1. Right-click the **Documents** folder in your screens file and select **New Document** from the pop-up menu.



The Document editor will appear:



2. Enter an appropriate name for the document in the **Document Name** text box (by default this will be the name of the rulebase).
3. Optionally provide a **Public Name** for the document. NOTE: This public name must be unique across all screen files in the project.

4. Specify the **Template** file (.rtf) to be associated with the document. A default blank template is automatically created (<Project name>\_Template.rtf) in the project under \include\templates\<locale>. TIP: Use the **Edit** button to open the template for editing in Microsoft Word. See [Develop a template for an interview document](#) for more information on editing templates.
5. Select the **Document Type** from the drop-down list. The options are: Excel, HTML, PDF or RTF.
6. Add any **Decision Report** attributes that you would like available to your interview document. (Click on the **Add** button and then select the attributes from the **Attribute Selector**. The only attributes available as decision report attributes are top level and intermediate level attributes with public names. Then click **OK**.) Adding entries into the Decision Reports list not only ensures that an attribute is present in the XML, but that a full decision report is also available for it. TIP: Take care not to add unnecessary entries here, as they may slow down the document generation process.
7. Click the **Export Schema** button. This button exports an XSD representation of the document definition. (NOTE: Only those attributes with public names are included in the generated XML schema. You therefore must have a [Properties file](#) in your project which contains your public names.) This is important as you will generally need to import this XSD file into the BI Publisher tool in Word in order to develop your document template. In the **Save** dialog box, enter a name for the XML schema file, then click **Save**. (NOTE: Schema files are document, not rulebase, specific so be sure to name your schema according to the document it relates to.)
8. Click **OK**.

### Modify an interview document

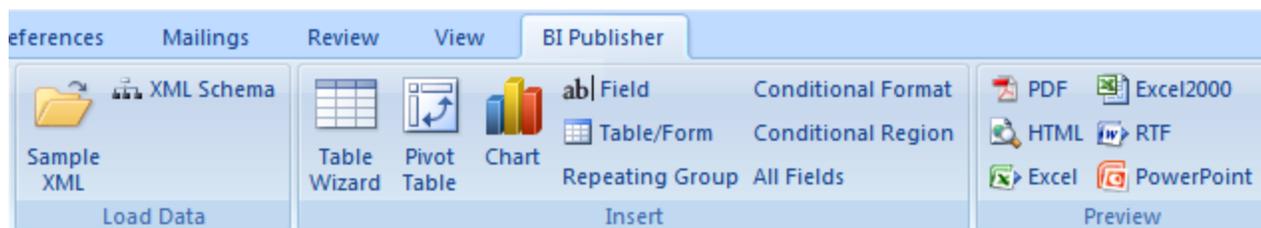
To modify an existing interview document, double-click on the document name in your screens file, or right-click and select **Open** from the pop-up menu. Make the necessary changes in the Document editor and then click **OK**.

### Delete an interview document

To delete an interview document, select the document name in your screens file and press **Delete**, or right-click and select **Delete** from the pop-up menu.

### Develop a template for an interview document

An RTF template is used to generate the interview document. This RTF file is created in Microsoft Word using the BI Publisher Template Builder. The Template Builder is a tool that simplifies the development of RTF templates.



There are 4 steps in the process of developing a template for an interview document:

1. Create a template file
2. Load the XML data into the file
3. Design the template
4. Preview the document

If you are familiar with developing BI Publisher templates and just need to know the format to use for the fields, refer to the [BI Publisher code for Oracle Policy Modeling](#) topic.

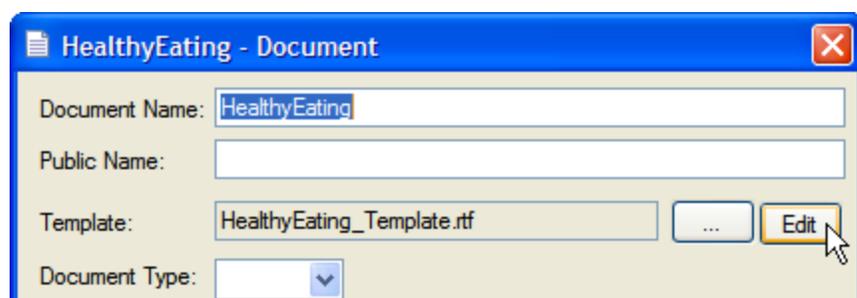
### Create a template file

A default RTF template is created automatically when you add a new document to your screens file. This file, <project name>\_Template.rtf, is located in the \include\templates\<locale> folder for the project.

Alternatively, you can reuse an existing RTF template file, but make sure that you select that file in the **Template** field in the Document editor in your screens file (it will then be automatically copied into the include\templates\<locale> folder).

### Open a template file

The template file can be opened for editing in Word by clicking on the **Edit** button in the Document editor in the screens file in Oracle Policy Modeling:



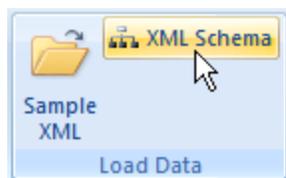
Alternatively, you can open the template file from its location in the project under \include\templates\<locale> using Microsoft Word or Windows Explorer.

### Load the XML data

In order to add fields to your template you first need to load data into the BI Publisher Template Builder in Word. The simplest way to do this is to import the XML schema file (XSD) that you created when you [added the document to your screens file](#).

### Load the XML schema

1. On the BI Publisher toolbar, select **XML Schema**:



2. In the dialog box, select the XML schema file (XSD) that you exported when you added the document to your screens file. You will be told when this data has been loaded successfully.

NOTE: Sample data can also be loaded and used in the development of the template. You would only do this though if you needed to see the full list of fields that are generated. Typically, you will only want to include the text and/or the formatted value of the attributes in the session in your template in which case the XML schema, rather than the sample XML, is considerably more user-friendly.

TIP: Even if you do want other field types in your template that are not included in the XML schema, you can use one of the fields for the attribute that is in the schema and customize it to be what you need (see below).

## Design the template

In addition to normal Word components, a template can include:

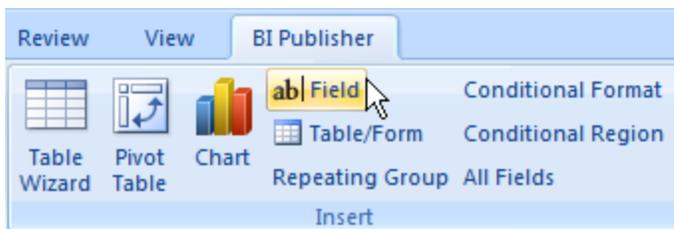
- fields
- tables
- charts
- pictures

## Insert a field

Fields in BI Publisher are used to display values and properties of attributes (global and [entity-level](#)), [decision reports](#) and [conditional text](#).

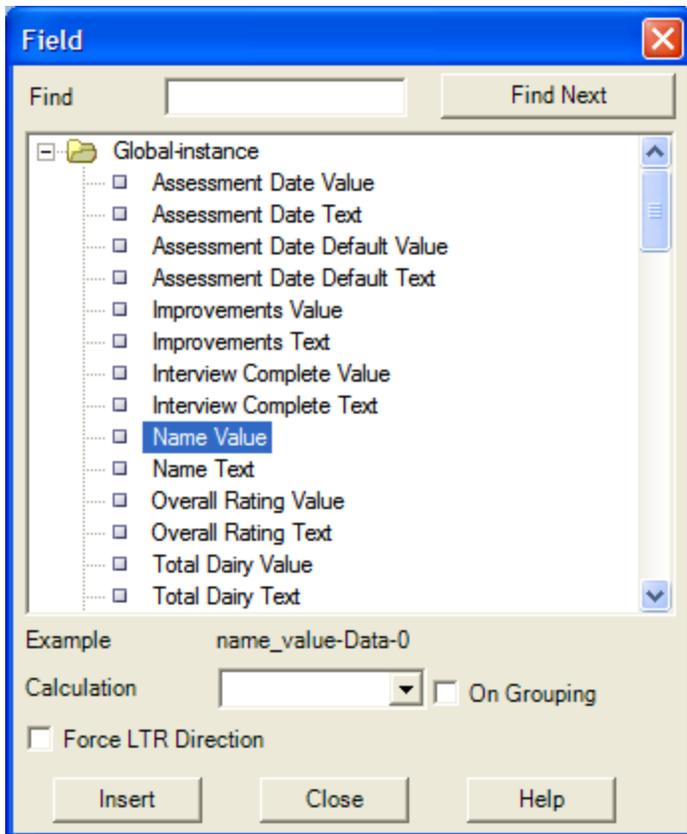
To insert a field into a template:

1. Put your cursor in the place in your document template where you would like to insert the field. Click the **Field** button on the BI Publisher menu:

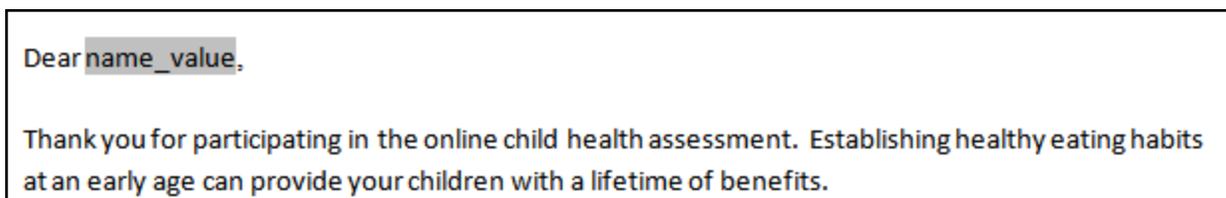


(If you have not already [loaded a data source](#), you will be prompted to do so now.)

2. In the **Field** dialog box, select the field that you want to insert (for example, to insert the text for the global attribute for the claimant's name, select the Name Value field in the Global-instance folder):



3. Either drag and drop the field to the desired location in your template, or click the **Insert** button in the Field dialog box. Close the Field dialog box. Your field should look something like this:



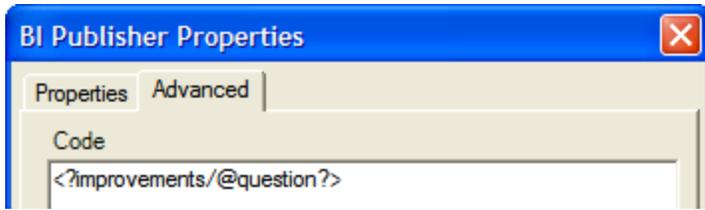
IMPORTANT NOTE: For attribute values (formatted for the project region) and attribute text this is all you need to do. For other field types (eg unformatted attribute values, attribute question text, attribute type, decision reports, conditional text etc) you need to customize a BI Publisher field (see below).

#### Customize a BI Publisher field

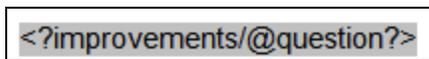
The BI Publisher code for all fields, other than attribute values (formatted) and attribute text, needs to be customized.

1. Follow the steps above for inserting a field into your template. Use either the Value or Text field for the relevant attribute (it does not matter which as you will be customizing the field anyway).
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: If double-clicking the field does not open the BI Publisher Properties dialog, re-load your XML.)

3. Click on the **Advanced** tab. The code here needs to be modified to point to the property of the attribute that you want to be displayed (note that the code is case-sensitive). For example, if you wanted to display the attribute question text for the publicly-named 'improvements' attribute, you would enter the **Code** "<?improvements/@question?>":



4. Select this text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field (either on the **Properties** or **Advanced** tab).
5. Click **OK**. The field in your template should look something like this:



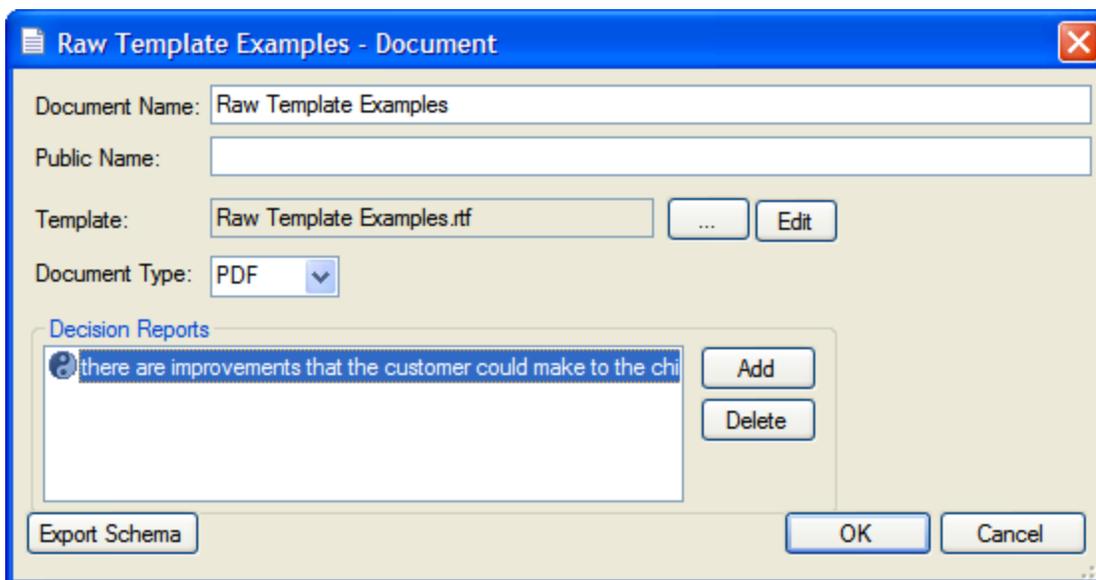
For the format required for other fields, see [BI Publisher code for Oracle Policy Modeling](#).

TIP: The [Healthy Eating example rulebase](#) that is installed with Oracle Policy Modeling, contains a Raw Template Example RTF file that you can use to copy and paste the code from for various fields. You just need to replace the attribute id (ie public name) in the code, and update the display text as necessary.

#### Insert a decision report

To insert a decision report for an attribute into your template there are 3 things you need to do:

1. In Oracle Policy Modeling, add the attribute to the [Decision Reports](#) available for the document.



2. Somewhere in your template document, have a "decision-report template" field, which tells BI Publisher how to structure and format a decision report. Follow the steps earlier for [inserting a field](#) into your template. The [field needs to be customized](#) to have the following code specified in the Advanced tab for the field:

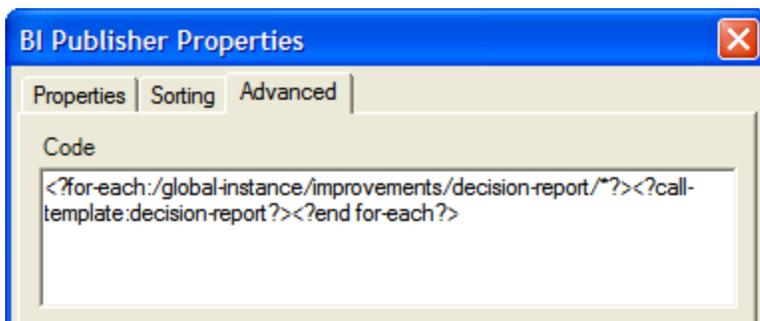
```
<?template@inlines:decision-report?>
<?if@inlines:"attribute-node"?>
<fo:list-block start-indent="{count(ancestor::attribute-node) * 7}mm">
<fo:list-item>
<fo:list-item-label>
<fo:block>*</fo:block>
</fo:list-item-label>
<fo:list-item-body>
<fo:block><xsl:value-of select="@text"/></fo:block>
</fo:list-item-body>
</fo:list-item>
</fo:list-block>
<?for-each@inlines:./attribute-node?><?call-template:decision-report?><?end for-
each?>
<?end if?>
<?end template?>
```

The **Text to display** setting (on either the **Properties** or **Advanced** tab) should be updated to say **decision-report template**.

3. In the place in your template document where you want the decision report to appear, have a "call decision report template" field which specifies the attribute ("attribute\_id") to give the decision report on. To do this, follow the steps earlier for [inserting a field](#) into your template. The [field needs to be customized](#) to have the following code specified in the Advanced tab for the field, where "attribute\_id" is replaced by the goal attribute that will be used for the decision report:

```
<?for-each:/global-instance/attribute_id/decision-report/*?><?call-template:decision-
report?><?end for-each?>
```

For example:



The **Text to display** setting (either on the **Properties** or **Advanced** tab) also needs to be updated to say **call decision report template**.

The resulting fields in your template should look like this:

decison-report template call decision report template

#### Insert conditional text

You can specify that certain text is only shown when a particular condition is met. Both simple conditions and multiple conditions are supported in BI Publisher. Conditional text can be achieved by:

1. Using the BI Publisher Conditional Region dialog, or
2. Manually defining the conditional fields.

The Conditional Region dialog can be used when the condition is that an attribute EQUALS a value. It can also be used for greater/less than comparisons for number variables provided you select 'Number' from the drop-down list (see below).

To insert conditional text using the Conditional Region dialog:

1. In your template write the text that you would like displayed under certain conditions.

If you would like help making these improvements, please contact your local health representative.

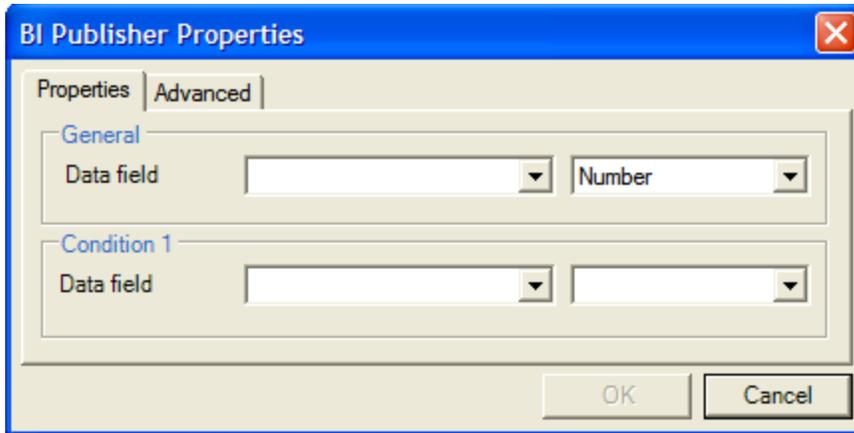
2. Select the text.

If you would like help making these improvements, please contact your local health representative.

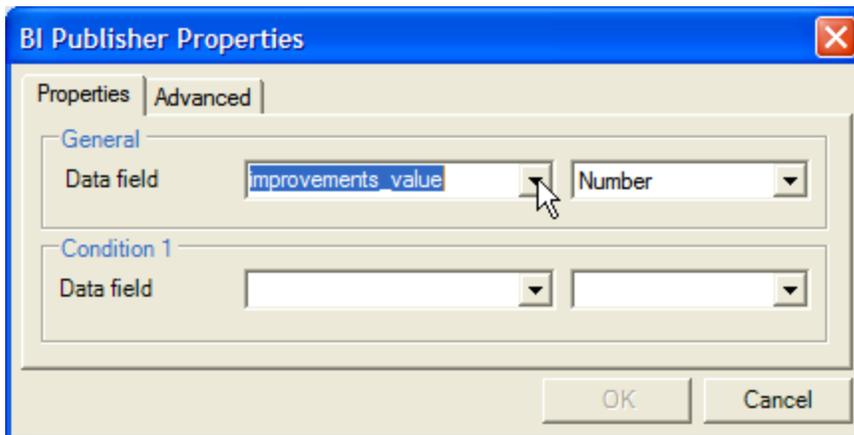
3. Click on the **Conditional Region** button on the BI Publisher menu:



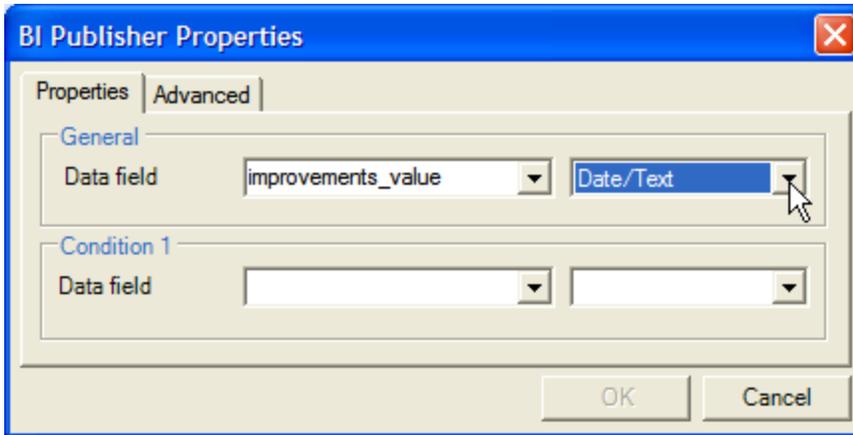
The following BI Publisher dialog will be displayed:



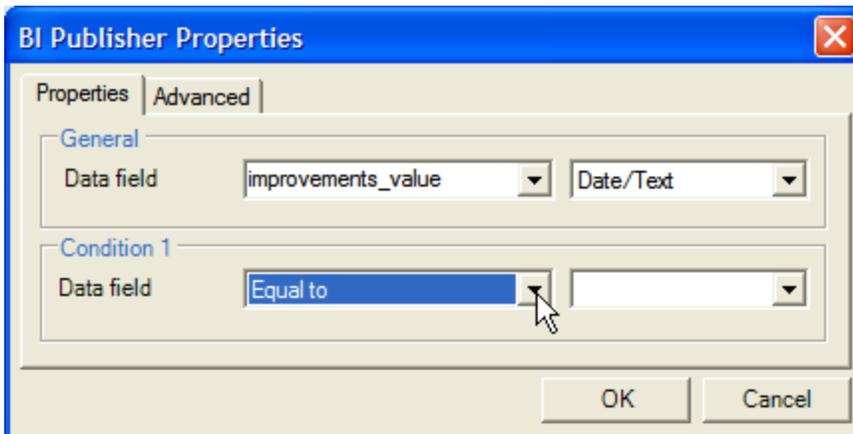
4. In the **General** box, select the attribute whose value you want to base the condition on from the **Data field** drop-down list. (TIP: Make sure you have the [XML schema loaded](#) in your document, not the sample data, to make it easier to use and find attributes in this drop-down list.)



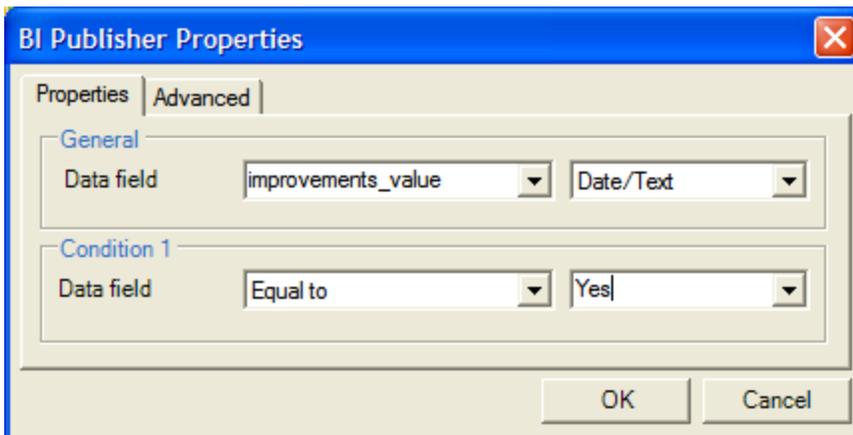
5. Also in the General box, select the attribute type from the drop-down list. NOTE: These are BI Publisher types so use "Number" for Oracle Policy Automation number variables and "Date/Text" for Booleans and for all other variable types including currency.



6. In the **Condition 1** box, select the condition that applies to the attribute from the **Data field** drop-down list.



7. Also in the Condition 1 box, type in the condition value (formatted).



8. Click **OK**. The following BI Publisher fields will appear around the conditional text in your template (C stand for Condition, EC stands for End Condition):

**C** If you would like help making these improvements, please contact your local health representative.**EC**

Alternatively, you can manually define your own conditional region by following the format described below.

Simple conditions take the following format:

- `<?if: condition?>` Display text when condition met `<?end if?>`

Multiple conditions take this format:

- `<?choose: ?>` `<?when: condition?>` Display text when condition met `<?end when?>` `<?otherwise: ?>` Alternate display text `<?end otherwise?>` `<?end choose?>`

Note that the *conditions* in these elements use [unformatted attribute values](#) and need to follow a particular syntax (see examples [BI Publisher code for Oracle Policy Modeling](#) for more information).

To manually insert simple conditional text:

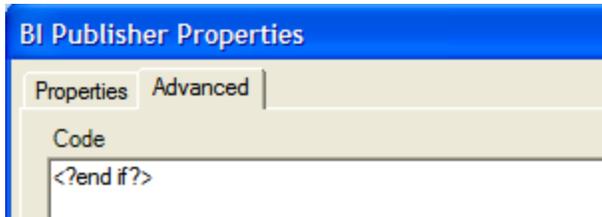
1. [Insert a field](#) for the relevant attribute value into your template.
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: If double-clicking the field does not open the BI Publisher Properties dialog, re-load your XML.)
3. Click on the **Advanced** tab. Enter the code for the start of the condition (eg `<?if: attribute_id/value='value'?>`):



4. Select the code text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field. Click **OK**.
5. In your template document after the if condition field, enter the text that you want to be displayed when the condition is met.

`<?if:improvements/value='true'?>`Please make an appointment with one of our friendly dieticians to discuss how you could improve your family's health.

6. After the display text, insert another field into your template (just as you did in step 1).
7. Double-click on the field to open the **BI Publisher Properties** dialog box.
8. Click on the **Advanced** tab. Enter the code for the end of the if condition, ie `<?end if?>`:



9. Select the code text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field. Click **OK**. Your template should look something like this:

`<?if:improvements/value='true'?>`Please make an appointment with one of our friendly dieticians to discuss how you could improve your family's health. `<?end if?>`

Further examples of conditional text, including ones with multiple conditions and conditional formatting, are given in [BI Publisher code for Oracle Policy Modeling](#). The BI Publisher Users Guide also provides further information on conditional formatting.

#### Insert entity-level attributes

To insert entity-level attributes in your template, you can use the **Repeating Group** button on the BI Publisher menu:



See the Template Builder for Microsoft Word help file for more information on how to set up your entity attributes in this way.

Alternatively, you can add entity-level attribute values and properties by following the format described below.

Entity-level attribute values and properties can be added to template documents in the same way as global attribute values and properties, but the group needs to:

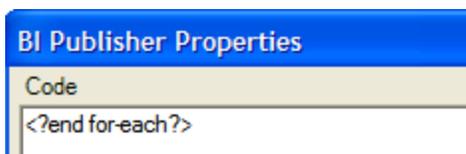
- be preceded by `<?for-each:entity_id?>`, and
- be followed by `<?end for-each?>`

To insert an entity-level attribute:

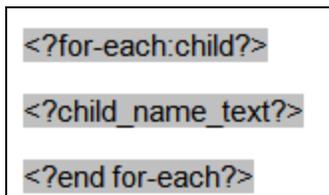
1. [Insert a field](#) for the relevant entity-level attribute value into your template.
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: If double-clicking the field does not open the BI Publisher Properties dialog, re-load your XML.)
3. Click on the **Advanced** tab. Enter the code that defines which entity the attribute belongs to, ie `<?for-each:entity_id?>`:



4. Select the code text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field. Click **OK**.
5. In your template document, on a new line after the "for-each" field, insert another field for the relevant entity-level attribute into your template (just as you did in step 1). If this field is for a formatted attribute value or attribute text continue on to the next step. If the field is for another type of attribute value/property (eg unformatted attribute value, attribute question text, attribute type) you will need to [customize the BI Publisher field](#).
6. In your template document, on a new line after the entity-level attribute field, insert another field for the relevant entity-level attribute into your template.
7. Double-click on the field to open the **BI Publisher Properties** dialog box.
8. Enter the code for the end of the "for-each" field, ie <?end for-each?>:



9. Select the code text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field. Click **OK**. Your template should look something like this:



Entity-level attributes can be displayed in several different ways.

To have information grouped by entity, have the "for-each" field around the whole group of attribute values/properties in your template document. For example:

```
<?for-each:child?>  
<?child_name_text?>  
<?child_rating_overall_text?>  
<?child_rating_overall/@question?> <?child_rating_overall_value?>  
<?end for-each?>
```

This would display as:

The child is Hayden.

Hayden's overall star rating is 4.

What is Hayden's overall star rating? 4

The child is Courtney.

Courtney's overall star rating is 2.

What is Courtney's overall star rating? 2

To have information grouped by attribute, have a "for-each" field around each individual attribute value/property in your template document. For instance:

```
<?for-each:child?> <?child_name_text?> <?end for-each?>  
<?for-each:child?> <?child_rating_overall_text?> <?end for-each?>  
<?for-each:child?> <?child_rating_overall/@question?> <?child_rating_overall_  
value?> <?end for-each?>
```

This would display as:

The child is Hayden.

The child is Courtney.

Hayden's overall star rating is 4.

Courtney's overall star rating is 2.

What is Hayden's overall star rating? 4

What is Courtney's overall star rating? 2

To display entity-level attributes in table form, the first cell in the row needs to start with the <?for-each:entity\_id?> field, and the last cell in the same row needs to end with the <?end for-each?> field (with the entity-level attribute fields in between). For example, a table like this in your template document:

Child Name	Rating
<?for-each:child?> <?child_name_value?>	<?child_rating_overall_value?> <?end for-each?>

This would display as:

Child Name	Rating
Hayden	4
Courtney	2

To display entity-level attributes in table form sorted alphabetically by entity name, follow the directions above but add an additional "sort" element to the opening "for-each" field:

```
<?sort:entity_name_id_value;'ascending';data-type='text'?>
```

For example,

Child Name	Rating
<pre>&lt;?for- each:child?&gt;&lt;?sort:child_name_value ;'ascending';data-type='text'?&gt; &lt;?child_name_value?&gt;</pre>	<pre>&lt;?child_rating_overall_value?&gt; &lt;?end for-each?&gt;</pre>

This would display as:

Child Name	Rating
Courtney	2
Hayden	4

TIP: Sometimes having the full display text for the "for each" fields at the start and end of the rows can upset the formatting of your table. If so, replace the full display text with an abbreviation (eg "F" for the <?for-each:entity\_id?> field, and "E" for the <?end for-each?> field ). See the Combined Form in the Social Services Screening rulebase that is installed with Oracle Policy Modeling for an example of this.

### Insert a table

To have a table in your template you can either:

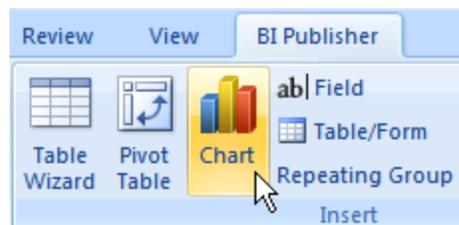
- add a native Microsoft Word table and then add the necessary BI Publisher fields to it, or
- use the **Table Wizard** on the BI Publisher menu, or
- use the **Table/Form** button on the BI Publisher menu (advanced).



For more information on using the BI Publisher table formats, see the Template Builder for Microsoft Word help file.

## Insert a chart

To have a chart in your template you must create it using the **Chart** button on the BI Publisher menu (you can't use a native Microsoft Word chart).



You then have the option of creating the chart yourself using the **Builder**, or, if you know the code for the chart that you want, adding the code directly to the **Advanced** tab.

For more information on inserting a chart using the chart builder, see the Template Builder for Microsoft Word help file.

An example of creating a pie chart by adding code to the Advanced tab is shown in the Raw Template Examples file in the Healthy Eating rulebase that is installed with Oracle Policy Modeling.

## Insert a repeating picture

You can have a picture repeated in a document depending on the value of a particular attribute. To do this you need to specify BI Publisher code in the **Format AutoShape | Alt Text** field for the image in the template.

An example of displaying a number of star images to represent a child's diet rating is shown in the Raw Template Examples file in the Healthy Eating rulebase that is installed with Oracle Policy Modeling.

## Preview the document

Using BI Publisher you can preview your RTF template using "real" data. To do this you generate some sample data, then load it into your template and then preview the output.

## Generate the sample data

Sample data can be generated from an Oracle Web Determinations session. To do this:

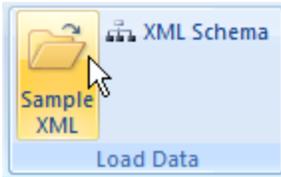
1. Open the [document control on the summary screen](#).
2. Select the **Generate Xml Data** checkbox, then click **OK**.
3. Select **Build | Build and Debug**. In the **Debug Options** dialog box, select the option to debug **With screens**, and select the option **Build and deploy with built-in Oracle Web Determinations**. It is important that you also select the option to **Replace deployed version of Web Determinations**.
4. In Web Determinations, enter your data until a conclusion is reached.
5. On the summary screen, click on the document link. Save the XML file from the session.

NOTE: When the XML data is generated, every attribute with a public name regardless of whether it is known, unknown or uncertain is output. However, entity instances and change point values are only output if they actually exist in the session, so if you don't have either of these you won't see them in your sample XML.

## Load the sample data

After you have generated your sample data you need to load it into the BI Publisher Template Builder in Word.

1. On the BI Publisher toolbar, select **Sample XML**:

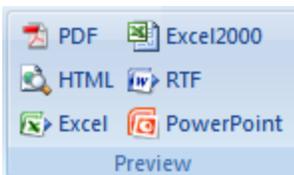


2. In the dialog box, select the XML file that contains your sample data (see above). You will be told when this data has been loaded successfully.

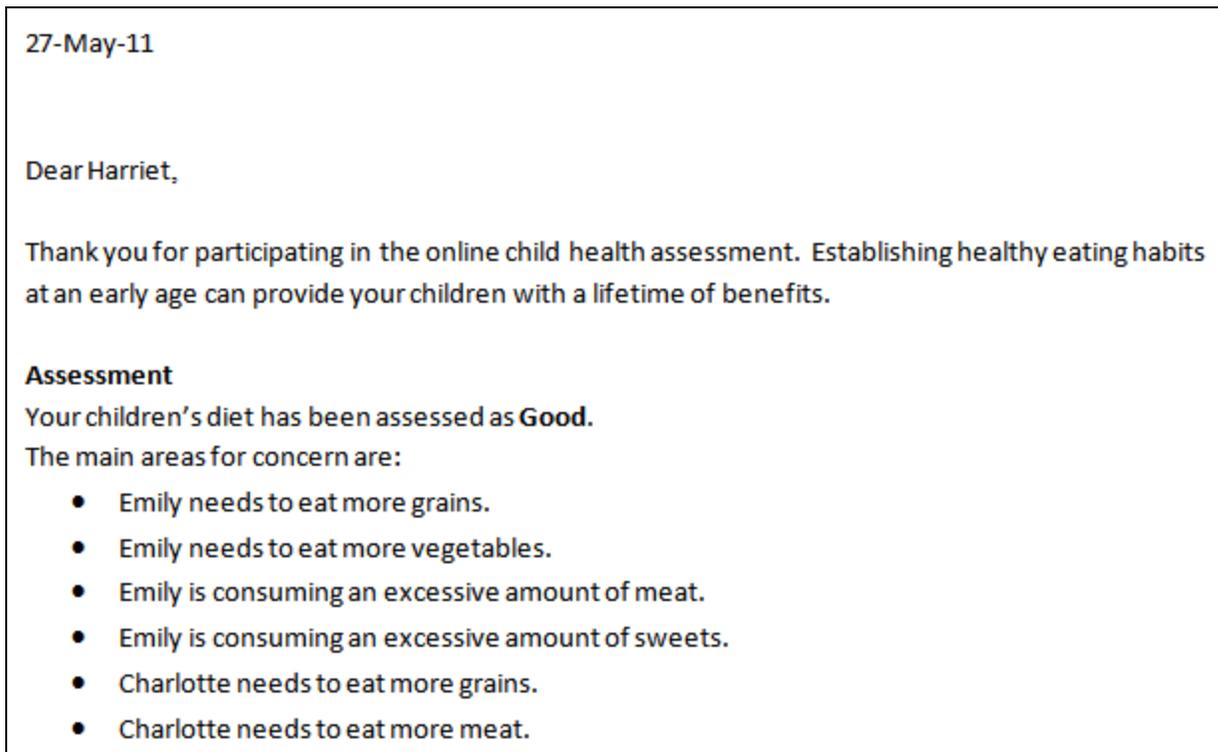
### Preview the output

After you have loaded your sample data you can preview the document in your choice of format.

1. Click on the appropriate **Preview** option:



2. Review the generated document:



3. If necessary, go back to your RTF template file to make any changes. Remember to [re-import your XML schema](#) if you want to continue editing your document using the BI Publisher **Fields** dialog.

If you are having problems with the display of any elements in your document, see the [Troubleshooting guide for using BI Publisher with Oracle Policy Modeling](#).

See also:

- the Template Builder for Word Help file (available under \Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word)
- the BI Publisher Users Guide (available under \Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\doc)
- [Localize interview document templates](#)

## Test an interview or screen flow

There are two ways in which you can test an interview or screen flow in Oracle Web Determinations:

- Using the debugger. This runs the rulebase using the server embedded in Oracle Policy Modeling.
- Using a deployed instance of Oracle Web Determinations. This runs the rulebase using an external server.

## What do you want to do?

[Use Oracle Web Determinations in the debugger](#)

[Use Oracle Web Determinations externally](#)

[Start an interview in Web Determinations](#)

[Investigate a goal in Web Determinations](#)

[Create entity instances in Web Determinations](#)

[Review the reason for a decision in Web Determinations](#)

[Review a document generated from the interview](#)

[Review the data collected in Web Determinations](#)

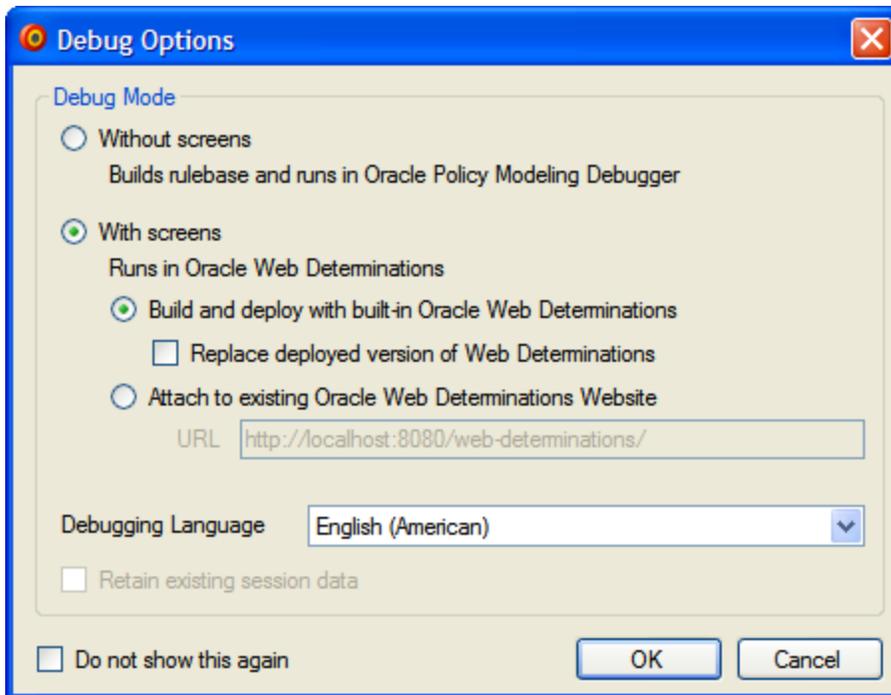
[Save an interview in Web Determinations](#)

[Know what to test for](#)

## Use Oracle Web Determinations in the debugger

When running Oracle Web Determinations from within Oracle Policy Modeling you have the advantage of being able to use the other features available in the debugger. To start testing your rules using Oracle Web Determinations embedded in Oracle Policy Modeling:

1. Select **Build | Build and Debug**.
2. In the **Debug Options** dialog box, select the option to debug **With screens**.



3. Select the appropriate deployment option. The options are:
  - \* **Build and deploy with built-in Oracle Web Determinations** - most commonly you would use this option. If you want to completely replace the previously deployed version of the project, click the checkbox to **Replace deployed version of Web Determinations**.
  - \* **Attach to existing Oracle Web Determinations Website** - use this option if you want to connect to an existing instance of Oracle Web Determinations for Java or .NET. Enter the **URL** of the deployed rulebase. See below for how to enable debugging when choosing this option.
4. Click **OK**. This will launch a session of Web Determinations. When you have finished testing your screens, stop the debugger by either closing the **Debug** view or by selecting **Stop Debugging** from the **Build** menu.

### Enable debugging in a deployed instance of Oracle Web Determinations

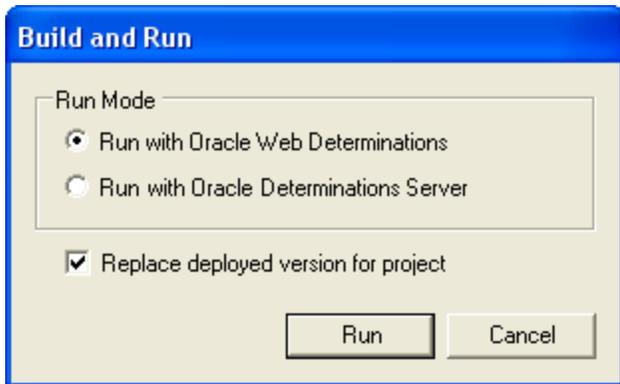
By default, a deployed instance of Oracle Web Determinations doesn't support debugging. To enable debugging for either Java or .NET, the following change needs to be made to the configuration file:

1. Open the **application.properties** file located in \Release\web-determinations\WEB-INF\classes\configuration.
2. In the Deployment Properties section, change the **enable.debugger** setting to **true**.
3. Save the application.properties file.
4. Restart IIS. (To do this go to Control Panel/ Administrative Services/ Internet Information Services. Select the local computer/Web Sites/Default Web Site. Right click and select **Stop**, and then right click and select **Start**.)

### Use Oracle Web Determinations externally

To start testing your rules using Oracle Web Determinations deployed to an external server:

1. Select **Build | Build and Run**.
2. In the **Build and Run** dialog box, select the run mode.



The options are:

- \* **Run with Oracle Web Determinations**
- \* **Run with Oracle Determinations Server**

If you want to completely replace the previously deployed version of the project (located in the Release folder), click the checkbox to **Replace deployed version for project**.

3. Click **Run**.

### Start an interview in Web Determinations

When you launch Web Determinations it opens to the [summary screen](#) which lists a set of goals that you can investigate. What you see on this screen will depend on the labels and goals that you have defined for the default summary screen in your screens file.

## ORACLE Web Determinations

Data Review

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

### Assessment Summary

Welcome to the **Business Assistance Grant Assessment Tool**.

This tool will help you to determine whether a business is eligible for the Business Assistance Grant.

- [Click here to determine if the business is eligible for a Business Assistance Grant](#)

(If no summary screen has been defined in Oracle Policy Modeling, the summary screen in Web Determinations will be blank.)

Click on any goal link to start investigating it.

### Open a saved investigation

To open a saved investigation:

1. Click on the **Load** link in the menu bar.

# ORACLE® Web Determinations

Data Review

Save | Save As | Load | Restart | Close

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

## Assessment Summary

Welcome to the **Business Assistance Grant Assessment Tool**.

This tool will help you to determine whether a business is eligible for the Business Assistance Grant.

- [Click here to determine if the business is eligible for a Business Assistance Grant](#)

2. This will open the **Load Case** screen which lists all of the existing saved cases.

# ORACLE® Web Determinations

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

## Load Case

*By ending this session all unsaved changes will be lost. Do you wish to continue?*

Saved cases available for user 'guest':

- [test case 1](#)
- [test case 2](#)

Cancel

3. Click on the case name to open the investigation.

Investigate a goal in Web Determinations

Question screens are used to collect information during an interview. For example:

# ORACLE Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 2

[Business Details](#) | [Financial Details](#) | [Date of Previous Grant](#) | [Innovative Field](#)

## Financial Details

*Please enter financial details for the business*

Business Outgoings

\*

Business Revenue

\*

Submit

### Questions

What you see on a question screen during your interview will depend on whether the attribute that needs an answer is defined on a question screen in your screens file in Oracle Policy Modeling (see [Create a question screen](#) for more information). If it is, the labels, format and behaviour of that attribute on the question screen in Web Determinations will be as defined on the question screen in that file. If the attribute does not exist on a question screen in the screens file then the attribute will be displayed on an Automatic question screen in Web Determinations.

You must answer questions in the format specified by the [Region setting for your rulebase](#), for example you must enter dates or currency values in the correct format.

On question screens, mandatory questions are identified by the icon \*. You must provide an answer to these questions before you can continue on to the next screen in the interview.

TIP: If you are using Web Determinations in the debugger, at any point in an interview you can switch to the **Data** view to see which attributes are known.

### Progress stages

At the top of the interview screen is a section which shows the screen/stage you are currently on (in bold) and gives a measure of 'how known' the goal is that you are investigating.

[Business Details](#) | [Financial Details](#) | [Date of Previous Grant](#) | [Innovative Field](#)

If question screens are grouped into sub-folders in the screens file, the subfolders will be used as the progress stages.

This feature only works when a [screen order](#) is defined. To turn off this feature, change the **show-progress-stages** setting to false in the **appearance.properties** file that is located in `\Release\web-determinations\WEB-INF\classes\configuration` for the project.

NOTE: This is not a navigation tool.

### Progress bar

Another option for the display of progress through an interview is a progress bar:



To turn on this feature, change the **show-progress-bar** setting to true in the **appearance.properties** file that is located in **\Release\web-determinations\WEB-INF\classes\configuration** for the project.

NOTE: The calculation of progress is based entirely on attributes, not on screens, which means that the progress bar works even if you are not using a [screen order](#) or even if you have automatic screens. It will not appear while executing a screen flow.

### Help text

If [help text](#) is available for a question, this can be accessed by clicking on the question text.

When you have answered the questions on a screen, click the **Submit** button to move to the next screen in the interview. Continue to work through the interview process until no more question screens are presented to you. The number of screens shown will depend on your answers to the questions.

### Conclusions

When Web Determinations reaches a conclusion for the chosen goal in the assessment, no more questions will be asked and you will be returned to the Summary screen where the conclusion is displayed.

## ORACLE® Web Determinations

[Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

*Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 2*

### Assessment Summary

Welcome to the **Business Assistance Grant Assessment Tool**.

This tool will help you to determine whether a business is eligible for the Business Assistance Grant.

- The business is eligible for a Business Assistance Grant. [\[Why?\]](#)

At this point you have several options:

- Investigate another goal (if there are any)
- [Review the reason for a decision](#)
- [View an interview document](#) (if there are any)
- [Review the data already collected](#)
- Clear the session and start again
- [Save the assessment \(or a copy of it\)](#)
- Close the assessment

### Create entity instances in Web Determinations

To create an entity instance in Web Determinations, you click on the **Add** button (eg Add New Instance, Add Child) when presented with the entity collection screen. (The entity collection screen is defined in Oracle Policy Modeling, including the name of the Add New Instance button, see [Define a screen for collecting entity instances](#) for more information. If you have not yet defined one, an automatic (default) entity collect screen will be displayed in Web Determinations.)

## Children In Your Care

Child name:

\*

Remove

Add Child 

Remove Selected Child

Submit

Fill in the required fields and use the **Add** button to create additional entity instances. Then click **Submit** to move to the next screen in the interview.

NOTE: In Web Determinations, the [entity completion status](#) is set automatically.

### Explain this further

The entity completion status is set as follows:

- the global entity is always set as complete
- a non-global entity is set as complete if all the screens that collect instances of the entity have been displayed.

For example, say a rulebase has an entity for 'the child' and 'the pet', and the following relationships are used to collect instances of these entities:

- 'the children' (from global to 'the child')
- 'the child's pets' (from 'the child' to 'the pet')

The screen that collects instances of 'the child' uses 'the children' as the relationship. Since the screen (and relationship) belongs to the global entity, there is only one such occurrence of this screen, so as soon as it is displayed, the entity 'the child' is set as complete.

The screen that collects instances of 'the pet' uses 'the child's pets' as the relationship. Since this belongs to 'the child', the screen can appear once for every child that the user has entered. If the user enters three children then the screen for collecting pets must be displayed for all three children before 'the pet' will be set as a complete entity.

### Review the reason for a decision in Web Determinations

When the interview has reached a conclusion, you can obtain a structured list of the reasons for that conclusion. This type of audit trail is known as a decision report.

The decision report is a "map" of the rules traversed in the rulebase in order to prove the current conclusion. Attributes that are proved by other attributes are displayed hierarchically, down to the level at which the user has entered data.

To see the decision report for your assessment, click on the **[Why ?]** link next to the interview goal on the Summary screen.

# ORACLE® Web Determinations

[Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

## Assessment Summary

Welcome to the **Business Assistance Grant Assessment Tool**.

This tool will help you to determine whether a business is eligible for the Business Assistance Grant.

- The business is eligible for a Business Assistance Grant. [\[Why?\]](#)



A decision report looks like this:

# ORACLE® Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

## The business is eligible for a Business Assistance Grant.

- ⊖ The business has between 4 and 12 employees.
  - [The number of people employed by the business is 5.](#)
- [The business is working in an innovative field.](#)
- ⊖ The business is struggling financially.
  - ⊖ The business expenses are greater than the business revenue.
    - [The business expenses is \\$15,000.00.](#)
    - [The business revenue is \\$10,000.00.](#)
- ⊖ The business is not excluded from receiving the grant due to previous eligibility.
  - ⊖ The business has not received the grant in the 12 months preceding the date of application.
    - [The business has not received the Business Assistance Grant previously.](#)

You can expand and collapse nodes in the report by clicking on the + and - signs.

Clicking on a base level attribute will open the screen on which that attribute was collected, allowing you to change the value for it.

Decision reports can be modified to prevent superfluous details from being shown. For more information, see [Hide information in a decision report](#).

Where an inferred attribute has several values over time these will be listed in the decision report:

# ORACLE Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Print](#)

Rulebase: Pension Locale: en-US User ID:

## The person's total entitlement for pension for the period is \$5,428.5

- [The start of the period is 10/10/08.](#)
- [The end of the period is 3/11/10.](#)
- [The person's daily entitlement for pension is {\\$0.00, \\$7.50 from 5/6/95, \\$10.50 from 7/1/06}.](#)
- [The person satisfies the age requirement.\({No, Yes from 5/6/95}\)](#)
- [The standard daily rate of benefit is {\\$5.00, \\$7.00 from 7/1/06}.](#)

Review a document generated from the interview

If there are any interview documents (eg assessment notices or personalized claim forms) you can access these from the Summary screen. Click on the appropriate link to generate the document.

# ORACLE Web Determinations

[Data Review](#)

[Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: HealthyEating Locale: en-GB User ID: guest Case ID: test case 4

## Welcome to the Healthy Eating Rulebase

This rulebase collects information about your children's eating habits.

- [Click here to view the Interview Summary Document](#)
- [Click here to view the Decision Letter](#)

If the interview document is a PDF, RTF or Excel file, you will be prompted to Open or Save the file. If the document is a HTML file, it will open directly in Web Determinations.

NOTES:

- a. If clicking the document link prompts you to save an XML file, the [Generate Xml Data checkbox for the document control](#) in the summary screen has been selected, and will need to be unselected in order to view the actual generated document.
- b. If your document is not generated, or if elements in your generated document do not appear, see the [Troubleshooting guide for using BI Publisher with Oracle Policy Modeling](#).

Review the data collected in Web Determinations

You can review the data already collected in an investigation by visiting the data review screen. To access this screen click on the **Data Review** link on the Summary screen. This screen provides a list of all the questions answered during an interview. Pre-seeded data will also be displayed.

An example of a data review screen is:

# ORACLE Web Determinations

Summary

Save | Save As | Load | Restart | Close

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

## Data Review

### Business Details

**Date of current application** 10/10/11

**Number of employees** 5

**Has the business received the Business Assistance Grant previously?** No

### Financial Details

**Business Outgoings** \$150,000.00

**Business Revenue** \$140,000.00

### Innovative Field

**Is the business working in an innovative field?** Yes

**Please provide details:** See website for details

**Is the business intending to work in an innovative field?** Yes

**Please provide details:** as above

On this screen you can click on the links provided to go directly to the relevant question screen. This allows you to change the information entered on those screens, and determine whether changes to that information affects system decisions.

If a screen order has been defined in the screens file in Oracle Policy Modeling, the data review screen in Web Determinations will list screens according to that order.

If no screen order has been defined in your screens file, the screens will be listed in a random order without reference to the order in which they have appeared in the interview. It is therefore recommended that you always define a screen order in your screens file. See [Define interview screen order](#) for more information.

The formatting of attribute values in Oracle Web Determinations, including date, number and currency values, is set based on the [Region](#) specified in the Project Properties for the rulebase. See the [Oracle Policy Automation Developer's Guide](#) for details on how to override this if required.

TIP: You can change the name of this screen. See [Change the title of the data review screen](#) for more information.

## Save an interview in Web Determinations

At any point during an interview, you may want to save it. Saved interviews can be [reloaded](#) at a later time, allowing you to continue the assessment or modify it.

To save a new interview, click the **Save** link on the Summary screen. To save an existing interview click on the **Save As** link on the Summary screen.

# ORACLE Web Determinations

Summary

Save | Save As | Load | Restart | Close

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

The **Save As** screen will be shown. This allows you to enter a name for the assessment and save the case.

## ORACLE Web Determinations

Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

### Save As

Save as case ID

\*

Save

Cancel

To save an interview that has previously been saved, on the Summary screen:

- click on the **Save** link to save the case, or
- click on the **Save As** link to save a copy of the case.

### Know what to test for

Using Web Determinations you can undertake an interview to ensure that your screens are effective and working as expected. You should make sure that you check that:

- the screens appear in the anticipated order
- any HTML tags are effective
- any substituted attributes are appearing where you had anticipated
- no headings are repeated
- any restricted inputs are working as expected
- question text makes sense
- commentary links work correctly
- the visual appearance (eg fonts, background colors, pictures, etc.) is as intended
- "Uncertain" has been enabled/disabled for each question as appropriate
- there are no spelling errors
- no screens are 'looping'

# Decision reports

Topics in "Decision reports"

- [Design a decision report](#)
- [Hide information in a decision report](#)
- [Add more information to a decision report](#)

## Design a decision report

A decision report is a report that can be viewed when the outcome of a goal is known, outlining the reasons for that decision. The value of every base level and intermediate attribute relevant to the final outcome is displayed in the decision report.

You may want to tailor your decision report to improve readability and to make sure the logic in the rules presents sensibly to a user. You may also choose to censor information for a particular audience.

The following steps should be undertaken when designing a decision report:

- Checking the decision reports contain enough information to explain the answer. This may require adding intermediate conditions and restructuring rules. See [Improve the wording of a rule](#) for more information.
- Checking the decision reports don't contain unnecessary information. Silent and invisible operators can be added to conditions in rules to selectively omit the inclusion of lower-level attributes in decision reports. See [Hide information in a decision report](#) for more information.
- Checking sentence construction and correcting parsing. Sometimes this can involve rewording attributes and even restructuring rules to reflect the reworded attribute. See [Change the text of an interview question or sentence](#) for more information.
- Checking that number variables are displaying as desired. By default, these will be shown as [formatted values](#). If you would like a number variable to display [unformatted](#) you need to select the checkbox "Unformatted" in the Attribute Editor for that variable.
- Reviewing decision reports to ensure they conform with the principles for writing rules. See [Rule principles for Oracle Policy Modeling](#) for more information.

## Hide information in a decision report

Decision reports can often be too verbose to provide a useful explanation of reasons for a decision to the user, particularly in these common areas:

- attributes which are used repeatedly throughout the rulebase (both base level and inferred)
- application-level rules, which only perform system functions which add no value to the user (eg the claimant is married -> the claimant marital status ="married")
- relationships, which are used repeatedly throughout the rulebase

You can trim decision reports with the use of the silent and invisible rule parameters, by preventing attributes and relationships from being displayed in the decision report, or hiding entire decision trees.

## What do you want to do?

[Hide all attributes in the decision report below a particular attribute](#)

[Hide a particular attribute in the decision report](#)

Cut off a decision report above a particular attribute

Hide a relationship in the decision report

Hide all attributes in the decision report below a particular attribute

The "silent" parameter is used to hide all attributes in the decision report below the attribute on which the parameter is used. You can make attributes silent at the rule level or globally.

To apply the silent parameter to an attribute at the rule level:

1. In your Word rules document place the cursor after the attribute text.
2. Click the **Silent Operator** button on the Oracle Policy Modeling toolbar or press **Alt+S**. (You can also apply the operator after an **and** or **or** operator, but never before the start of the attribute.)

For example:

**[b7] the claimant is eligible for child care benefit if**

[b15] the claimant satisfies the work/training/study test [silent] and

[b2] the claimant has at least one child in child care

The silent parameter can also be applied to a rule conditionally depending on the value of the attribute. To do this, just add the text "if true", "if false", "if certain", or "if uncertain" after the "silent". For example:

**[b7] the claimant is eligible for child care benefit if**

[b15] the claimant satisfies the work/training/study test [silent if true] and

[b2] the claimant has at least one child in child care

To apply the silent parameter globally:

1. In your properties file in Oracle Policy Modeling, double-click on the attribute in the Attribute view to open it in the **Attribute Editor**.
2. Select the **Decision Reports** tab.
3. Select the appropriate check boxes in the **Silent** section. For boolean attributes, you can mark the attribute as always silent, or silent only if the attribute is true, false or uncertain. Similarly, for non-boolean attributes, you can mark the attribute as always silent, or silent only if it is certain or uncertain.

Hide a particular attribute in the decision report

The "invisible" parameter is used to hide the attribute on which the parameter is used in the decision report. As for the silent parameter, you can make attributes invisible at the rule level or globally.

To apply the invisible parameter to an attribute at the rule level:

1. In your Word document place the cursor after the attribute text.
2. Click the **Invisible Operator** button on the Oracle Policy Modeling toolbar or press **Alt+I**. (You can also apply the operator after an **and** or **or** operator, but never before the start of the attribute.)

For example:

**[b18] the claimant is eligible for long service leave if**

[b21] the claimant qualifies for long service leave under section 45 [invisible]

The invisible parameter can also be applied to a rule conditionally depending on the value of the attribute. To do this, just add the text "if true", "if false", "if certain", or "if uncertain" after the "invisible". For example:

**[b18] the claimant is eligible for long service leave if**

[b21] the claimant qualifies for long service leave under section 45 [invisible if false]

To apply the invisible parameter globally:

1. In your properties file in Oracle Policy Modeling, double-click on the attribute in the Attribute view to open it in the **Attribute Editor**.
2. Select the **Decision Reports** tab.
3. Select the appropriate check boxes in the **Invisible** section. You can mark the attribute as always invisible, or invisible only if the boolean attribute is true, false or uncertain, or if the non-boolean attribute is certain or uncertain.

Cut off a decision report above a particular attribute

Silent and invisible rule parameters can be used together in rules so that both an intermediate attribute and any rules that prove that attribute are not included in the decision report. It may help to think of this as "chopping off" the decision report immediately above the attribute to which the pair of parameters is attached.

Hide a relationship in the decision report

The "invisible" parameter is used to hide a relationship, and the entity instances that are members of that relationship, in the decision report.

The "silent" parameter is only used with inferred relationships where it is used to show the target entities in the [membership rule](#) but hide the decision report.

To apply the invisible and/or silent parameters to a relationship:

1. In your properties file in Oracle Policy Modeling, double-click on the relationship in the Relationship view to open it in the **Relationship Editor**.
2. Select the **Invisible** and/or **Silent** checkboxes as appropriate.

See also:

- [Definition of relevant in decision reports](#)

Add more information to decision report

Decision reports may sometimes be too succinct to provide a user with a useful explanation of reasons for a decision. There are several ways you can add more information to decision reports.

**What do you want to do?**

[Add intermediate rules](#)

Remove existing silent and invisible operators

Substitute the value of an attribute for its text

Show the names of entity instances

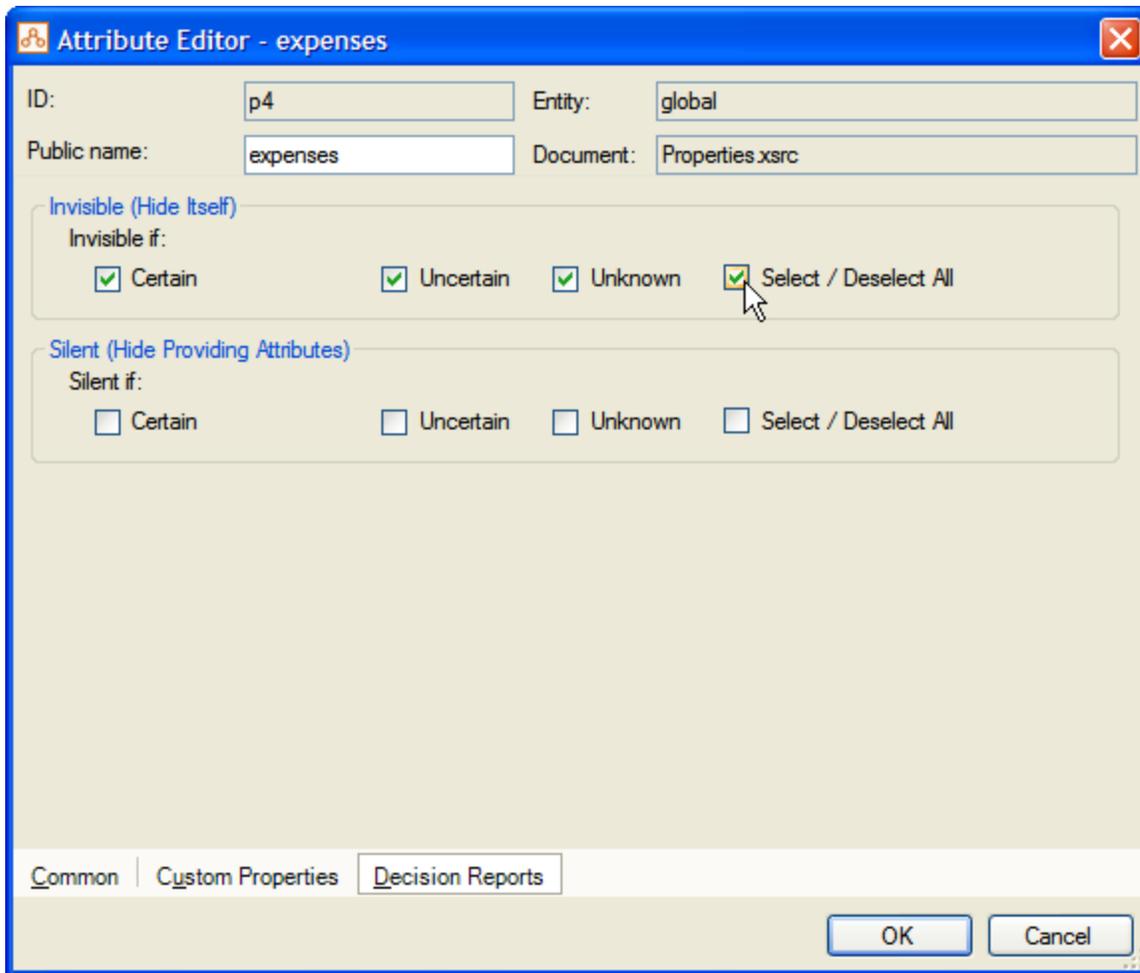
Add intermediate rules

You can add 'intermediate-level' rules to provide an additional layer of explanation between rules, which will help users see how the rule logic is operating. You can do this by [using variable comparisons to infer boolean attributes](#), or by [replacing grouping attributes with new attributes](#).

Remove existing silent and invisible operators

Sometimes when silent and/or invisible parameters have been used in the rules to hide attributes and decision trees, the resulting decision reports can be difficult to read and understand. In this case you may need to find where these parameters have been used and remove them. To do this:

1. In Oracle Policy Modeling, select **Build | Build and Debug**.
2. In the **Debug Options** dialog box, select the **Without screens** option, then click **OK**.
3. In the **Data** view, select the attribute you are interested in, right-click and select **Investigate**. TIP: It may be most effective to identify an intermediate goal proving the section of your rules that you wish to examine, and check this attribute for silent and invisible operators applied to its influencing attributes.
4. In the **Decision** view, select the option to **Show silent and invisible**. If this changes the decision view, then silent and/or invisible operators are being applied to the attributes in the decision view.
5. If this is the case, identify the attribute(s) that are affected, then open the properties file for the project and double-click the attribute to open the **Attribute Editor**.
6. Select the **Decision Reports** tab and see if there any silent and/or invisible parameters set. If so, remove them.



7. If the attribute does not have any silent and/or invisible parameters set on it in the properties file, then these operators must be operating at the rule level rather than globally. Open the rules document containing the rule, locate the rule and delete the silent and/or invisible operator.
8. Repeat steps 5 to 7 for all relevant attributes in your decision report, until you are satisfied that all appropriate attributes are being displayed correctly.

### Substitute the value of an attribute for its text

You can substitute the text of a variable with its actual value when it is used in another attribute in the rulebase. This substitution can make decision reports much more meaningful, for example:

the claimant's sibling lives in the claimant's sibling's country with the claimant

can become:

Charlene lives in Morocco with Anne

where "the claimant's sibling", "the claimant's sibling's country" and "the claimant" are all substituting variables.

For more information on how variable substitution operates, see [Substitute the actual value of a variable for its text](#).

## Show the names of entity instances

You can show the names of entity instances in decision reports to improve the readability of your decision report and make it clear which entity instance is being referred to. There are two places to do this:

1. In entity-level attributes. To substitute the name of the entity instance into entity-level attributes (eg "David's date of birth is 10/03/96" instead of "The child's date of birth is 10/03/96") and into relationship text (eg "David's school" instead of "the child's school"), you need to set up attribute substitution. See [Substitute the actual value of a variable for its text](#) for details of how to set this up.
2. In lists of entity instances. To show the name of each entity instance in the details of entity or relationship collect screens (eg "Sydney High", "Melbourne High", "Perth High" instead of generic labels like "#1", "#2", "#3" for the entity 'the school'), you need to have an identifying attribute for the entity. By default, an identifying attribute is automatically created when a new entity is created, so typically this will already work. (For details on how to set up an identifying attribute if you don't already have one, click [here](#).)

## ORACLE Web Determinations

[Summary](#) | [Data Review](#)

[Save](#) | [Save As](#)

Rulebase: InterviewServiceTest

### James is eligible for education expenses assistance.

- [James's annual salary is \\$50,000.00.](#)
- [David's date of birth is 10/03/96.](#)
- [David's school](#)
  - [Sydney High](#)
  - [Sydney High's type is SECONDARY.](#)
  - [Sydney High's number of students is 870.](#)
  - [Lee's date of birth is 21/05/97.](#)
- [Lee's school](#)
  - [Melbourne High](#)
  - [Melbourne High's type is SECONDARY.](#)
  - [Melbourne High's number of students is 912.](#)
  - [Brian's date of birth is 12/02/98.](#)
- [Brian's school](#)
  - [Perth High](#)
  - [Perth High's type is SECONDARY.](#)
  - [Perth High's number of students is 1,203.](#)

See also:

- [Definition of relevant in decision reports](#)

## Compiling and building

Topics in "Compiling and building"

- [Compile rules and correct errors](#)
- [Include extra files in the build](#)
- [Build a rulebase](#)
- [Create rules that can be shared with another project](#)
- [See the results of a recent build or deploy operation](#)
- [Define attribute names for use by external applications](#)
- [Check that a rule references the right data elements](#)
- [Fix a build error](#)
- [Exclude a rule file from the build](#)
- [Build the rulebase from the command line](#)

See also:

- [Check the rulebase against the data model](#)

### Compile rules and correct errors

The rulebase project is compiled to produce the required files to conduct an investigation.

After you have written your rules you need to compile them.

Compilation in Microsoft Word and Excel is triggered by the Compile  button on the Oracle Policy Modeling toolbar.

Clicking the Compile button starts the parse and validation process.

After your rules have been successfully compiled, you can then view your rules in Oracle Policy Modeling. There is a one-way direction for editing Oracle Policy Modeling documents. This means that you must alter your rule documents in Word or Excel and re-compile to make changes to your rule models. You cannot update your rules or attributes in Oracle Policy Modeling.

### What do you want to do?

[Correct rule errors](#)

[Understand what parsing means](#)

[Review the attribute parses](#)

[Identify the operative verb](#)

[Select an alternate parse](#)

[Delete unused attributes](#)

[Understand attribute IDs](#)

[Compile rules from within Oracle Policy Modeling](#)

### Correct rule errors

If there are errors in your Oracle Policy Modeling format, the compilation process will cease, and you will be prompted to correct those errors.

In the **Compile Errors** dialog, select the error message and then click the **Go To** button. The part of the rule that is causing the error will be highlighted in the rules document. Fix the error and then re-compile.

After your rules have been successfully compiled, any changes to attributes will be displayed and you will be informed that the process is complete. From this point, you can then view your rules in Oracle Policy Modeling.

### Understand what parsing means

All boolean attributes need to be parsed to produce their positive, negative, uncertain and question forms. The process of parsing is to identify the primary verb in the attribute and build these text forms around that verb.

For example, parsing the attribute "the dog bit the man" would generate:

the dog bit the man	positive form
the dog did not bite the man	negative form
did the dog bite the man?	question form
the dog might have bitten the man	uncertain form

Attributes can be entered in rules using the positive, negative or uncertain form. The parser can handle this and will still generate the other forms correctly.

NOTE: This description of parsing applies to the fully-featured parser (for example, English US) in Oracle Policy Modeling. If you have a project which uses a RLS (Rapid Language Support) parser, the sentence parses are generated using a generic statement defined in the configuration for that particular RLS parser. For more information on using an RLS parser, and changing individual sentence forms in such a project, see the Help available in the Rapid Language Support Tool.

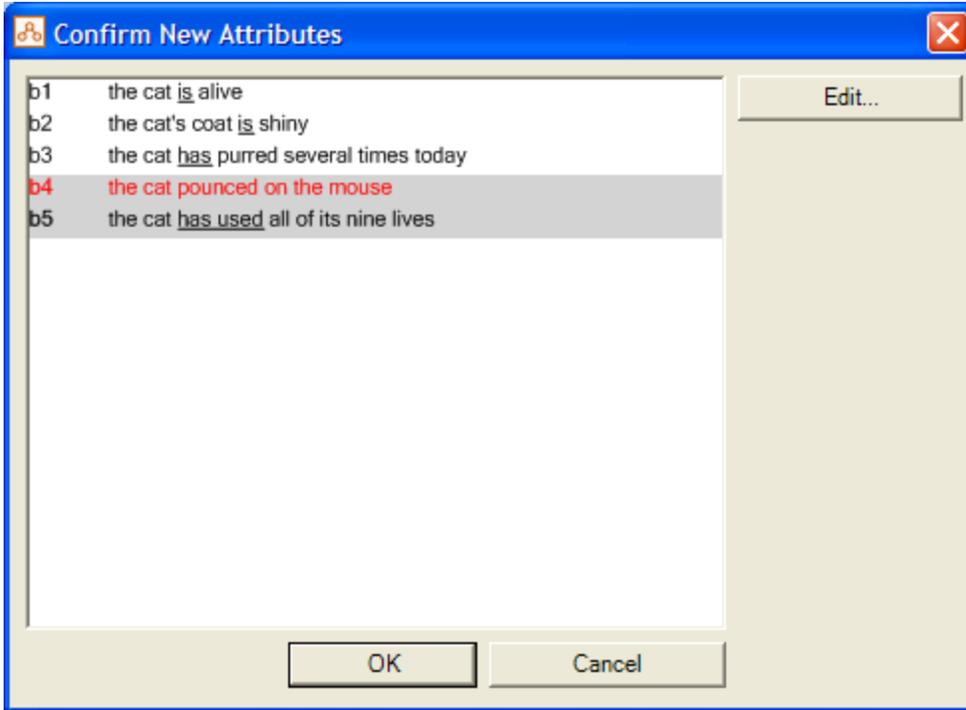
### Review the attribute parses

#### Review the attribute parses in a rules document

In Word and Excel, when you click the Compile button on the Oracle Policy Modeling toolbar, any new attributes will be automatically parsed.

You should review the attribute parses to see if any attributes have not been parsed correctly. You can do this using the **Confirm New Attributes** dialog which is displayed whenever you compile after adding new attributes.

In this dialog, the verb which is being used for the parse is underlined for each attribute in the list.



The table below describes what you should be looking for when reviewing this list of new attributes.

What to look for	What this means	What to do
Attributes highlighted with a gray background	The attribute contains more than one recognized verb (even if the word is not functioning as a verb in that particular attribute, as in the example above)	<p>If the underlined verb is the correct verb (ie the operative verb around which the sentence forms should be based) you can leave the parse as is.</p> <p>If the underlined verb is not the correct verb around which the parse should be based, you need to <a href="#">select an alternate parse</a>.</p> <p>If you are not sure if the underlined verb is the correct verb around which the parse should be based, you can view the sentences forms generated for each parse. In the <b>Confirm New Attributes</b> dialog, select the attribute and click the <b>Edit</b> button. Select the parse in the top box to view the sentence forms for that parse in the box below.</p>
Attributes shown in red (and also highlighted in gray)	The attribute does not contain a recognized verb and no sentence forms have been generated	<a href="#">Add the verb to the custom verbs list</a> for the project and then reparse the attribute
Attributes which contain compound verbs (ie verbs made up of several words) where	The verb has not been recognized by the parser as a compound verb, resulting in potentially incorrect sentence	<a href="#">Add the verb to the custom verbs list</a> for the project and then reparse the attribute

What to look for	What this means	What to do
the entire verb is not underlined	generation	

After you have confirmed your attributes, click **OK** in the **Confirm New Attributes** dialog box.

### Review the attribute parses in a properties file

In Oracle Policy Modeling, when you add a new boolean attribute to your properties file, you click the **Parse** button in the **Attribute Editor** to parse the attribute. The sentence forms for that parse will be shown in the box below.

#### Identify the operative verb

Sometimes, attributes only include one simple verb, in which case it is easy for the parser to identify the verb and generate the correct sentence forms. Often though, an attribute will contain more than one verb. For this reason, you need to be able to identify the operative verb in an attribute so that you can assess whether the attribute has been parsed correctly.

Some attributes contain two verbs but only one verb is operating as a verb in the attribute. It is easy to identify the operative verb if you consider how the attribute should be negated.

For example, the attribute "the car started to roll down the hill" contains two verbs, 'to start' and 'to roll'.

To negate this attribute you would place the "not" in front of the verb 'to start' (ie "the car did not start to roll down the hill") so 'to start' is the operative verb.

Similarly, "the people watched the boat go by" contains two verbs, 'to watch' and 'to go'. This attribute would be negated by placing the "not" in front of the verb 'to watch' (ie "the people did not watch the boat go by") so 'to watch' is the operative verb.

#### Select an alternate parse

If you want to change the parse for an attribute this should be done in the properties file in Oracle Policy Modeling to ensure that the change applies across all rule documents.

To select an alternate parse for an attribute:

1. Open the properties file for the project.
2. Double-click on the attribute in the Attribute view to open it in the **Attribute Editor**. (If the attribute does not already exist in the properties file, ie because it was added directly in the rules document/s, you will need to add it to the properties file. Right-click in the Attributes view and select **New Attribute**.)
3. Select the **Parse** button to open the **Select Parse** dialog. In the **Text** field the attribute is shown with the primary verb underlined.
4. Select an alternate parse from the list. This will display the sentence forms for that parse in the box below.
5. Click **OK**. The Attribute Editor will now show the new parse for the attribute.

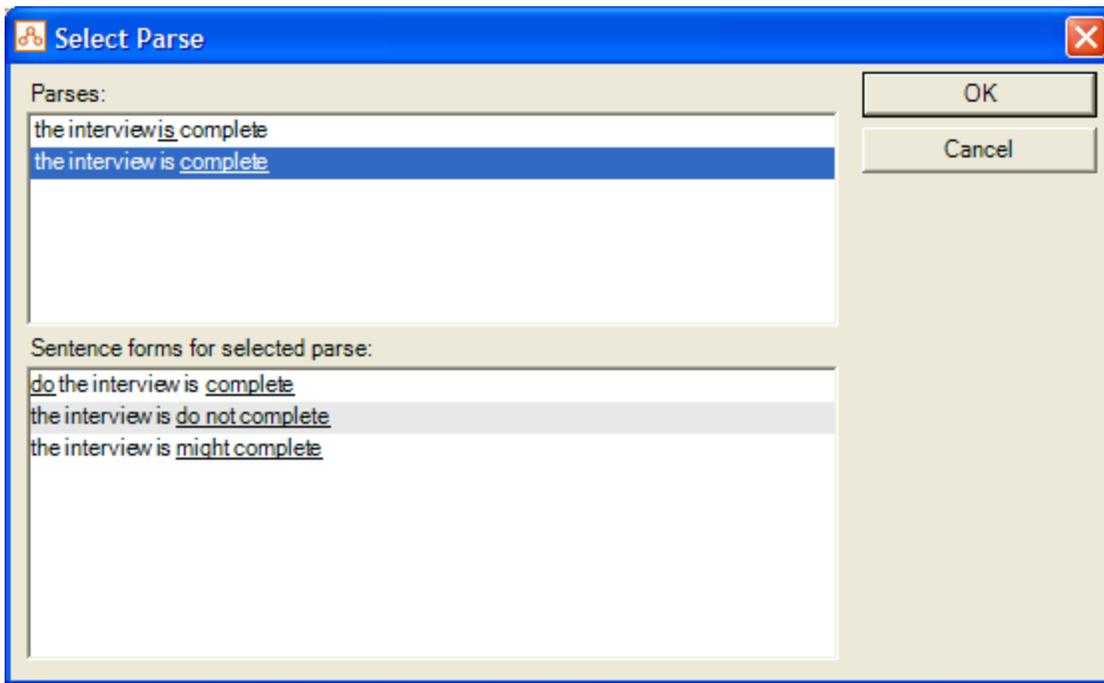
### Show me an example of an attribute which needs to be reparsed

The attribute "the interview is complete" has two possible parses as it contains two recognized verb forms, "is" and "complete". If "complete" is the primary verb the sentence forms are:

do the interview is complete

the interview is do not complete

the interview is might complete



These are not the correct sentence forms. Selecting the parse using "is" as the primary verb shows the correct forms:

is the interview complete

the interview is not complete

the interview might be complete



Select the attributes that you want to delete and click the **Yes** button.

## Understand attribute IDs

Oracle Policy Modeling automatically assigns an ID to each attribute as it is parsed during compiling. This is evident in the Oracle Policy Modeling mark-up (red text) which is inserted into your rule document in Word on compiling:

**[b7] the claimant satisfies the Financial Qualification if**

[b24] the claimant's weekly rent doubled is more than one half of the claimant's weekly net pay

[(p2\*2)>(p3/2)] (the claimant's weekly rent \* 2) > (the claimant's weekly net pay / 2)

Boolean attributes are named "bx" where x is a sequential integer (eg b1, b2, b3, etc). Non-boolean attributes are named "px" where x is a sequential integer (eg p1, p2, p3 etc). These IDs are used by Oracle Policy Modeling to notate rules.

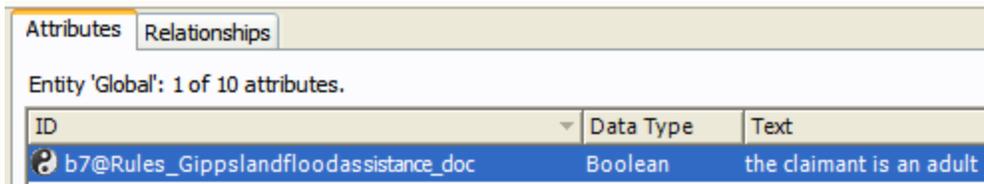
Because IDs are consecutively generated as b1, b2 etc on a per-document basis, Oracle Policy Modeling needs to distinguish between IDs from one rule document and another.

To do so, it automatically assigns a document ID to each entity and attribute ID, based on the document location and file name.

For example, b7 in the following document:



becomes:



ID	Data Type	Text
b7@Rules_Gippslandfloodassistance_doc	Boolean	the claimant is an adult

## Customize document IDs

Documents with long file and folder names can become somewhat unwieldy. To avoid this problem, Oracle Policy Modeling allows you to rename the document ID used in the automatic generation of attribute IDs. To do this:

1. Select the document in the Project Explorer, right-clicking it and selecting the **Properties** pop-up menu option.
2. Uncheck the **Base document ID on file name** check box and define a new, more comprehensible name for the document. (Names may only contain alphanumeric characters and the underscore ("\_"). Spaces are not permitted.)
3. Click **OK**.

The attribute IDs for that document will be updated with the new document ID:

Attributes Relationships		
Entity 'Global': 1 of 10 attributes.		
ID	Data Type	Text
b7@Rules_GFA_doc	Boolean	the claimant is an adult

## Compile rules from within Oracle Policy Modeling

To compile your rules from within Oracle Policy Modeling, right-click on the document name in the Project Explorer and select **Compile** from the pop-up menu. Oracle Policy Modeling will open the document in the appropriate program (Word or Excel) and automatically run the compile process.

It is also possible to compile all the documents in a project by selecting **Compile All** from the **Tools** menu. Word or Excel will open each document in the project one-by-one and automatically run the compile process for each one. NOTE: Compile All only compiles documents that have been modified (ie where the source document has a more recent time-stamp than the xgen file).

Once you have compiled your rules, you can build and [debug](#) them using the debugger. This allows you to explore your rules interactively.

## Include extra files in the build

Sometimes it is useful to include extra files in the compiled rulebase zip file, either because they need to be there or because it is convenient to do so because they directly relate to the compiled rules. Some examples are:

- Commentary text when running the rules in Web Determinations
- Compiled custom functions used by the rulebase
- Custom formatters for attribute values
- Custom inferencing listeners
- XML configuration of custom functions/formatters/inferencing listeners

Detailed information about these and other files that can be packaged in the compiled rulebase zip file is available in the [Oracle Policy Automation Developer's Guide](#).

Files such as these are automatically added to the compiled rulebase zip file if they appear in \include folder of the project. This is not created by default in a new project but can be easily created.

To include extra files in the rulebase zip file:

1. Use Windows Explorer to navigate to the folder containing a rulebase project.
2. Create a new folder called "include".
3. Copy any files or folders to be included into the include folder.

Folders copied into the include folder will retain their structure inside the zip file. Hidden files or folders (such as those created by source control tools like Subversion) are not included.

## Build a rulebase

In order to run your rules in the Oracle Determinations Engine you will need to build a set of files which represent the entire rulebase.

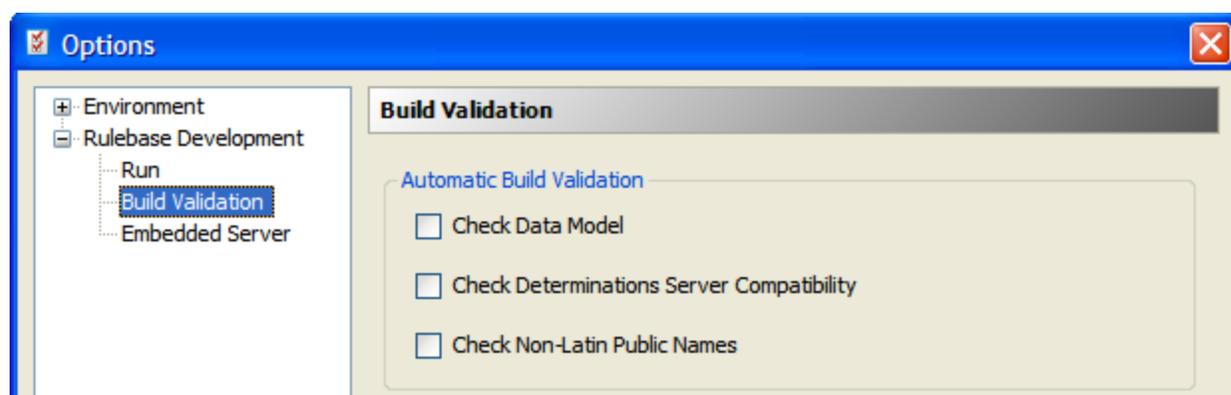
To build your rulebase, select **Build** from the Oracle Policy Modeling **Build** menu. A check will firstly be done to ensure that there are no source documents that need **compilation**. If there are you will be prompted to recompile these before continuing.

The build process will create the built rulebase files in the project output folder. The output folder is not visible through Oracle Policy Modeling but can be viewed in Windows Explorer under the project folder.

Single file rulebase deployment means that building a project in Oracle Policy Modeling automatically builds a <project>.zip file in the output folder. This package of all of the individual output components of a rulebase is the preferred method of deploying rulebases rather than as individual files.

NOTE: Any other files placed into the output folder will also automatically be included as part of this zip file, so unless the documentation explicitly directs you to, you should **not** put anything into the output folder. Also, whenever you do a build of a rulebase, the entire contents of the output directory are deleted. If the build is successful the only thing you will see in the output folder is the rulebase you have just built. If the build is not successful the output directory will be empty.

There are a few checks that can be automatically performed every time you build the rulebase. These checks are set up using the options under **Tools | Options | Rulebase Development | Build Validation**.



These options are:

- **Check Data Model** - Select this check box if you want a check to be performed of the data model when you build your rulebase. This check ensures that each base level attribute and each entity have corresponding public names.
- **Check Determinations Server Compatibility** - Select this check box if you want a check to be performed when you build your rulebase to ensure it is compatible with Oracle Determinations Server. This check ensures that all attributes, including goals, have corresponding public names.
- **Check Non-Latin Public Names** - Select this check box if you want a check to be performed when you build your rulebase for any attributes with public names containing non-Latin characters. (Non-Latin characters may cause problems when deploying the rulebase in IIS 5.1 and earlier.)

See also:

- [Build the rulebase from the command line](#)

## Create rules that can be shared with another project

Modules can be used to share aspects of a rulebase defined in one project with another project (or multiple projects). Information that can be exported through modules includes:

- the rules defined within a rulebase project, and
- entities, attributes and relationships defined in a project, including additional properties such as validations, defining attributes, as well as any custom property definitions.

### Define what can be used by other projects

Modules work on the principle of abstraction behind an interface, which means that the user can define which entities, attributes and relationships can be used by other rulebases but information on how those entities, attributes and relationships are used within that module are kept private. This is important because it enables modules to be altered and deployed independently of any other rulebase or module that might use them. For example, if a module author wishes to allow other rulebases to use the goal attribute "the person is eligible for benefit A", along with all the rules that prove it, they need only export that attribute and all the base level entities, attributes and relationships that participate in its proof. If the author subsequently wishes to change the way in which that attribute is proved, they may alter the rules, re-compile and redeploy that module without having to alter any of the rulebases or modules that depend on it (there are certain exceptions to this rule which are detailed below).

Generally, exporting an entity, attribute or relationship is simply a matter of adding it to the project's external data model by adding a public name to it. More specifically, what gets put into the module interface is determined by the following rules:

- For an attribute to be exported, it must have a public name and the entity to which it is attached must also be exported. If an attribute has a gender attribute defined for it, then the gender attribute must also be exported.
- For an entity to be exported, it must have a public name, its parent entity must also be exported and its containment relationship must have been exported. This has a flow-on effect such that if you had, for example, global -> parent -> child -> pet, then you cannot export the entity "pet" unless the "child" and "parent" entities are also exported (the global is always exported). Any specified identifying attribute must also be exported in order for an entity to be exported.
- For a relationship to be exported, both ends must have a public name, and both the source and target entities must also be exported. This applies to both containment and reference relationships.

Additionally, custom properties, as well as intrinsic properties, on attributes (such as validation, decision report, substitution parameters, "unformatted" flags for number attributes and "display seconds" flags for time of day and date time), entities (identifying attributes) and relationships (decision report parameters) are exported into the module.

Any translations provided in a rulebase for attribute text, validation text and error/warning event message text are also exported into the module.

### Build a module

To build a module, select **Build | Build Module** from the main menu in Oracle Policy Modeling.

The build process will create the built module file in the project output folder. (The output folder is not visible through Oracle Policy Modeling but can be viewed in Windows Explorer under the project folder.) Note that whenever you build a module, the entire contents of the output directory are deleted. If the build is successful, the only thing you will see in the output folder is the rulebase or module you have just built. If the build is not successful, the output directory will be empty.

During a module build, there are two classes of module specific warnings that may be displayed:

- Warnings caused by an entity, attribute or relationship that has a public name but cannot be exported for some other reason (see above).
- Warnings to indicate that a base level entity, attribute or relationship that participates in the proof of an exported inferred entity, attribute or relationship is not itself exported.

These warnings can be ignored, but may result in unintended behavior of the module.

For information on how to link a module to a rulebase, see [Include rules defined in a separate project](#).

## Deploy changes to a single module

As noted above, a module works on the principle of abstraction behind an interface to allow it to be modified independently of any other rulebase or module that might rely on it. Due to the need to maintain the integrity of the resulting rulebase at runtime there is, however, a limit to what changes can be made to a module without forcing any other rulebase or module that relies on it to also be recompiled.

Simply put, a module can be changed and redeployed without requiring any rulebase or module that relies on it to be recompiled provided the changes do not affect the modules interface. The modules interface consists of the following items:

- The ID, base text, data type and inferencing type (base or inferred) of any exported attributes.
- The ID, text and inferencing type (base or inferred) of any exported entities.
- The ID, reverse ID, text, reverse text, type, source entity, target entity and inferencing type (base or inferred) of any exported relationships.

Additionally, the data model of the interface itself must remain static which means that entities, attributes and/or relationships cannot be added or removed without affecting the interface.

Attempting to deploy a module with an altered interface will cause the engine to refuse to load any rulebase that depends on that module. In such a case, all the modules and rulebases that directly rely on that module must also be recompiled and redeployed. It is also possible that changes to a particular module could cause a loop or multiply proven attribute when that updated module is loaded and the entire rulebase is re-formed at runtime, and this would also result in the rulebase failing to load.

TIP: If you want to be able to update your module independently of the rulebases that rely on it, it is advisable to only export the base level attributes that are required to prove the particular inferred attributes that the parent rulebases use, rather than all the intermediate attributes as well.

## See the results of a recent build or deploy operation

When you build the rulebase, the progress and results of that build are logged in the Output window and Error List.

- The Output window displays the progress of various processes, such as compilation, importing and exporting. It also logs the firing of any event rules.
- The Error List view shows any build and model errors and warnings encountered during the build process. You can double-click on the error/warning in the error list to open the file in which the error has occurred or to access a report explaining the error.

When you deploy a rulebase to the embedded web server, the progress and results of that deploy are logged in the Embedded Web Server Output window. If there are any problems encountered during the deploy process, they will be logged in this window.

## Define attribute names for use by external applications

Oracle Policy Modeling automatically assigns an identifier to every attribute in the rulebase. These IDs are used by Oracle Policy Modeling to notate rules. The attribute ID is stored in the rulebase along with the attribute text and attribute type. By default, Boolean attributes are prefixed with the letter b and variable attributes are prefixed with the letter p.

Attribute IDs are regenerated every time a rule document is compiled and change values as attributes are re-worded. For this reason, public names, which are user-defined attribute IDs, should be used on a project because they ensure that the attribute IDs for important attributes are reliable and static and are therefore suitable for use by external applications. For example, the automatically assigned attribute ID "b1@Doc1" could be replaced with the more meaningful public name "date\_of\_birth".

Public name information is stored in the properties file for a project. After the rules have been written and compiled, public names should be assigned to all attributes that the application needs to access. This includes all base level attributes and all top level attributes.

Important intermediate attributes also need to have public names. For more information, see [Set public identifiers for entities and attributes](#).

## What do you want to do?

[Automatically generate public names for base and top level attributes](#)

[Replace auto-generated public names with meaningful ones](#)

[Check that all base level attributes have public names](#)

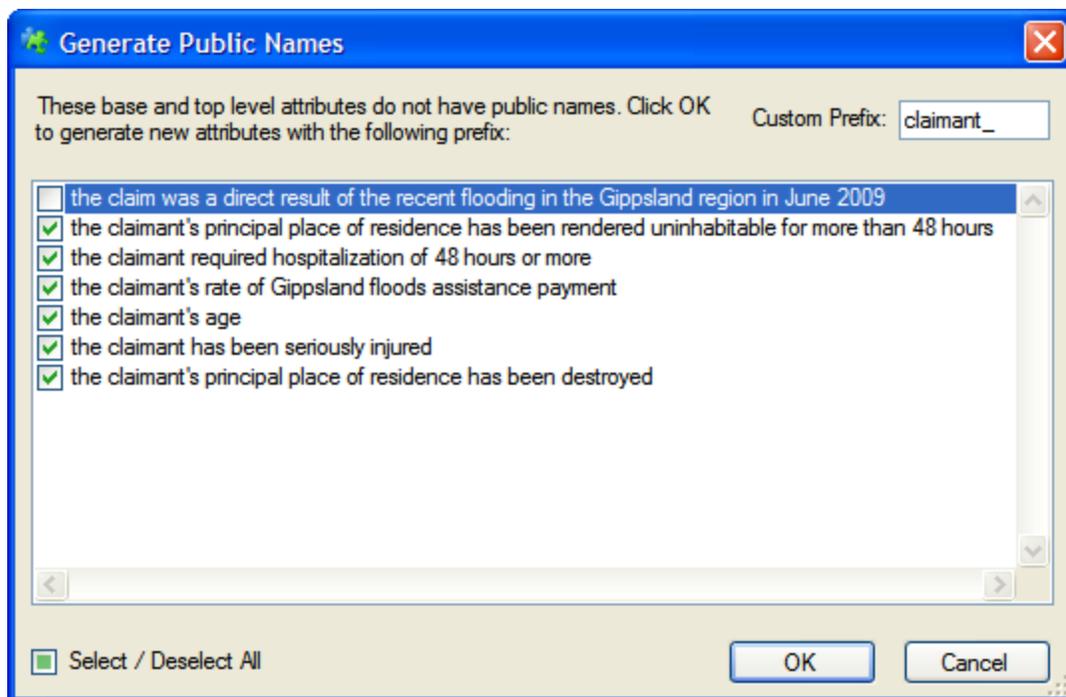
[Maintain public names](#)

## Automatically generate public names for base and top level attributes

Once you have created your rules in Word or Excel and compiled them, you need to generate public names for all base and top level attributes.

To automatically generate public names:

1. In Oracle Policy Modeling, open the properties file for your project.
2. Right-click in the Attributes window and select **Generate Public Names...** from the pop-up menu.
3. In the **Generate Public Names** dialog box you will be shown a list of base and top level attributes that do not have public names. Ensure that all appropriate attributes are selected.
4. Change the prefix, if necessary, to something more meaningful for the selected attributes (eg "claimant\_").



5. Click **OK**. The Attributes list in the properties file will now show each attribute and its public name (ID).

ID	Model ID	Data Type	Text
claimant_1	claimant_1	Boolean	the claimant has been seriously injured
claimant_2	claimant_2	Boolean	the claimant required hospitalization of 48 hours or more
claimant_3	claimant_3	Boolean	the claimant's principal place of residence has been destroyed
claimant_4	claimant_4	Boolean	the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours
123 claimant_5	claimant_5	Number	the claimant's age
claimant_6	claimant_6	Currency	the claimant's rate of Gippsland floods assistance payment

### Replace auto-generated public names with meaningful ones

After you have generated your public names, you may want to replace the auto-generated name with something more meaningful for each attribute.

To edit an attribute's public name:

1. In Oracle Policy Modeling, open the properties file for your project.
2. Right-click on the attribute in the Attributes list and select **Edit Attribute...** from the pop-up menu.
3. In the Attribute Editor change the name in the **Public Name** text box to something more meaningful (see below).

Attribute Editor - claimant\_5

ID: p1 Entity: global

Public Name: claimant\_age Document: Properties.xsrc

Data Type: Number  Unformatted

Text: the claimant's age

4. Click **OK**. The Attributes list in the properties file will be updated to reflect the new public name.

ID	Model ID	Data Type	Text
claimant_1	claimant_1	Boolean	the claimant has been seriously injured
claimant_2	claimant_2	Boolean	the claimant required hospitalization of 48 hours or more
claimant_3	claimant_3	Boolean	the claimant's principal place of residence has been destroyed
claimant_4	claimant_4	Boolean	the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours
123 claimant_age	claimant_age	Number	the claimant's age
claimant_6	claimant_6	Currency	the claimant's rate of Gippsland floods assistance payment

## Choose a meaningful public name

Your choice of public name may be influenced by a number of factors including:

- the need to identify the attribute with related attributes (for example you may want all attributes related to the claimant's address to begin with "claimant\_address\_");
- the need to identify what entity the attribute belongs to (this does not need to be the full entity name, for example, "hnm\_" would be a suitable public name prefix for attributes belonging to 'the household member' entity)
- the way in which the attribute will be used (for example you may want all attributes controlling screen behaviour to begin with "screen\_");
- any requirements imposed by an external data model, or the application in which the rulebase will be deployed.

Naming attributes clearly and consistently can make finding and sorting attributes much easier on large projects.

Note that public names cannot have spaces in them but underscores and dots can be used.

## Check that all base level attributes have public names

It is important that all base level attributes in a project have public names. Oracle Policy Modeling can optionally check that all base level attributes have public names every time you build the rulebase. To turn on this feature, go to **Tools | Options | Rulebase Development | Build Validation** and select the **Check Data Model** checkbox.

If base level attributes are detected without public names you will be informed that the Data Model Check has not been successful. You will then need to provide public names to these base level attributes before you can successfully build.

## Maintain public names

Over time rules naturally change, either due to legislative changes or business policy. There are three different scenarios that a rule developer may face regarding public name maintenance:

- *If the meaning of the attribute associated with the public name stays the same but the rule proving the attribute changes* - there are no changes required to the public name.
- *If the meaning of the attribute changes* - if this occurs, and the public name was specific enough then the public name attached to the attribute is probably out of date. A new public name which is associated with the attribute's meaning should be attached to the attribute. The old public name should be either moved to a corresponding new attribute or deleted.
- *If a new level of proof is needed for the base level attribute so that it no longer is a base level question* - sometimes a base level attribute will need to become an inferred attribute due to rule changes. Public names are typically only associated with base level questions, which are at the user input level of an interview. In this scenario follow these steps:
  1. Add a new proof to the current base level rule.
  2. If the public name can now be moved to a new base level attribute that is used to prove the newly inferred rule, move the public name.
  3. If the public name cannot be moved onto a new identical attribute then delete the public name.
  4. Add any new public names that are necessary for any new base level questions that have been created by the new rule proof.

## Check that a rule references the right data elements

Whenever you compile your rules you should check that the rules reference the correct data elements.

Things to check for when you compile a rule document:

- that any new attributes identified are exactly as you expect. For example, look in the New Attributes list for any attributes which have been created unintentionally because they are slightly different from pre-existing attributes.
- that functions have been parsed correctly. You can do this by checking the red mark-up text at the start of rule. Functions that have not been written correctly will be parsed as new attributes (so also check for this in the New Attributes list).
- that rules using entities and relationships correctly identify these components. Once again, check the red mark-up text to see that the rule has been parsed in the right way.

After you have compiled and built your rules you can use the Rule Browser to confirm that the right attributes are being referred to in the rule. For more information, see [Check the structure of a rule](#).

If you have had to write rules within a constrained data model, you should also use the Check Data Model build validation to ensure that the rules conform to that data model. For more information, see [Check the rulebase against an external data model](#).

See also

- [Exclude a rule file from the build](#)

## Fix a build error

When you build the rulebase any errors or warnings that are detected will be automatically logged in the Error List. Using this list you can see the type of error/warning (  build warning,  build error,  model warning,  model error) and a description of the error. Where relevant, the file that the problem has occurred in will also be listed.

There are two checks that are always performed when you build the rulebase:

- a check for multiply proven attributes, and
- a check for logical loops.

There are a couple of other optional checks that can be performed when you build the rulebase. See [Build a rulebase](#) for more information.

To fix a build error, select the error in the Error List and double-click it. If the error relates to:

- a multiply proven attribute, the **Rule Browser** will be shown. This view lists all the proving rules for the attribute and allows you to navigate easily to the rules within Word or Excel in order to fix them (either by [using rule fragments](#), or by making the attributes not multiply proven). An attribute which is proven by multiple rules, where these rules are not tagged as rule fragments, will not function correctly in the Engine because of the operation of the automatic alternate conclusion in every rule. That is, the closed logic of alternative conclusions will prevent both rules being traversed - the first traversed will close off the possibility of the other form operating.
- logical loops, the **Logical Loop Check report** will be shown. A Logical Loop Check report generates a list of any undefined self-referential rules, ie where an attribute is proved by itself and not defined as a rule loop, in the entire project. The Attribute Chain column in this report shows the chain of connections between attributes resulting in the self-reference. The Participating Rules column shows each of the rules involved in the loop. Generally, loops do not occur in single rules (eg x if x) but more commonly in highly nested layers of rules. Having self-referential rules has the result that those rules can never be fully proved. That is, the rule will repeatedly undergo a question search down the looping branch, cycling endlessly and never locating a base attribute within it. Use this report to identify the loop and then rectify it in your rules. If you are sure that the logic you need to model requires a loop in your rules, you may [define the rules as rule loops](#).

- c. an attribute needing a public name, the xgen file for that attribute will be shown. You will need to add a public name for that attribute in the associated properties file. To do this, select the attribute in the attributes list in the xgen file, right-click and choose **Create Public Name In** and your properties file from the pop-up menu.

To save a copy of the list of errors:

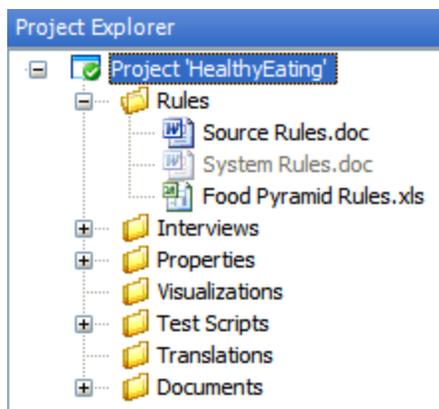
1. Right-click anywhere in the Error List window.
2. Select **Copy List**.
3. Open Microsoft Excel or Word (or another authoring tool) and paste the list.
4. Save the list.

### Exclude a rule file from the build

By default all documents contained in a project will be included in the build. In some circumstances it will not be appropriate to include a particular document in a build, either permanently or temporarily.

To exclude a document from the build follow these steps:

1. Select the document in the Project Explorer in Oracle Policy Modeling.
2. Right-click and select **Properties...** from the pop-up menu.
3. In the **Properties** dialog box, clear the **Include document in build** check box.
4. Click **OK**. You will now notice in the Project Explorer that the document icon is grayed out to indicate that the document is not included in the build.



### Build the rulebase from the command line

The Oracle Policy Modeling Command Line Compiler provides a means of building a rulebase from an Oracle Policy Modeling project using the command line. This allows the rulebase build process to be automated by including the command in a script.

The tool operates off an Oracle Policy Modeling project file. The project file settings and the documents included in the project are used to build the rulebase. The tool loads the project file, compiles the documents included in the project and builds the rulebase and other output files. The build process performed is the same as using the **Build | Build** menu item in Oracle Policy Modeling.

The build tool may also be used to compile and deploy a rulebase to the Determinations Server. The build and deploy process performed is the same as using the **Oracle Determinations Server** option under the **Build | Build and Run...** menu item in Oracle Policy Modeling.

By default, the tool performs validation on the rulebase model for rule loops and multiply-proven attributes. If the options detailed below are specified, additional validation can be performed. The build will fail if any validation errors are detected.

Projects created in old versions of Oracle Policy Modeling can be upgraded using the tool. Note that the project files will be copied to a backup location to ensure that you have the original version of the project to refer to if necessary. Release folders are not included in the upgrade process. The treatment of entities and their [containment relationships](#) in particular must be brought up to date from older project versions. See [Principles for the upgrading of entities and their containment relationships](#) for more information.

## Syntax

The Oracle Policy Modeling Command Line Compiler is executed from the command line using the following format:

```
buildtoolpath projectpath [build options] [validation options] [report options] [upgrade options] [help options]
```

## Parameters

Parameter	Description
buildtoolpath	The relative or absolute path of the Oracle.Policy.Modeling.CommandLineCompiler.exe file
projectpath	The relative or absolute path of the Oracle Policy Modeling project file to be built
<b>Build Options</b>	
-sb	Recompiles source documents before building the rulebase
-m	Builds the project as a module
-n <build number>	Sets the version number of the built rulebase/module
<b>Validation Options</b>	
-vd	Validates the rulebase model against the data model specified in the Oracle Policy Modeling project
-vds	Validates the rulebase for compatibility with Oracle Determinations Server, notably that all relevant attributes have public names
<b>Report Options</b>	
-cd	Analyzes a *.coverage file and produces a document-oriented report (.xml)
-cg	Analyzes a *.coverage file and produces a goal-oriented report(.xml)
<b>Upgrade Options</b>	
-upgrade	Checks if the project is compatible with the current version. If it is compatible, it proceeds to compilation. If it needs upgrade, the project is upgraded before being compiled. If it is not compatible (ie the project was created before v9.0), an error is displayed then it exits.
-remReadOnly	Removes write-protection for read-only files. This flag is only valid in the presence of the <i>-upgrade</i> flag. When set, write-protection will be removed for read-only project files.

Parameter	Description
	When not set, read-only project files will still be copied to the upgraded project directory but won't be processed.
<b>Help Options</b>	
-h	Prints the help message

### Example

For example, a command to build a project called Eligibility, recompile the source documents and then validate the rulebase model against the data model might look like this:

```
C:\Oracle.Policy.Modeling.CommandLineCompiler.exe C:\Eligibility\Eligibility.xprj -sb -vd
```

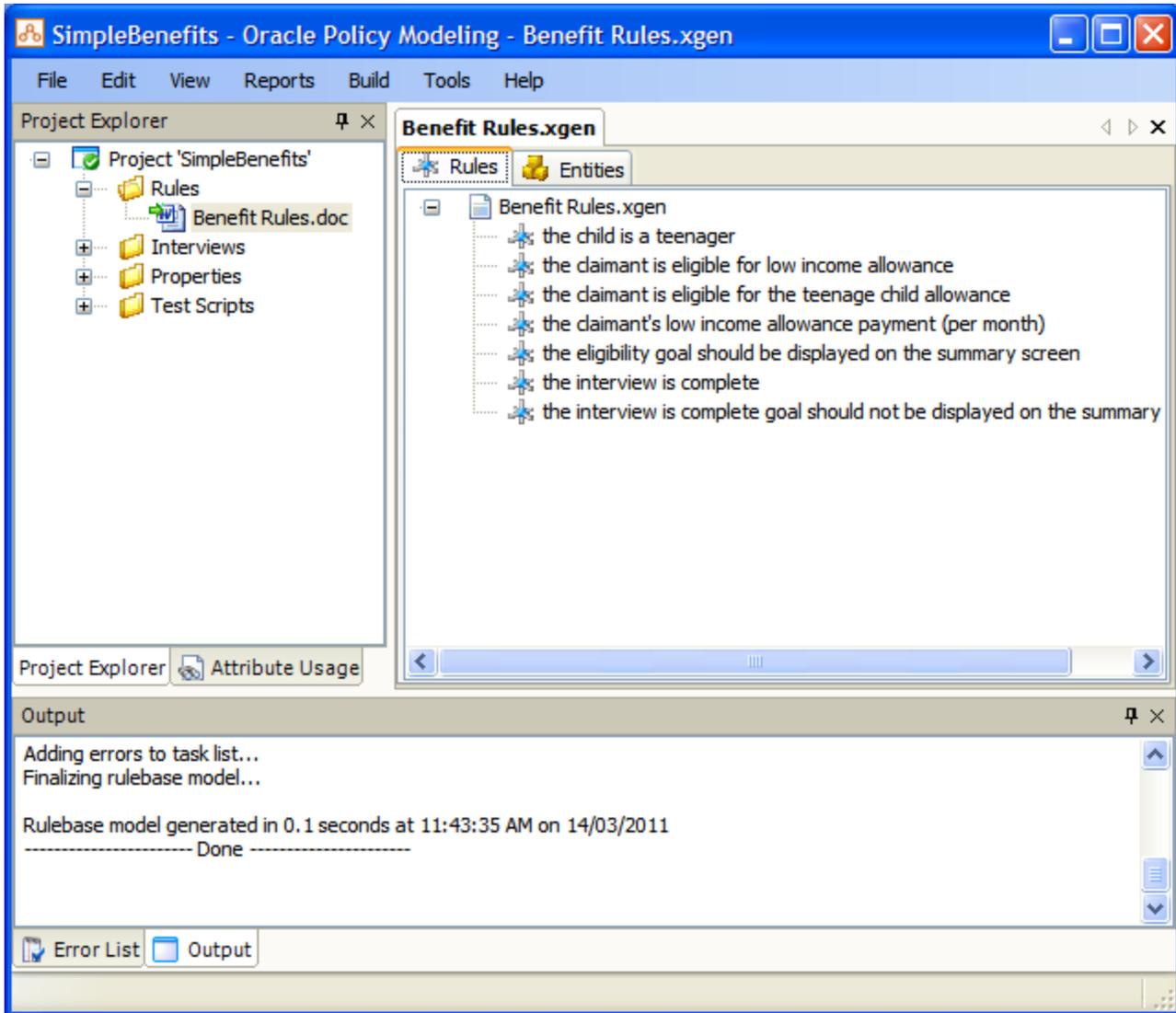
## Finding and reporting

Topics in "Finding and reporting"

- Find your way around the Oracle Policy Modeling user interface
- View list of entities and attributes
- Find the entity for an attribute
- Find rules that use an attribute or relationship
- Find dependent rules
- See the structure of a rule
- Check rule structure and dependencies
- Find input data needed to reach a conclusion
- Spell-check all interview screens
- Create, modify or delete a rulebase visualization

Find your way around the Oracle Policy Modeling user interface

The Oracle Policy Modeling interface, as shown below, has a [menu bar](#) and a multiple pane view below it.



By default, the left hand pane displays the **Project Explorer** which is the main management tool for working with project documents. The Project Explorer conveys important information about the project and the files within it such as:

- whether the project needs saving because changes have been made - indicated by an asterisk next to the project name
- whether a file needs to be compiled - indicated with a green arrow next to the file icon
- whether a file is excluded from the build - indicated with a red mark in the bottom right corner of the file icon
- whether a project (and files) are under source control - indicated by a padlock next to the file icon

You can open any of the files in the project from the Project Explorer. Double clicking the file name will open:

- Word and Excel files in their own applications
- Screens, Source, Test Script and Visual Browser files in the top right hand pane in Oracle Policy Modeling

The **Attribute Usage** view can also be shown in the left hand pane.

The right hand pane is used to display tabs for each of the various files, views and reports in the rulebase. The debugger is also shown in this pane.

The bottom pane displays the **Output Window** and **Error List**.

Each of these components can be closed and opened as needed.

## Oracle Policy Modeling menu bar and commands

The menu bar provides operations relating to the Oracle Policy Modeling project. The various menus and their commands are explained below.

### File menu

Command	Description
New Project...	Opens the New Project dialog which you use to <a href="#">create a new project</a>
Open Project...	Opens the Open Project dialog box where you can locate and open an existing Oracle Policy Modeling project
Close Project	Closes the project
Import Project...	Opens the Import Project dialog box where you specify the file and folder necessary to <a href="#">import a project</a>
Export...	Opens the Export Project dialog box where you select the destination and file name for the project interchange file which is used the <a href="#">export the project</a> to an external rules repository
Add	Accesses various options for adding files and folders to the project.
Save <Selected Item>	Saves the selected file
Save All	Saves all the files in the project
Source Control	Accesses various options for managing your project using source control
Project Properties...	Opens the Properties dialog box used to specify common properties, deploy properties, custom property definitions and regression tester properties
Project Statistics	Opens the Project Statistics dialog box. This displays <a href="#">statistics</a> for the build model and the files in the project.
Edit Verbs...	Prompts you to create a custom verbs file if you do not have one, and then opens the Verbs List dialog box where you can find, add, edit and delete verbs for your project. NOTE: The Edit Verbs menu is not available for projects which use a Rapid Language Support language parser.
Most	Lists the most recently opened projects in Oracle Policy Modeling. (The number of items displayed in this list in the

Command	Description
recently used projects list	File menu is specified in the <a href="#">Tools   Options</a> menu. The default setting is 4.) Clicking on a project name will opens that project in Oracle Policy Modeling.
Exit	Closes Oracle Policy Modeling

### Edit menu

Command	Description
Find Model Attribute...	Opens the Find Model Attribute dialog box enabling you to search the list of attributes for a particular <a href="#">attribute in the build model</a>
Find Document Attribute...	Opens the Find Document Attribute dialog box enabling you to search the list of attributes for a particular attribute contained within the Oracle Policy Modeling documents
Find Screen...	Opens the Find Screen dialog box enabling you to <a href="#">find a particular screen</a> in your project

### View menu

Command	Description
Project Explorer	Shows the Project Explorer view in the left hand pane. The Project Explorer is the main management tool for working with project documents. It displays all of the files in a project in a tree structure.
Attribute Usage	Shows the Attribute Usage view in the left hand pane. This enables you to <a href="#">find rules that use a particular attribute</a> .
Build Model	Opens the Build Model view in the top right hand pane. This view shows the <a href="#">full list of entities and attributes</a> from all the documents in a project.
Attribute Dependencies	Opens the Attribute Dependencies view in the top right hand pane. This is used to view the dependencies of a selected attribute. This is handy for <a href="#">seeing the structure of a rule</a> and for <a href="#">finding the input data needed to reach a particular conclusion</a> .
Data Model	Opens the Data Model view in the top right hand pane. This shows the <a href="#">rulebase data model</a> which is a definition of all data elements (base attributes) and their relationships to be maintained.
Error List	Opens the Error List in the bottom pane. This is a list of any <a href="#">errors</a> or warnings that are detected when you build the rulebase.
Output Window	Opens the Output Window in the bottom right hand pane. This displays the progress of various processes, such as compilation, importing and exporting. It also logs inferencing events during a debug session.
Embedded Web Server Output	Opens the Embedded Web Server Output Window in the center pane. This displays the progress of various processes when deploying to the embedded web server.

## Reports menu

Command	Description
Logical Loop Check	Generates a <a href="#">Logical Loop Check report</a> which is displayed in the right hand pane. This report contains a list of any self-referential rules, where an attribute is proved by itself, in the entire project.
Multiply Proven Attributes	Generates a <a href="#">Multiply Proven Attributes report</a> which is displayed in the right hand pane. This report shows attributes which are proven by more than one rule, and which are not marked as being rule fragments.
Top Level Attributes	Generates a <a href="#">Top Level Attributes report</a> which is displayed in the right hand pane. This report shows a list of attributes in the rulebase which are only proved by other attributes in the rulebase (ie they don't prove other attributes in the rulebase).
Base Level Attributes	Generates a <a href="#">Base Level Attributes report</a> which is displayed in the right hand pane. This report contains a list of attributes in the rulebase which are not proved by any other normal rules (forward chaining only rules).
Uncollected Attributes	Opens the Uncollected Attributes Report Options dialog where you can specify if you want to include base level attributes proven by shortcut rules in the report. (This report lists all base level attributes not collected on a screen.) The <a href="#">Uncollected Attributes report</a> is then displayed in the right hand pane.
Attributes Collected on Multiple Screens	Generates an Attributes Collected on Multiple Screens Report which is displayed in the right hand pane. This report lists any <a href="#">attributes that are collected on more than one question screen</a> .
Dependent Base Attributes	Generates a <a href="#">Dependent Base Attributes report</a> which is displayed in the right hand pane. This report shows a list of base level attributes which are dependent on a selected attribute.
Screens	Generates a Screens report which is displayed in the right hand pane. This report lists the contents of all the question and summary screens (including document links) defined in the project. This can be used for <a href="#">spell-checking the screens</a> .
Untranslated Text	Generates an <a href="#">Untranslated Text</a> report which is displayed in the right hand pane. This report lists all relevant rulebase elements for which a translation has not yet been supplied.
Custom Properties	Opens the Custom Properties Report Options dialog box where you can specify the property types to include in the report. The <a href="#">Custom Properties Report</a> then opens in the right hand pane.
Inferred Screen Attributes	Generates an <a href="#">Inferred Screen Attributes report</a> which is displayed in the right hand pane. This report shows a list of inferred attributes which appear on screens. Inferred attributes are attributes which are proved by other attributes in the rulebase.
View Test Script Specification	Opens the View Test Script Specification dialog box where you can select the test scripts that you want to view the details of. <a href="#">Test Specifications</a> for each selected script are then displayed in the right hand pane.
Run Multiple Test Scripts...	Opens the <a href="#">Run Multiple Test Scripts</a> dialog box where you can select which test scripts to run. The selected test scripts will then run and the resulting Test Report will be displayed.

Command	Description
Test Script Coverage	Analyzes the test scripts in the project and generates a Test Script Coverage report in the right hand pane. This report shows the <a href="#">coverage</a> for each condition in every goal in any test case.
Analyze Coverage Report...	Opens a dialog box where you can select a coverage file (that has been generated using the batch processor) to <a href="#">analyze</a> in Oracle Policy Modeling.

## Build menu

Command	Description
Build	Builds the rulebase project
Build and Debug	Opens the Debug Options dialog box where you specify whether to debug with screens (ie in Oracle Web Determinations) or without (ie using the debugger). NOTE: This dialog box is only shown if the "Show 'Debug Options' before starting debugger" option is selected in <a href="#">Project Properties</a> .
Stop Debugging	Ends the debugger session. (This menu is only enabled when debugging mode is on ie after Build and Debug has been selected.)
Build and Run	Opens the Build and Run dialog box where you have the option to run the rulebase with <a href="#">Oracle Web Determinations</a> or <a href="#">Oracle Determinations Server</a> .
Build Module	Builds the rulebase project as a module that contains the external data model (ie all entities and relationships, and any attributes with public names). Other projects can then <a href="#">link to this module</a> .
Generate Commentary Files...	Opens the Generate Commentary Files dialog box. This is used to specify the settings for the <a href="#">automatic generation of commentary files</a> .

## Tools menu

Command	Description
Clean Up Unused Attributes and Relationships...	Checks to see if there are any unused attributes or relationships in the project. (Unused attributes and relationships are those that have been defined in a properties file but that aren't used in a rule or screen.) If so, the Clean Up Unused Attributes and Relationships dialog box opens which lists all of the unused attributes and relationships so that you can select which ones you want to delete.
Compile All	<a href="#">Compiles</a> all the documents in the project that have been modified (ie where the source document has a more recent time-stamp than the xgen file)
Repair Attribute References...	Checks to see if there are any <a href="#">attribute references that need repairing</a> . If so, opens the Repair Attribute References dialog box so that you can select what to do with each broken reference.
Update Oracle Policy Modeling	Opens the Template Update Wizard that allows you to do a bulk <a href="#">update of all Oracle Policy Modeling templates</a>

Command	Description
Templates...	
Options...	Opens the Options dialog box. Here you can configure various options for the Oracle Policy Modeling environment and rulebase development.

## Help menu

Command	Description
Oracle Policy Modeling User's Guide	Opens the Oracle Policy Modeling User's Guide in a new window
Function Reference	Opens the <a href="#">Function Reference</a> in the language of the rule project
About Oracle Policy Modeling	Displays the product version number, copyright and patent details

## View list of entities and attributes

To view a list of all the entities and attributes from all the documents in a project, you use the build model view.

## What do you want to do?

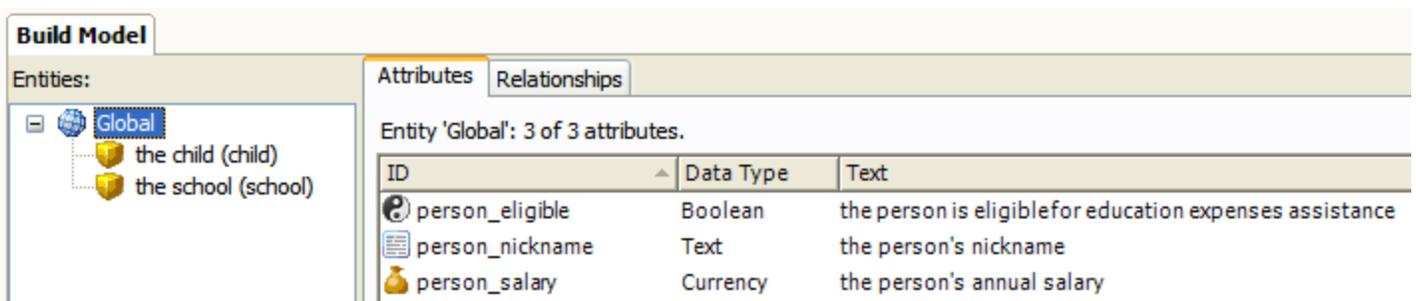
[View the entities and attributes in the build model](#)

[Find an attribute in the build model](#)

[Find where an attribute is used in the rulebase](#)

View the entities and attributes in the build model

1. To open the build model view, go to **View | Build Model**.



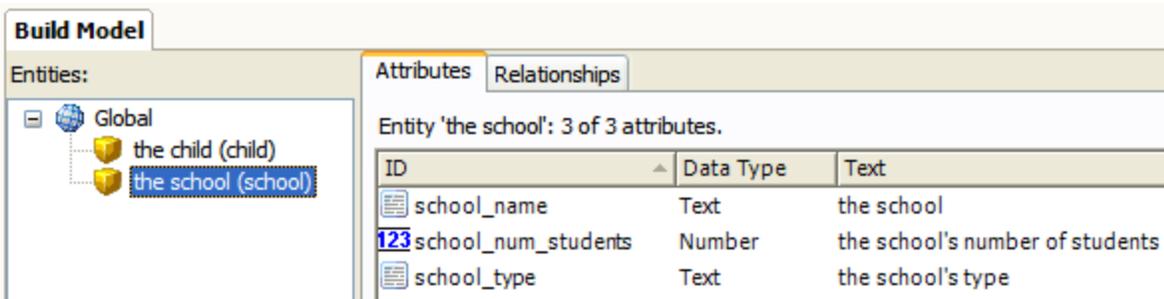
The screenshot shows the 'Build Model' interface. On the left, under 'Entities:', there is a tree view with a 'Global' entity containing two sub-entities: 'the child (child)' and 'the school (school)'. On the right, the 'Attributes' tab is active, showing 'Entity 'Global': 3 of 3 attributes.' Below this is a table:

ID	Data Type	Text
person_eligible	Boolean	the person is eligible for education expenses assistance
person_nickname	Text	the person's nickname
person_salary	Currency	the person's annual salary

2. The left hand pane in the build model shows the entities in the rulebase.



- To view the attributes that belong to a particular entity, select the entity in the left hand pane. The right hand pane will show the attributes (ID, data type and text) for the selected entity.



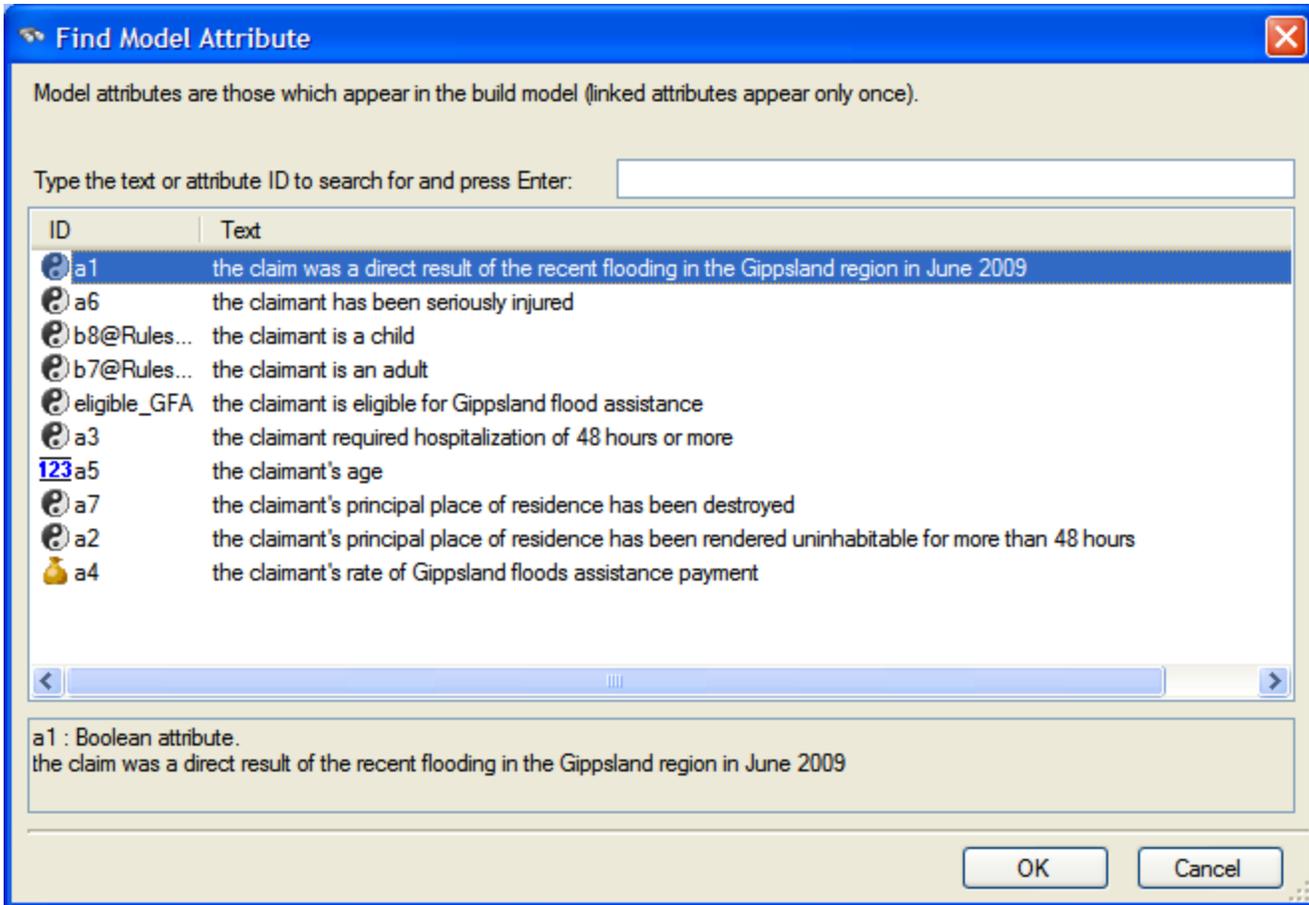
(Attributes which do not operate at the entity level are Global.)

You can also view the relationships for a particular entity by selecting the entity in the left hand pane, and then selecting the **Relationships** tab in the right hand pane. The relationships (text, target, type and reverse text) for the selected entity will be shown.



Find an attribute in the build model

The fastest way to find attributes is to use the **Find Model Attribute** search. To open the Find Model Attribute search, go to **Edit | Find Model Attribute...**

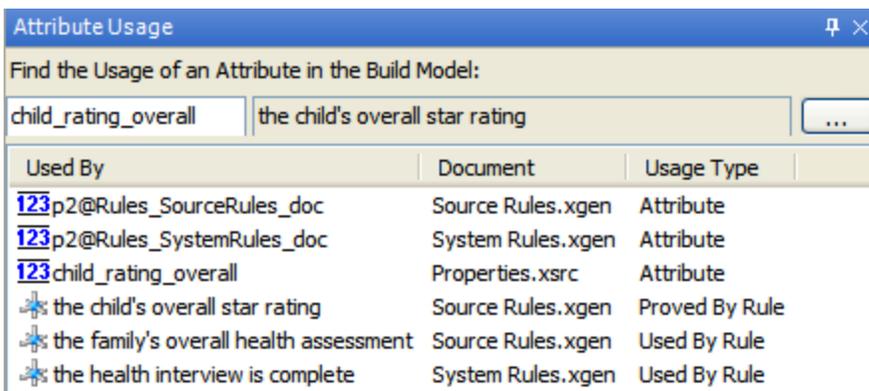


All the model attributes in the rulebase are listed. To narrow the list down, enter the text or attribute ID you want to search for in the text field provided. Only those attributes that match the search criteria will be shown.

Find where an attribute is used in the rulebase

To find where attributes are used in the rulebase, right-click on the attribute in the build model and select **Find Attribute Usage**.

The **Attribute Usage** view will open displaying all rule documents, source files, properties files, screens and flows on which the attribute appears:



## Find the entity for an attribute

After you have defined an entity, every attribute which contains the entity text will attach to that entity. Attributes which do not contain entity text are global.

For example, assume the attributes in the following table are part of a rulebase where "the household member" has been defined as an entity:

Attribute text	Entity level	Explanation
the household member is male	the household member	contains "the household member"
a household member is eligible	global	"a household member" does not match "the household member"
the former household member has left	global	"former" interrupts the attribute text
the household member's annual income	the household member	adding extra letters or characters on the left or right hand side is ok
the date of birth of the household member	the household member	entity text may appear anywhere in the attribute text

Both boolean and non-boolean attributes can be defined to belong to an entity in this way.

## Check attribute entity levels

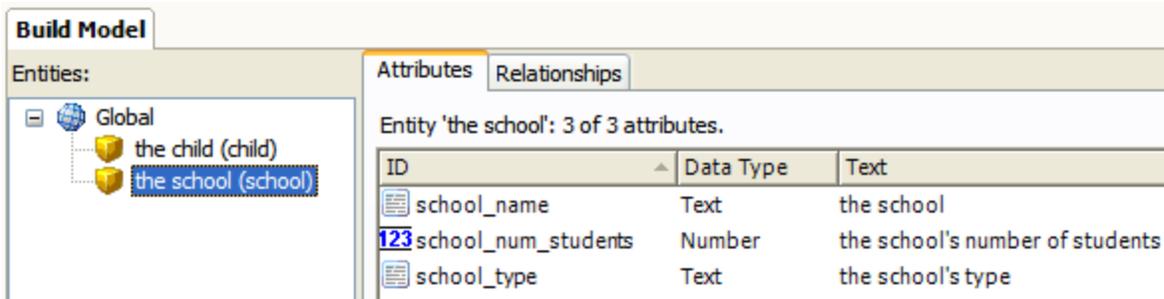
Once you have compiled your rules, you can check entity levels in the build model in Oracle Policy Modeling. To open the build model view, go to **View | Build Model**.

Attributes which are not within the scope of an entity are placed in the Global level. The list of global attributes are displayed in the right-hand pane:

The screenshot shows the 'Build Model' window with the 'Attributes' tab selected. On the left, the 'Entities' pane shows a 'Global' entity containing 'the child (child)' and 'the school (school)'. The main area displays 'Entity 'Global': 3 of 3 attributes.' with a table listing the attributes:

ID	Data Type	Text
person_eligible	Boolean	the person is eligible for education expenses assistance
person_nickname	Text	the person's nickname
person_salary	Currency	the person's annual salary

To view a list of entity-level attributes, click on the entity name. The list of entity-level attributes will be displayed in the right-hand pane:



## Why attribute scope is important

Once you define an entity, you cannot use attributes which belong to that entity in rules which operate outside the context of that entity.

For example, the following rule would be invalid (assume an entity "the child" has been defined):

**the claimant is eligible for transport assistance if**

the child travels a long distance to get to school

This is because we don't know which instance of the child (eg Max, Kat, Sarah) should be used in this rule.

## Find rules that use an attribute or relationship

### Find rules that use an attribute

To find the rules that use a particular attribute you can use the Attribute Usage view. To open the Attribute Usage view:

1. Go to **View | Attribute Usage**. (TIP: If you are using the debugger you can access the Attribute Usage view by right-clicking an attribute in the **Data** view and selecting **Show Attribute Usage**.)
2. Click on the browse button in the **Attribute Usage** view.
3. In the **Attribute Selector** dialog box, search for the attribute you want to find in the build model. (TIP: If your rulebase is very large, searching for an attribute in the Attribute Selector will be quicker if you turn off the **Filter search results on each keystroke** option under **File | Project Properties | Common Properties | General**.)
4. Once you have selected the attribute, click **OK**.  
The Attribute Usage view will display the selected attribute and show what it is used by, which document it is used in and the type of usage. Rules that use a particular attribute are shown by the icon  and have the type of usage 'Used by rule'.
5. To view the rule in the rules document, right-click and select **View in Word** or **View in Excel**.

Alternatively, you can use the Rule Browser to see how the attribute is used in rules. To do this:

1. Select **View | Build Model** to open the build model view.
2. In the **Attributes** pane, right-click the attribute and select **Rule Browser**.  
The Rule Browser will open to show any rules that prove the attribute, as well as any rules that are used by the attribute.
3. To view the rule in the rules document, click the **edit** link next to the name of the xgen file for the rule.

### Find rules that use a relationship

You can use the Rule Browser to see how a relationship is used in rules. To do this:

1. Select **View | Build Model** to open the build model view.
2. In the **Relationship** pane, right-click the relationship and select **Rule Browser**.  
The Rule Browser will open to show any rules that prove the relationship, as well as any rules that are used by the relationship.
3. To view the rule in the rules document, click the **edit** link next to the name of the xgen file for the rule.

See also:

- [Check rule structure and dependencies](#)

## Find dependent rules

To find dependent rules you can use the Rule Browser or a rulebase visualization.

### Find dependent rules using the Rule Browser

To launch the Rule Browser, right-click on a rule document in the Project Explorer and select **Open Rule Browser**. You can also right-click on an attribute in the Build Model view (**View | Build Model**) and select **Rule Browser**.

In the Rule Browser linked attributes are displayed as hyperlinks, allowing you to click on any attribute to see what rules the attribute is proved by and what rules the attribute is used by.

### Find dependent rules using a rulebase visualization

Using a rulebase visualization you can generate diagrams of rule structures to see how the attributes influence one another. For more information, see [Create a rulebase visualization](#).

Once you have generated your rule structure you can click on any attribute in the tree and view the rule text in the right hand pane.

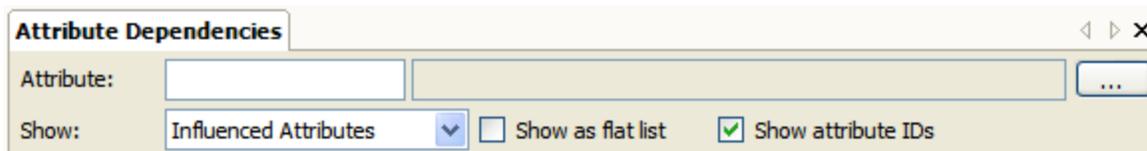
See also:

- [Check rule structure and dependencies](#)

## See the structure of a rule

To see the structure of a rule you can use the Attribute Dependencies view. This view shows the dependencies of a selected attribute. This is a useful tool for checking whether or not intermediate attributes are only proved by the attributes you expect to be proving them.

To open the Attribute Dependencies view, select **View | Attribute Dependencies**. In this view, use the browse button to select the attribute whose dependencies you want to view.

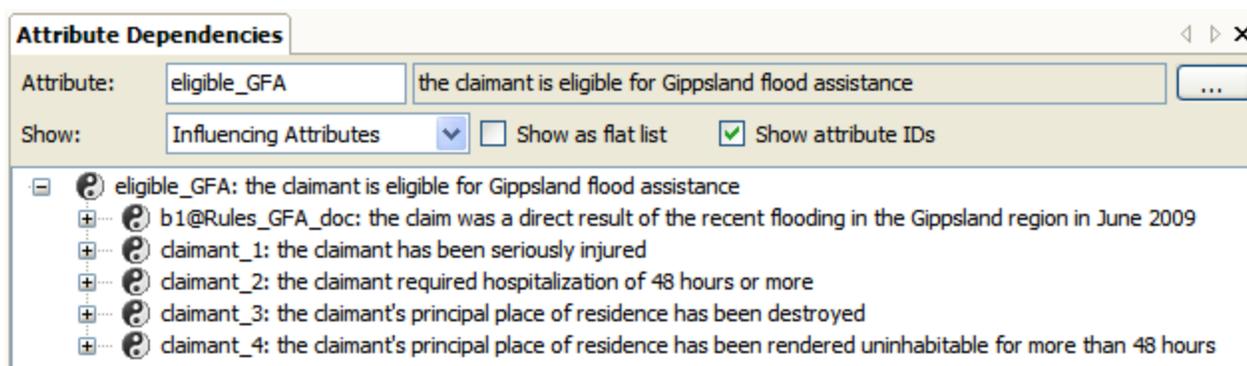


**TIP:** If your rulebase is very large, searching for an attribute in the Attribute Selector will be quicker if you turn off the **Filter search results on each keystroke** option under **File | Project Properties | Common Properties | General**.

Once you have selected the attribute, use the following options to customize the view.

- **Influencing attributes** option - shows only those attributes which influence the selected attribute.
- **Influenced attributes** option - shows only those attributes which are influenced by the selected attribute.
- **Show as a flat list** option - shows the attribute dependencies in a list view with attributes listed by ID, level and text. For influencing attributes, the attributes will be either base or intermediate. For influenced attributes, the attributes will be either top or intermediate. NOTE: Intermediate attributes are only shown if the **Show intermediates** check box is selected.  
If the **Show as a flat list** option is not selected, the attribute dependencies will be shown in a tree view which can be expanded and collapsed.
- **Show attribute IDs** check box - if selected, shows attribute IDs in the tree view. This check box is only enabled if the **Show as a flat list** option is not selected above.
- **Show intermediates** check box - if selected, includes intermediate attributes in the list view. This check box is only enabled if the **Show as a flat list** option is selected above.

As you change these options, the view will update in the pane below.



You can also see the structure of your rules using rulebase visualizations to see how the attributes influence one another. For more information, see [Create, modify or delete a rulebase visualization](#). Once you have created a rulebase visualization you can add dependencies to it by selecting **Generate Influencing Rules** when generating the rule structure.

## Check rule structure and dependencies

There are several reports you can generate in Oracle Policy Modeling to check the dependencies between rules. To check the structure of rules, you use the Rule Browser.

### What do you want to do?

[Check connections between rules](#)

[Check the structure of a rule](#)

#### Check connections between rules

Using the top and base level attribute reports in Oracle Policy Modeling you can check if there are any intermediate attributes which have unintentionally become top or base level attributes because they have not been correctly constructed to fit into the rule

hierarchy.

A top level attributes report shows a list of attributes in the rulebase which are only proved by other attributes in the rulebase (ie they don't prove other attributes in the rulebase).

To run a top level attributes report, select **Reports | Top Level Attributes**.

A base level attributes report generates a list of attributes in the rulebase which are not proved by any other normal rules (forward chaining only rules). These are the attributes which will be presented to users as questions during interviews (if running the rules interactively), and which will be the basis for all decisions made with the rulebase.

Generating the base level attributes report for your rules assists you in reviewing all of these attributes and determining whether or not they are at an appropriate level of granularity.

To run a base level attributes report, select **Reports | Base Level Attributes**.

### **Check structural connectivity with the base level attributes report**

An additional use of the base level attributes report is to determine whether any structural attributes have not been connected to base attributes properly. Structural attributes are those which refer to structural elements of your rules, such as "Section 1 is satisfied", "Paragraph 1(a) is satisfied", and which are typically generated automatically by Oracle Policy Modeling.

A common error in rule formatting is to write similar but not exactly the same structural attributes, creating duplicate attributes. Whilst you intend for these attributes to be identical, their textual difference means that they are added as separate attributes to your model. The consequence of this is that these attributes become accidental base level or top level attributes.

When you have completed work on a section of rules, you should generate the base level attributes report and review the list to ensure that attributes have not unintentionally been duplicated through the use of inconsistent text forms.

### **Check connections between rules using the dependent base level attributes report**

The dependent base attributes report generates a list of base level attributes which go towards proving a particular inferred attribute. This report can be very useful when working with large rule models.

To generate the report, select an attribute in the Build Model, then select **Reports | Dependent Base Attributes** from the main menu.

### **Check the structure of a rule**

The Rule Browser is a helpful way to understand the links between rules across your rule documents.

To launch the Rule Browser, right-click on a rule document in the Project Explorer and select **Open Rule Browser**. You can also right-click on an attribute in the Build Model view (**View | Build Model**) and select **Rule Browser**.

In the Rule Browser linked attributes are displayed as hyperlinks, allowing you to jump from rule to rule to check the rule structure.

The **Attributes** drop down list allows you to specify the attribute ID format displayed:

- **Build Model** uses public names and fully qualified attribute IDs (which include the document name in which the attribute is defined);
- **Document Model** uses the attribute IDs allocated within individual rule documents when they are compiled;
- **None** omits all attribute IDs and displays the attribute text only.

## Rule Browser

Attributes: Build Model

Back

Forward

Browse...

### there are improvements that the customer could make to the children's diet

Source Rules.xgen [\[edit\]](#)

[improvements](#): there are improvements that the customer could make to the children's diet **is true**; if  
**Exists**(children, [child\\_improvements](#): there are improvements that the customer could make to the child's diet)  
**else false**

### there are improvements that the customer could make to the child's diet

Source Rules.xgen [\[edit\]](#)

[child\\_improvements](#): there are improvements that the customer could make to the child's diet **is true**; if  
**either**  
[b2@Rules SourceRules doc](#): the child needs to eat more grains **or**  
[b3@Rules SourceRules doc](#): the child is consuming an excessive amount of grains **or**  
[b4@Rules SourceRules doc](#): the child needs to eat more vegetables **or**  
[b5@Rules SourceRules doc](#): the child is consuming an excessive amount of vegetables **or**  
[b6@Rules SourceRules doc](#): the child needs to eat more fruit **or**  
[b7@Rules SourceRules doc](#): the child is consuming an excessive amount of fruit **or**  
[b8@Rules SourceRules doc](#): the child needs to eat more meat **or**  
[b9@Rules SourceRules doc](#): the child is consuming an excessive amount of meat **or**  
[b10@Rules SourceRules doc](#): the child needs to eat more dairy food **or**  
[b11@Rules SourceRules doc](#): the child is consuming an excessive amount of dairy food **or**  
[b12@Rules SourceRules doc](#): the child is consuming an excessive amount of sweets  
**else false**

### the family's overall health assessment

Source Rules.xgen [\[edit\]](#)

<a href="#">overall_rating</a> : the family's overall health assessment	
"Excellent"	<b>ForAll</b> (children, <a href="#">child_rating_overall</a> : the child's overall star rating >= 5)
"Needs Improvement"	<b>ForAll</b> (children, <a href="#">child_rating_overall</a> : the child's overall star rating <= 2)
"Good"	<b>otherwise</b>

### the child's overall star rating

Source Rules.xgen [\[edit\]](#)

[child\\_rating\\_overall](#): the child's overall star rating **is** [child\\_rating\\_grains](#): the child's star rating for grains consumption + [child\\_rating\\_v](#)  
vegetable consumption + [child\\_rating\\_fruit](#): the child's star rating for fruit consumption + [child\\_rating\\_meat](#): the child's star rating for  
[child\\_rating\\_dairy](#): the child's star rating for dairy food consumption + [child\\_rating\\_sweets](#): the child's star rating for sweets consumption

### the total grains consumed by the children

Source Rules.xgen [\[edit\]](#)

[total\\_grains](#): the total grains consumed by the children **is** **InstanceSum**(children, [child\\_servings\\_grains](#): the number of servings of grain)

To quickly jump to a rule within a Word document, click on the **Edit** link in the Rule Browser. For rules defined in Word documents, this will open the document and jump to the rule. For rules defined in Excel documents, the Rule Editor is opened (although note that the rule may not actually be modified in this view).

## Find input data needed to reach a conclusion

To find the input data needed to reach a particular conclusion you can use the Attribute Dependencies view. To do this:

1. In Oracle Policy Modeling, select **View | Attribute Dependencies**.
2. Browse to select your goal (conclusion) attribute. TIP: If your rulebase is very large, searching for an attribute in the Attribute Selector will be quicker if you turn off the **Filter search results on each keystroke** option under **File | Project Properties | Common Properties | General**.
3. Select the option to **Show influencing attributes in a list**.
4. Uncheck the **Show intermediates in list view** option.

This will give you a list of the base level attributes (input data) that need a value in order to prove the goal attribute.

TIP: If you want to see what input data is needed at runtime to infer a goal for a specific scenario or subset of scenarios you need to run the debugger.

## Get a list of all attributes proving a goal

To get a list of all the attributes that prove a particular goal, follow the steps above to show the attributes in a list in the Attribute Dependencies View. Then:

1. Right-click any attribute in the list and select **Copy List**.
2. In the application where you want to save the list (eg Microsoft Excel), right-click and select **Paste**. This will give you a list of the public name, attribute level and attribute text for each attribute proving your chosen goal.

## Spell-check all interview screens

A Screens Report lists the contents of all the screens (question and summary) defined in the project. This is a useful tool for reviewing all of your screens in a single document. This file can also be used for spell checking your screens.

To spell check your screens following these steps:

1. In Oracle Policy Modeling, select **Reports | Screens**.
2. Select all the text in the report, right-click and select **Copy**.
3. Open Microsoft Word and select **Edit | Paste** (or press **Ctrl+V**).
4. Select **Tools | Spelling and Grammar...** and run the spell check, making note of any spelling errors identified.
5. Go back into Oracle Policy Modeling, open the relevant screens file/s and correct the errors.

## Create, modify or delete a rulebase visualization

Rulebase visualizations are a handy way of displaying your rulebase, or a branch of your rulebase, in a tree structure which shows how the attributes influence one another. Visualizations can be printed, and also exported to Windows Media Format. Visualizations are created in visual browser files.

## What do you want to do?

[Create a new visual browser file](#)

[Create a rulebase visualization](#)

[Modify a rulebase visualization](#)

[Print a rulebase visualization](#)

[Export a rulebase visualization](#)

[Delete a rulebase visualization](#)

## Create a new visual browser file

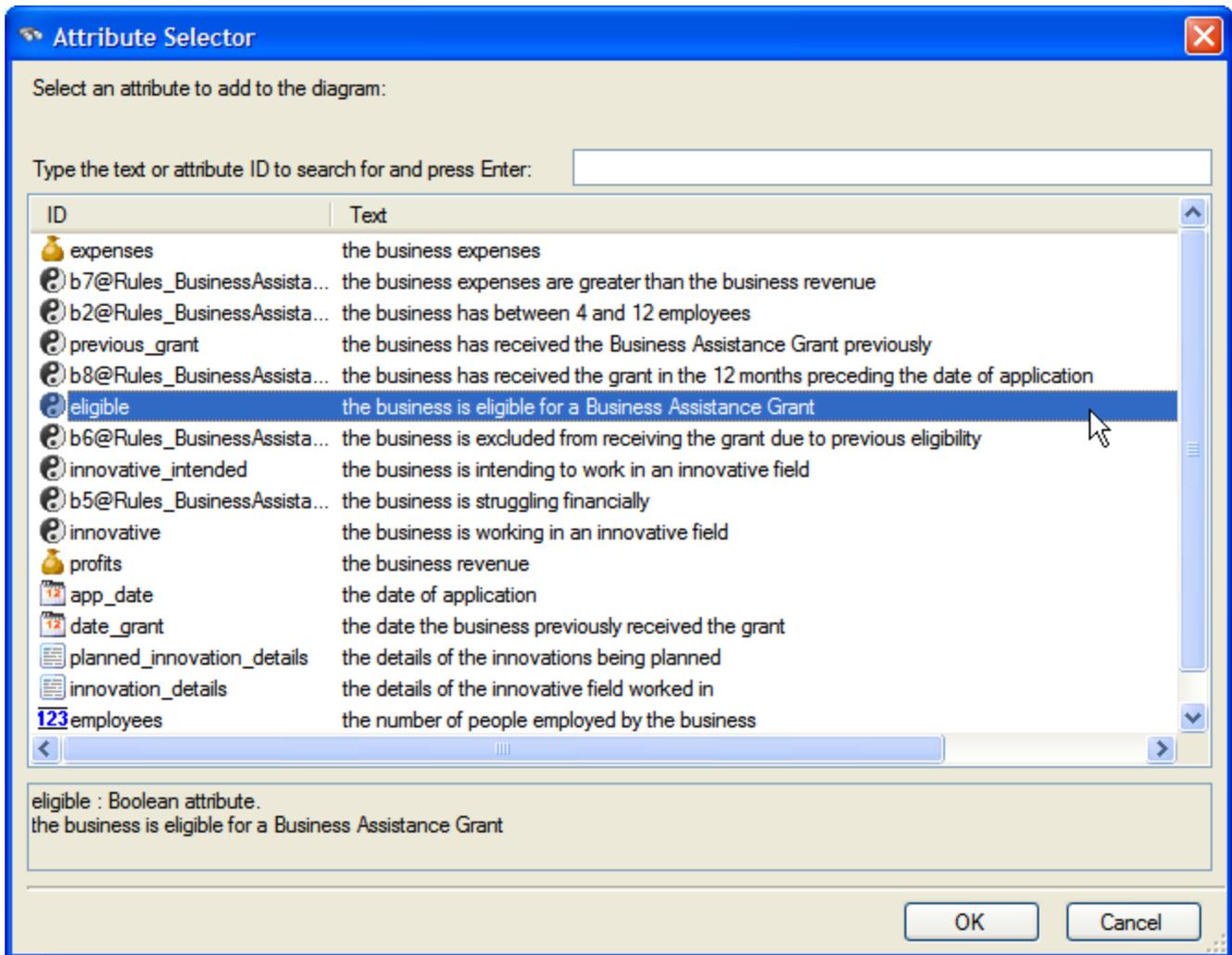
To add a new visual browser file to your project:

1. In Oracle Policy Modeling, right-click the Visualizations folder in the Project Explorer and select **Add New Visual Browser File** from the pop-up menu.
2. A new visual browser file will be added to your project. Type a name for your visual browser file, for example, "Visualizations".
3. Save your project by selecting **File | Save All** from the main menu.

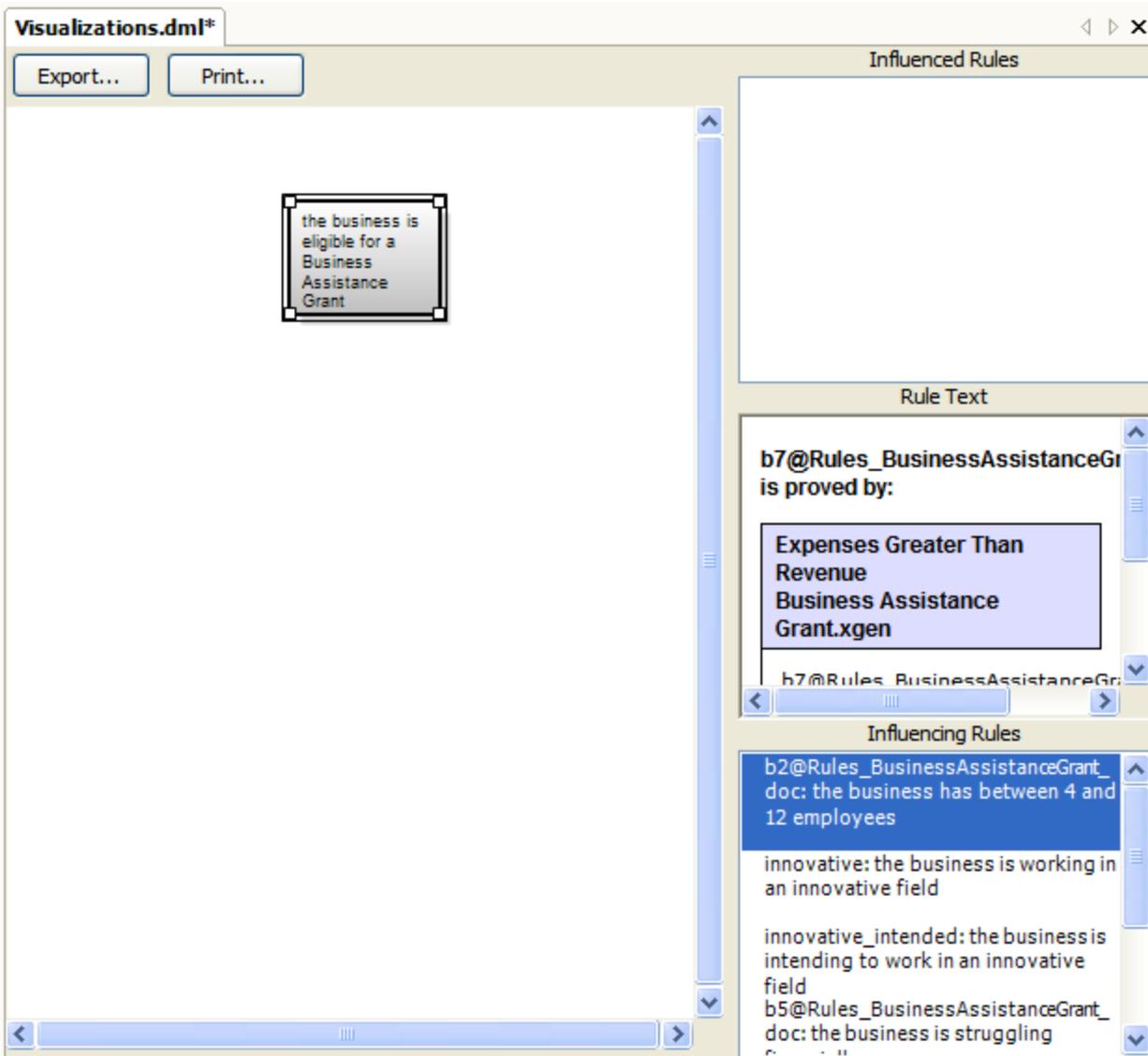
## Create a rulebase visualization

To create a rulebase visualization:

1. In Oracle Policy Modeling, double click the visual browser file in the Project Explorer to open it for editing.
2. In the visual browser file pane, right-click and select **New Item...**
3. In the **Attribute Selector**, select the attribute to add to the diagram, then click **OK**.



The attribute will be added as a node to the left hand pane:

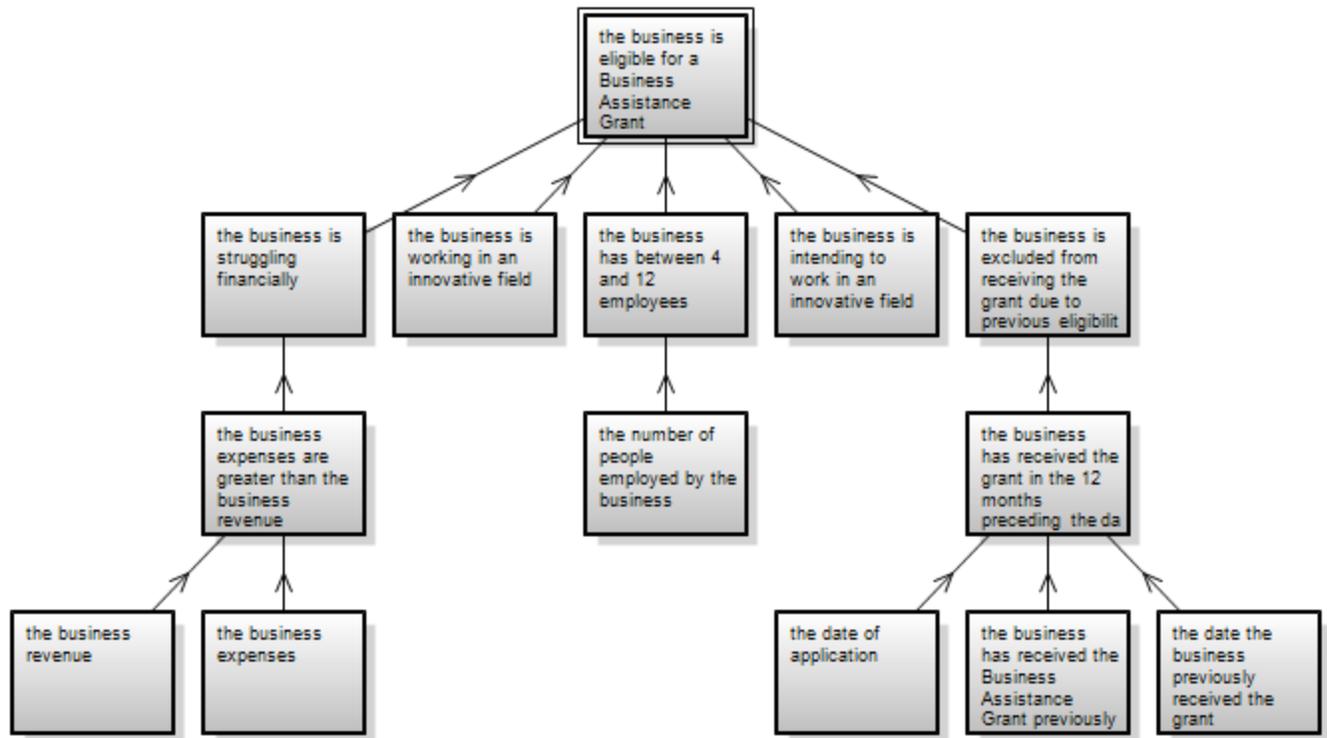


Note that the influenced/influencing rules and rule text for that attribute are displayed in the right hand boxes.

4. Right-click the node and select **Generate Rule Structure...**
5. In the **Generate Rule Structure** dialog, specify whether you want influenced and/or influencing rules, and how many rule levels you want to limit the tree to.



Click **OK** and the rule structure is generated:



## Modify a rulebase visualization

After you have created a rulebase visualization, there are many ways in which you can modify it.

### Move the nodes

When a rule structure is generated, Oracle Policy Modeling makes a best guess at the nicest way to present the tree. You may want to improve the appearance of the tree by moving the nodes around.

To do this, simply select a node and drag it to the desired location in the diagram. (The lines attached to the node move with it.)

### Change the formatting of nodes

You can change the display text, text font, text color and background color of any node in your rule structure. This is useful if you want to highlight important nodes in your tree.

To format a node:

1. Select the node, right-click and select **Properties...**
2. In the **Item Properties** dialog box you can:
  - Change the display text by typing directly into the **Display** field.
  - Select the **Text Color...** button to open the **Color** dialog box and select a different color for the attribute text.
  - Select the **Font...** button to open the **Font** dialog box and change the font.
  - Select the **Background Color...** button to open the **Color** dialog box and select a different background color for the node.

## Delete nodes

To delete a node in your rule structure, select the node, right-click and select **Delete Object**.

## Hide relationships

You can hide relationships between nodes by selecting a node, right-clicking and selecting **Hide Relationships...** You can then specify which relationships you want to hide using the check boxes in the **Hide Item Relationships** check box.

## Regenerate the rule structure

You can select any node in your tree and regenerate (or generate for the first time) the rule structure for that node. Select the node, right-click and select **Generate Rule Structure...**

## Adding labels and boxes to the diagram

You can add labels to the rulebase visualization, and boxes that sit behind the diagram. This can be useful for identifying a group of nodes.

To add a label:

1. Right-click in your diagram (not on a node) and select **New Label**. This will add the text "Label" to your diagram.
2. To change the text and format the label, click on the label, right-click and select **Properties...** You can then change the text of the label, and change the font and text color as required.

To add a box:

1. Right-click in your diagram (not on a node) and select **New Box**.
2. To format or add text to your box, select the box, right-click and select **Properties...** You can then add text (specifying the text color and font) and change the background color of the box as required.

Labels and boxes can be moved to new locations in the diagram by selecting them and dragging them to a new position. Both labels and boxes can be deleted by selecting them and pressing the **Delete** key.

## Print a rulebase visualization

To print a rulebase visualization:

1. In the visual browser file pane, click the **Print** button in the top left.
2. In the **Print** dialog box, click **OK**.

## Export a rulebase visualization

To export a rulebase visualization:

1. In the visual browser file pane, click the **Export** button in the top left.
2. In the **Save As** dialog box, specify the location to save your rulebase visualization to and provide a name for the file. Then click the **Save** button.

The rulebase visualization is saved in .wmf format (Windows Media Format).

When you save, the file automatically opens in Windows Picture and Fax Viewer. From here you can click the save button and save the image in a different format (BMP, JPEG, GIF, TIFF, PNG).

TIP: PNG is probably the best format to use as the other formats save the image with a black background which means the arrows in the diagrams are not visible.

### Delete a rulebase visualization

To delete a rulebase visualization file:

1. In the Project Explorer in Oracle Policy Modeling, right-click the visual browser file and select **Delete**.
2. Click **OK** to confirm the permanent deletion.

TIP: To only remove the file from your Oracle Policy Modeling project (but not delete it from your file system as well), right-click it in Oracle Policy Modeling and select **Remove from Project**.

# Analysis

Topics in "Analysis"

- [Conduct what-if analysis using an Excel workbook](#)
- [Analyze the outcomes of a large number of test cases](#)
- [Use the Batch Processor](#)

## Conduct what-if analysis using an Excel workbook

Using Excel with an Oracle Policy Modeling rulebase model, you can easily analyze the results that different policy model versions yield, in order to decide which policies are the best ones to use. This is done by creating what-if analysis documents in the OPM project, providing the necessary inputs for the attributes, entities and relationships in Excel and using the batch processor to analyze the results.

Note that what-if analysis is only available when using Microsoft Excel 2007 or later.

## What do you want to do?

[Create a what-if analysis document](#)

[Populate the what-if analysis document with input data](#)

[Analyze the results of the policy model](#)

[Export the what-if analysis to CSV files](#)

[Export the what-if analysis to a test script file](#)

## Create a what-if analysis document

A What-If Analysis document, based on the rulebase model of the project, can be created by following these steps:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the file to be placed in.
2. Right-click and select **Add New What-If Analysis Document**.
3. Type a name for the new document, then press **Enter**.

The Excel what-if analysis file will now appear in the Project Explorer in Oracle Policy Modeling.

## Populate the what-if analysis document with input data

In order to analyze the results of your policy model, you must first enter your input data.

In the Project Explorer in Oracle Policy Modeling, double-click the what-if analysis document to open it in Excel.

Initially, the document will just contain a worksheet for the global entity. From this starting point you can add additional worksheets for entities and many-to-many relationships, and to each worksheet you can add new columns for attributes and other relationships.

## Add a worksheet (for entities and many-to-many relationships)

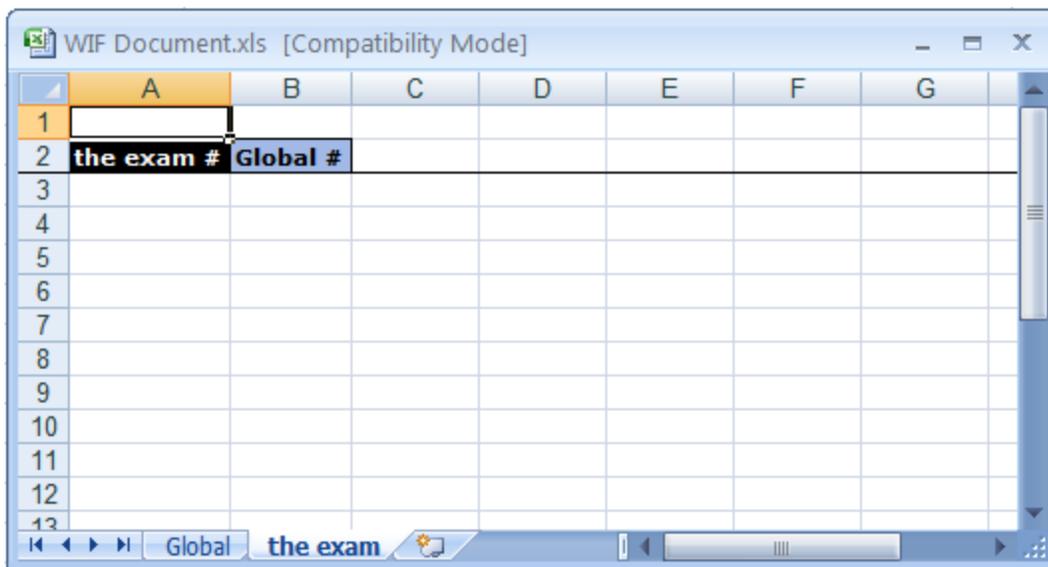
To add an entity or many-to-many relationship to a what-if analysis document, you need to add a new worksheet:

1. On the Oracle Policy Modeling toolbar, select the **Add Worksheet** button.
2. In the **Add Worksheet** dialog, select the checkbox for each entity or many-to-many relationship that you want to add. (Note that only those entities and many-to-many relationships that do not already exist as worksheets in the document are listed. Also, for a many-to-many relationship to be added it must have relationship text defined in Oracle Policy

Modeling.)



3. Click **OK**. Each worksheet is created containing any required columns. The **entity name #** column (eg the exam #) is always required, as is the containing entity (eg Global #).

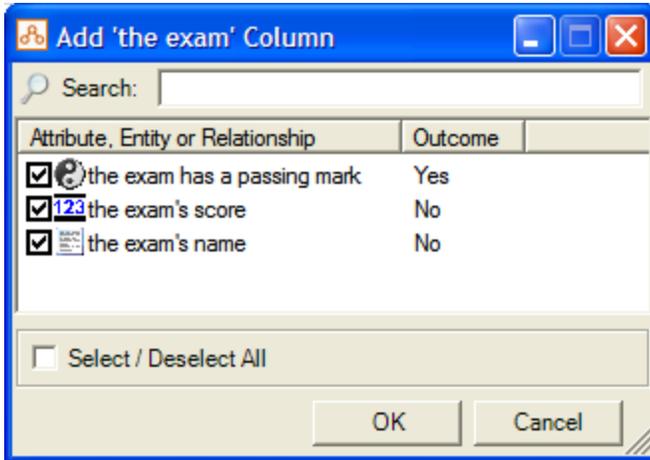


Note that in your what-if analysis document you can have additional worksheets not intended for analysis, as long as there are no styled cells.

### Add a column (for attributes and other relationships)

To add attributes or other (non many-to-many) relationships to an existing worksheet in a what-if analysis document, you need to add a new column:

1. Select the tab for the entity that the attribute or relationship relates to. (For one-to-many and many-to-one relationships, the relationship column is added to the entity on the many side of the relationship.)
2. On the Oracle Policy Modeling toolbar, select the **Add Column** button.
3. In the **Add Column** dialog, select the checkbox for each attribute or relationship that you want to add. (Note that only those attributes and other relationships that do not already exist as columns in the active worksheet are listed. Also, for one-to-one and many-to-one relationships to be added they must have relationship text defined in Oracle Policy Modeling. For a one-to-many relationship to be used in what-if analysis, the reverse text of the relationship must have been defined.)



4. Click **OK**. The columns are added to the active worksheet.

	A	B	C	D	E
1					
2	the exam #	Global #	(the exam has a passing mark)	the exam's score	the exam's name
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Base attributes are colored green which indicates that these are input columns.

Inferred attributes are colored orange which indicates that these are output columns. The names of the output columns

are also enclosed in parentheses.

Notes on formatting:

- i. Columns - you can interchange the columns (as long as the styles are intact) and have spaces between them (as long as the blank columns are not styled).
- ii. Rows - you can have blank rows.
- iii. Cells - you can format the cells (eg 2 decimal places for numbers) and use Excel functions and formulas.

### Enter data for global entities

On the Global worksheet you need to:

1. Enter your global instances in the **Global #** column. These act as the IDs for each instance and should therefore always be a number and always be unique (eg 1, 2, 3 etc).
2. Enter values for the base-level attributes for each instance (ie in the green columns). These values need to be the correct type for that attribute (eg 'true' or 'false' for booleans, numbers for number/currency attributes etc).

	A	B	C	D	E
1					
2	<b>Global #</b>	<b>the student's name</b>	<b>the student's subject</b>	<b>(the student passed the subject)</b>	<b>(the student's average mark)</b>
3	1	Beth	Mathematics		
4	2	Anne	English		
5	3	Fran	History		
6					
7					

### Enter data for non-global entities:

On a non-global entity worksheet you need to:

1. Enter your entity instances in the **entity name #** column. These act as the IDs for each entity instance and should therefore always be a number and always be unique (eg 1, 2, 3 etc).
2. Enter ID references in the containing entity column (eg the **Global #** column). These should always be the numbers that correspond to the associated instances of the containing entity (eg 1, 2, 3 etc).
3. Enter ID references for any other (many-to-one, one-to-many or one-to-one) relationships (ie in the other blue columns). These should always be the numbers that correspond to the associated instances of the target entity.
4. Enter values for the base-level attributes for each entity instance (ie in the green columns). These values need to be the correct type for that attribute (eg 'true' or 'false' for booleans, numbers for number/currency attributes etc).

	A	B	C	D	E	F	G
1							
2	<b>the exam #</b>	<b>Global #</b>	<b>the exam's score</b>	<b>the exam's name</b>	<b>(the exam has a passing mark)</b>		
3		1	1	70 Algebra			
4		2	1	75 Calculus			
5		3	1	80 Geometry			
6		4	2	90 Grammar			
7		5	2	40 Poetry			
8		6	3	88 Medieval History			
9							
10							

In the examples above, Beth (Global ID #1) has taken three exams (Algebra, Calculus and Geometry), Anne (Global ID #2) has taken two exams (Grammar and Poetry) and Fran (Global ID #3) has taken one exam (Medieval History).

### Enter data for many-to-many relationships

On a many-to-many relationship worksheet you need to:

1. Enter ID references for each source and target entity instance (ie in the blue columns). These should always be the numbers that correspond to the entities' instances.

For example, if you had a many-to-many relationship 'the child's parents' between 'the child' and 'the parent', and you defined the following parents and children:

	A	B	C	D	E
1					
2	<b>the parent #</b>	<b>Global #</b>	<b>the parent</b>		
3		1	1 John Smith		
4		2	1 Lisa Smith		
5		3	1 Edward Campbell		
6		4	1 Nancy Campbell		
7					

	A	B	C	D	E
1					
2	<b>the child #</b>	<b>Global #</b>	<b>the child</b>		
3		1	1 Jamie Smith		
4		2	1 Sarah Smith		
5		3	1 Kim Campbell		
6		4	1 Jason Campbell		
7					

Then on the worksheet for the child's parents (the many-to-many relationship) you would specify how these entity instances relate to one another:

	A	B	C	D	E
1					
2	<b>the child #</b>	<b>the parent #</b>			
3	1	1			
4	1	2			
5	2	1			
6	2	2			
7	3	3			
8	3	4			
9	4	3			
10	4	4			
11					

This tells us that:

- Jamie Smith's parents are John Smith and Lisa Smith.
- Sarah Smith's parents are John Smith and Lisa Smith.
- Kim Campbell's parents are Edward Campbell and Nancy Campbell.
- Jason Campbell's parents are Edward Campbell and Nancy Campbell.

### Analyze the results of the policy model

To analyze the results of the policy model, click the **Analyze** button on the Oracle Policy Modeling toolbar. (If your rulebase needs to be built, you will be prompted to do this now.) The document will be processed and, if successful, the output columns (ie the orange ones) will be populated with values.

	A	B	C	D	E
1					
2	<b>#</b>	<b>the student's name</b>	<b>the student's subject</b>	<b>(the student passed the subject)</b>	<b>(the student's average mark)</b>
3	1	Beth	Mathematics	TRUE	
4	2	Anne	English	FALSE	
5	3	Fran	History	TRUE	

	A	B	C	D	E
1					
2	#	Global	the exam's score	the exam's name	(the exam has a passing mark)
3	1	1	70	Algebra	TRUE
4	2	1	75	Calculus	TRUE
5	3	1	80	Geometry	TRUE
6	4	2	90	Grammar	TRUE
7	5	2	40	Poetry	FALSE
8	6	3	88	Medieval History	TRUE

Note that if there are errors in your what-if analysis document, the analysis process will cease, and you will be prompted to correct those errors.

Analysis is performed using the batch processor. For more information on this utility, see the [Oracle Policy Automation Developer's Guide](#).

### Export the what-if analysis to CSV files

The what-if analysis can be exported to a set of CSV files that can then be run through the batch processor with zero configuration.

To export the what-if analysis to CSV files:

1. Click the **Export** button on the Oracle Policy Modeling toolbar. (If your rulebase needs to be built, you will be prompted to do this now.)
2. In the **Export What-If Analysis** dialog box, select the **CSV file folder** option.
3. Specify the folder where you want to save the CSV files to, then click **OK**.

Note that if there are any errors, the export process will cease, and you will be prompted to correct those errors.

The number of CSV files that are exported corresponds to the number of relevant worksheets in the what-if analysis document. The input fields in the CSV files are the same as those in the what-if analysis document.

To run the CSV files through the batch processor, follow the steps for using the batch processor with zero configuration, making sure that you specify (i) the rulebase path, (ii) the location of the CSV files, and (iii) the output path for the CSV files.

After batch processing is complete, the output CSV files will be in the specified output directory. These files will show both the input and the determined output fields, with the same results as our what-if analysis document.

### Export the what-if analysis to a test script file

The what-if analysis can be exported to a test script file that can then be added to the rulebase project.

To export the what-if analysis to a test script file:

1. Click the **Export** button on the Oracle Policy Modeling toolbar. (If your rulebase needs to be built, you will be prompted to do this now.)
2. In the **Export What-If Analysis** dialog box, select the **Test script file** option.
3. In the **Save As** dialog, specify a location and name for the test script file, then click **Save**. Then click **OK** in the Export What-If Analysis dialog.

Note that if there are any errors, the export process will cease, and you will be prompted to correct those errors.

To add the generated tsc file to the rulebase project, go to **File | Add | Add Existing File...** and select the file.

When you open the test script file in the project you will notice that the number of test cases in the test script is the same as the number of global records in the associated what-if analysis document. The data in the test cases is the same as in the what-if analysis document.

## Analyze the outcomes of a large number of test cases

After you have [created a large number of test cases based on real-world data](#), you can perform some insightful analysis on the outcomes of those test cases.

### Identify the frequency of each outcome

To identify the frequency of each outcome, you can:

- use Oracle Policy Modeling's [what-if analysis](#) to bulk process the data, and then use Excel's native data analysis tools (eg sorting, filtering, statistical analysis) to analyze the outcomes, or
- [use the batch processor](#) to bulk process the data, and then use another BI tool to analyze the outcomes.

This could be useful for seeing if any of the test cases result in unusual outcomes (eg outliers or negative results) which would point to errors in the rules. This process could also be used to identify each unique outcome.

### Identify conflicting outcomes

To identify if two rules are ever true at the same time, you can:

1. Create a rule that tests if the offending combination of logic is true. For example,

**there is an error in the rulebase if**

the person is male and

the person is pregnant

2. Run the test cases, through the [batch processor](#) or as [test scripts](#) in Oracle Policy Modeling, to identify if the rule is ever true.

This could be useful for identifying double payments or conflicting outcomes.

### Identify used and unused rules and conditions

To identify which rules and conditions are used and which are unused you can use the [Test Script Coverage report](#). In this report:

- rules and conditions are shown to be unused if zero percent are covered by the test suite. This is shown in the report by a 0 out of X at the relevant level, and by greyed out rules/conditions in the bottom pane.
- rules and conditions are shown to be used if more than zero percent are covered by the test suite. This is shown in the report by at least 1 out of X at the relevant level, and by bolded 'true/false' values against conditions or by 'used' against conditionless rules.

## Use the batch processor

The batch processor allows a large number of 'cases' to be processed in batch. This is useful for:

- [conducting what-if analysis using Excel](#)
- [generating test scripts from existing Excel data](#)
- [analyzing the outcomes of a large number of test cases](#)

The batch processor is installed with Oracle Policy Modeling and is invoked from the command line. It is available in both Java and .NET implementations to enable support for platform specific custom functions. For more details on invoking the batch processor and on the XML schema used for configuration, see the Batch Processor section of the [Oracle Policy Automation Developer's Guide](#).

# Test cases

Topics in "Test cases"

- Define, modify or remove test scripts
- Create a test case from within an interview
- Import test cases from another project
- Create test scripts from existing data
- Compare test case results with expected results
- Debug a failing test case
- Create test cases with temporal data or outcomes
- Measure the coverage of a test suite
- Improve test script coverage
- Use the regression tester from the command line

See also:

- Set the time period to use for calculations
- Exclude a rule file from the build
- Define data to use in a test case or a debug session
- View the attributes inferred in a test case or debug session
- Change a rule while debugging

## Define, modify or remove test scripts

A test script is a file which contains test cases and the set of outcome attributes (both global and entity attributes, including defined tolerances) that will be used by the test cases. Oracle Policy Modeling has an integrated regression tester which can be used to create test scripts so as to compare outcomes from a rulebase with another set of outcomes.

Test scripts use the runtime model of the rulebase so if you make any changes to your rulebase while regression testing you will need to close and re-open your test script for those changes to be reflected in your test script file.

## What do you want to do?

Create a new test script file

Create new test cases

Copy an existing test case

Create input data

Specify expected results

Create an outcome set

Modify a test script

Validate a test script

[View the details of a test script](#)

[Remove a test script](#)

[Change the platform that the regression tester runs on](#)

## Create a new test script file

To add a new test script file to your project:

1. In Oracle Policy Modeling, select the **Test Scripts** folder in the Project Explorer.
2. Right-click and select **Add New Test Script File** from the pop-up menu.  
A new test script file will be added to your project. The new file will be selected and highlighted in the list.
3. Type a name for your test script file, for example, "Test Scripts".
4. Save your project by selecting **File | Save All**.

**TIP:** Multiple test scripts can exist in a project. Using a single test script on a large project may present problems if the project is under source control since, generally speaking, only one person can edit a file at a time. To ameliorate this problem multiple test scripts can be defined so that each can be edited separately. Multiple test scripts may also be defined to enable different reports to be created for a given set of test cases and/or to enable the use of different outcome sets for a test script.

## Create new test cases

A test case is a combination of an input data set and expected results.

- The input data is the set of data from which the actual results (outcome values) of the test case are generated.
- The expected results is the data set which is matched against the actual results.

Test cases can be created, edited and deleted in Oracle Policy Modeling.

To add a new test case to your test script:

1. In Oracle Policy Modeling, open your test script file by double-clicking it in the Project Explorer.
2. Select the test script file in the Test Cases tab, right-click and select **New Test Case** from the pop-up menu. A new test case will be added to your test script. The new test case will be selected and highlighted in the list.
3. Type a name for your test case (see Tips below), then press **Enter**.

## Tips for naming test cases

Each project should have a unique naming convention to be used when creating test cases. Some guidelines for establishing a naming convention are given below. The names used for test cases should contain:

- A prefix indicating the origin of the test case, and
- A unique identifier for the test case.

Suggested prefixes are given in the table below:

Prefix	Purpose
unit_	Unit test cases to be used by developers.

Prefix	Purpose
formal_	Test cases that are derived from the formal test case script set up for the project.
client_	Test cases or use cases specifically requested by the client.

Other project specific prefixes may be used if required.

The unique identifier for each file will be dependent on the origin of the test case. The suggested approach to creating the unique identifier is:

Origin	Unique identifier
Unit	<p>The unique identifier is to include:</p> <ul style="list-style-type: none"> <li>• The creating developer’s initials</li> <li>• An abbreviation to identify the section of the rulebase being tested</li> <li>• A sequential number.</li> </ul> <p>For example, the tenth unit test case created by John Smith for Retirement Pensions Category C would be called unit_JSRPC10.xml.</p> <p>This format allows developers to readily identify their own test cases.</p>
Formal Test Script	<p>The formal test script is to be maintained by the testing team.</p> <p>Use the unique identifier assigned to the test case in the formal test script.</p> <p>If a test case that is identified as necessary for regression testing has not been previous recorded in the test script, it should be recorded there and assigned an identifier before being added to the regression testing script.</p> <p>This will help to maintain a database of test case IDs and descriptions.</p> <p>The unique identifier obtained from the formal test script will reflect the benefit type/general area of the rulebase that it being tested. For example, RPA01 is the first test case for Retirements Pension Category A.</p>
Client	<p>As for unit testing. These cases should have their own identifier, like the unit test cases. Instead of initials, use a unique identifier for the client eg client_DWPRPC02.xml.</p>
Business Development/Partners	<p>As for Client.</p>

TIP: When you open your test case, you can add a description of the test case in the Notes field.

Test cases can also be imported and exported to allow for external creation and editing. See [Import test cases from another project](#) and [Create a test case from within an interview](#) for more information.

## Copy an existing test case

To create a copy of an existing test case in your test script:

1. In Oracle Policy Modeling, open your test script file by double-clicking it in the Project Explorer.
2. Select the test case you wish to copy in the Test Cases tab, right-click and select **Copy** from the pop-up menu. The test case will be copied to a new test case called "Copy (1) of <original test case name>".
3. Rename the new test case as required.

## Create input data

Once you have created your new test case, you need to set up the input data for your test case. The input data is the set of data from which the actual results (outcome values) of the test case are generated. The input data contains attribute instances and entity instances, along with the values that should be assigned to them.

The test case editor is used to investigate goals, infer relationships and set values for base level attributes in Oracle Policy Modeling. The test case editor can be accessed by double-clicking a test case on the Test Cases tab in the test script. (The test case editor is very similar to the debugger with a Data view and a Decision view.)

### Investigate a goal

To investigate a goal in the test case editor:

1. In the Data view select the goal you want to investigate.
2. Right-click and select **Investigate**. This will open the Decision view with the attribute you have selected in the **Attribute** field. All of the relevant paths to the goal are shown in the text box below. Entities for which no instances have been created yet will be shown just by the relationship icon and the entity text.
3. Work your way through the list of questions, setting answers (see below). In order to investigate any attributes which belong to an entity, you will need to add instances of that entity. (See [Set up entities and containment relationships](#) for more information.) Add your entity instances and continue investigating attributes until a value for the goal is known.

### Investigate an inferred relationship

After you have added any entity instances in the test case editor, you can investigate an inferred relationship. To do this:

1. In the Data view select the inferred relationship that you want to investigate.
2. In the right hand pane, click the **Investigate** button. This will switch to the Decision view.
3. Set the values for any base level attributes (see below). The Decision view will be updated as you go to show which entity instances have been inferred for this relationship, and the attributes contributing to this conclusion.
  - In the case of existing entity instances that have been inferred as members of a relationship (ie using [IsMemberOf rules](#)), these will be shown as selected items in the right hand pane of the Data view. (These entity instances will not be shown under the inferred relationship in the left hand pane as they have not inferred a containment relationship).
  - In the case of entity instances that have been created as members of a relationship (ie using [InferInstance rules](#)), these are also shown in the left hand pane of the Data view under the containment relationship that they have inferred.

### Set the value for an attribute

To set the value of an attribute in the test case editor:

1. Select the attribute in the Data view or in the Decision view.
2. Right-click and select from any of the following Set options from the menu:
  - Set Value** - this opens the **Set Attribute Value** dialog box where you can enter a value or set the value to 'uncertain' or 'unknown'. Variable values must be entered in the correct format: See [Formatting of variable values](#). You can also specify change points for the attribute.
  - Set to True** - this option is only available for boolean attributes
  - Set to False** - this option is only available for boolean attributes
  - Set to <value>** - this option is only available for non-boolean text attributes. The values that appear here will be the values used in the rules or on screens.
  - Set to Unknown** - this option is used to clear the value of the attribute
  - Set to Uncertain**

Alternatively, you can double-click the selected attribute to open the **Set Attribute Value** dialog box and then select the appropriate value, ensuring that it is entered in [the correct format](#).

After setting a value, the list of attribute values in the Data and Decision views will be updated with the value you specified, as well as the values for any other attributes which have been inferred as a result.

### Create input data in an interview

Input data can also be created by setting values for attributes in the debugger or Web Determinations and then saving/exporting this data as an XDS file which can then be imported into a test case in Oracle Policy Modeling.

See [Create a test case from within an interview](#) for more information.

### Specify expected results

Once you have created the input data for your test case, you need to specify the expected results for the test case. The expected results is the data set which is matched against the actual results when the input data is loaded into the rulebase. The expected results contains instances of the attributes and entities found in the outcome set. When attributes are added to or deleted from the outcome set, all the expected results of the test cases in that test script will be updated accordingly.

To specify the expected result for an attribute:

1. In the Data view for the test case, select the inferred attribute that you want to add an expected result for. NOTE: The attribute must already be in the outcome set. If it is not, add it to the outcome set (see below). Attributes of inferred entity instances can be selected.
2. Right-click and select from the following options:

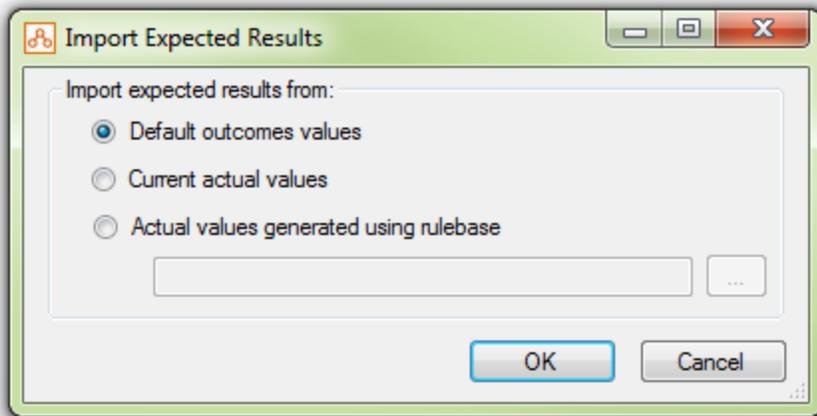
Option	Behavior
<b>Set Expected Value...</b>	Opens the <b>Edit Expected Result</b> dialog box where you can specify a particular value for the expected result, an expected result of uncertain, or an expected result of unknown. You can also specify change points for the expected result.
<b>Set Expected Value to Default (&lt;default expected result value&gt;)</b>	Defaults the expected result to the value specified as the default value in the <b>Edit Outcome</b> dialog box.

Option	Behavior
<b>Set Expected Value to Current Value</b>	Sets the expected value to the current value of the attribute instance. The current value of the attribute instance is shown in angle brackets in the Value column in the Inferred Attributes list.
<b>Set Expected Value to true</b>	Sets the expected value to 'true'. (This option is only available for boolean attributes.)
<b>Set Expected Value to false</b>	Sets the expected value to 'false'. (This option is only available for boolean attributes.)
<b>Set Expected Value to Unknown</b>	Sets the expected value to 'unknown'.
<b>Set Expected Value to Uncertain</b>	Sets the expected value to 'uncertain'.

- The expected value is shown in square brackets after the current value of the attribute in the Value column in the Inferred Attributes list.

To do a bulk import of expected results:

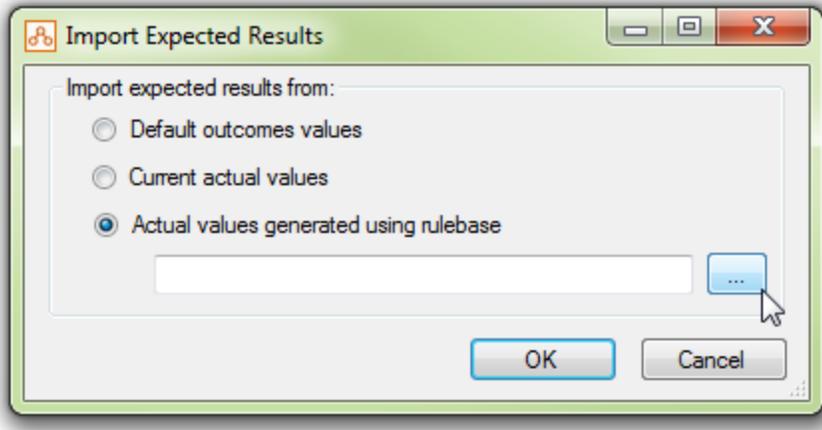
- Right-click the test case on the Test Cases tab in your test script file and select **Import Expected Results...**
- In the **Import Expected Results** dialog, select where you want to import the expected results from. The options are:
  - Default outcomes values
  - Current actual values
  - Actual values generated using rulebase



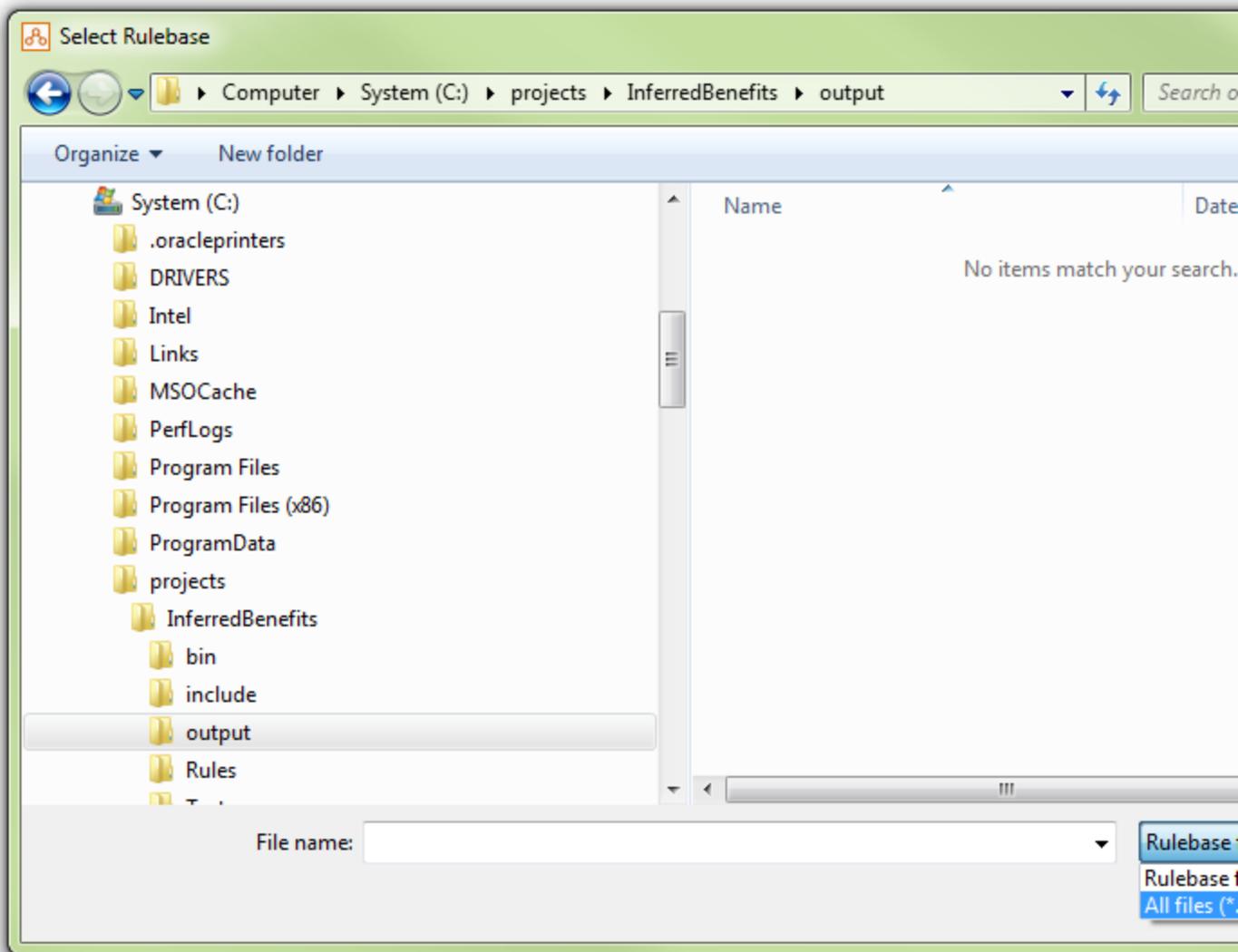
- (Optional) If you select **Actual values generated using rulebase**, you need to specify the location of the rulebase you want to import expected results from ('your target rulebase'). Tip: the expected file format for your target rulebase is a .zip file. You will find this file in the **output** folder of your target rulebase project. Alternatively, if you have saved a copy of the file to another location, you can specify that location.

To specify the location of your target rulebase, in the **Import Expected Results** dialog, either:

- type the path to the target rulebase .zip file into the text entry field, or
- browse to the target rulebase .zip file using the ... button adjacent to the text entry field.



This will open the **Select Rulebase** dialog. Note: by default, the **Select Rulebase** dialog searches for an .xml file rather than a .zip file. This means that the dialog will not actually display any .zip rulebase files, even if they are present in the relevant folder. To fix this problem, click the drop-down menu adjacent to the **File name:** text entry field and select **All files (\*.\*)**.



Locate and select your target rulebase .zip file and click **Open**. This will return you to the **Import Expected Results** dialog, with the file path to your target rulebase displayed in the text entry field.

4. In the **Import Expected Results** dialog, click **OK**.

#### Create an outcome set

A test script will have an outcome set for its test cases and this should contain all the inferred attributes that will be used for the comparisons to determine if the rulebase produces the correct results.

The following types of attributes would be appropriate outcome attributes:

- Inferred attributes that are displayed on the summary screen (eg goal attributes).
- Inferred attributes that are included in any generated documents.
- Any interim determinations or inferred attributes that may be useful for tracking the cause of failures.

TIP: Too many outcome attributes increases initial start-up time and maintenance overheads, and can make the reports less manageable. The maximum number of outcome attributes should therefore be limited to 10-12 if possible. For unit testing, the choice of outcome attributes may be slightly different as the very nature of unit testing means that intermediate attributes are monitored, rather than the overall end result.

There are two ways to add outcomes to your test script:

- From within the outcome set editor
- From within the test case editor

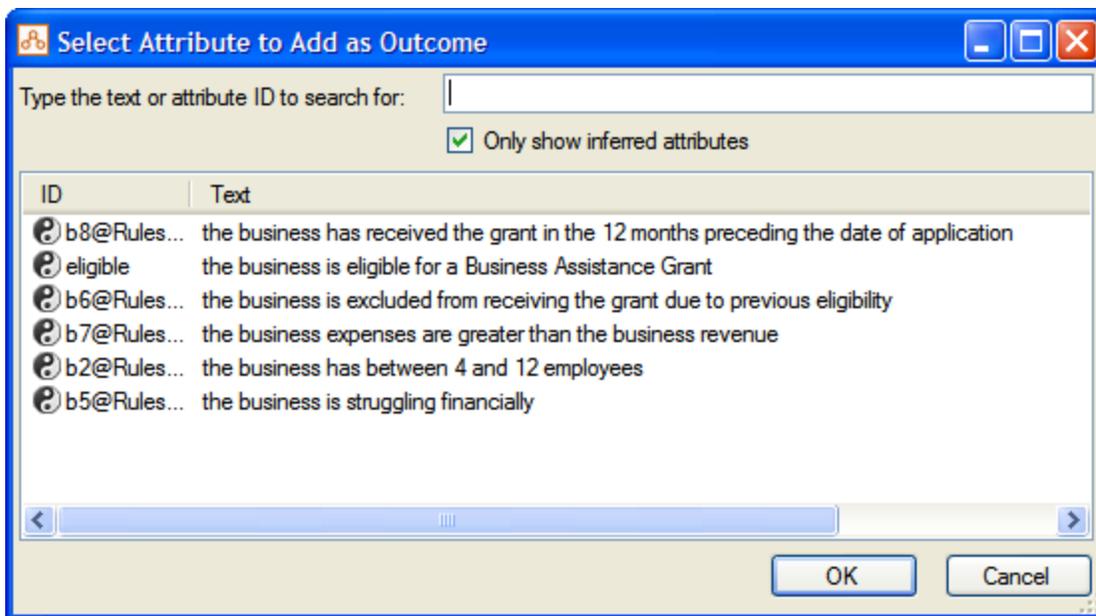
Attributes from any entity can be added as outcomes.

### Add outcomes in the outcome set editor

The outcome set editor can be accessed by clicking on the Outcomes tab in the test script file.

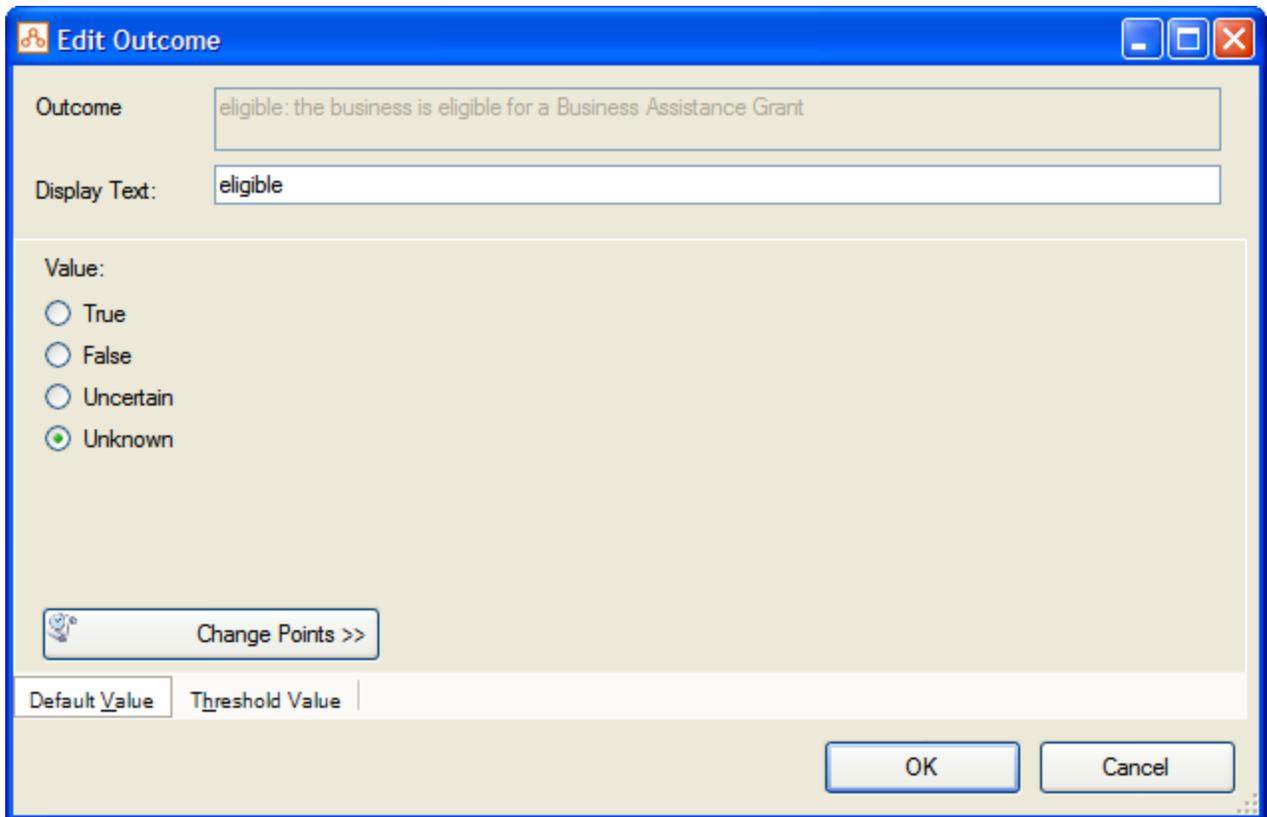
To add an outcome attribute in the outcome set editor:

1. Right-click anywhere in the outcome set editor and select **Add New Outcome....**  
The **Select Attribute to Add as Outcome** dialog will be displayed.



(By default, only inferred attributes will be shown. If you want to see all attributes, uncheck the **Only show inferred attributes** check box.)

2. Select the attribute you want to add as an outcome, then click **OK**.  
The **Edit Outcome** dialog is displayed.



3. Change the **Display Text** for the attribute if you want to. This is the name that will appear in the attribute list in the outcome set editor, and in the regression tester report.
4. Change the **Value** from unknown if appropriate. This is the value that the attribute instance will be set to when the attribute is first created. By default this value is set to "unknown". You can also specify change points for the attribute.
5. Enter a **Threshold Value** if required (see below).
6. Click **OK**. The new outcome attribute will now appear in the list of attributes in the outcome set editor.

TIP: Outcomes can be reordered in the outcome set editor by right-clicking and selecting **Move Up** or **Move Down**.

### Add outcomes in the test case editor

To add an attribute as an outcome from the test case editor:

1. Right-click on any inferred attribute in the right hand pane of the Data view. Select **Add as outcome....**  
The **Edit Outcome** dialog will be displayed.
2. Follow steps 3 to 5 above.

Outcome attributes are shown underlined in the Inferred Attributes list in the test case editor.

### Specify threshold values

Threshold values tell the regression tester that a given test case should pass if an actual value falls within a specified range. To specify a threshold for an attribute, select the **Threshold Value** tab in the **Edit Outcome** dialog.

Threshold value

Value:  Absolute Value ▼

Apply threshold value to

Both Upper and Lower bounds  Upper Bounds Only  Lower Bounds Only

Ignore

Unknown Values  Uncertain Values

The following table explains how to set a threshold:

Setting	Applies to	Description
Value	Date, currency or number attributes	A date threshold is defined as a number of days, months or years. A number threshold can be either an absolute value or a percentage. Number and currency thresholds can either be integer or decimal values.
Apply threshold value to	Date, currency or number attributes	Specifies whether the threshold applies above and/or below the expected outcome, as follows: <ul style="list-style-type: none"> <li>Both upper and lower bounds – the threshold will be applied as <math>Y - T \leq X \leq Y + T</math> (default)</li> <li>Upper bounds only – the threshold will be applied as <math>Y - T \leq X \leq Y</math></li> <li>Lower bounds only – the threshold will be applied as <math>Y \leq X \leq Y + T</math></li> </ul> <p>where X = Actual Result, Y = Expected Result and T= threshold value.</p>
Ignore		Specifies whether unknown and or/uncertain values should be ignored, as follows: <ul style="list-style-type: none"> <li>Unknown values – this means that a test will pass if Expected Value = Actual Value (to within whatever threshold is specified) OR Actual Value = unknown.</li> <li>Uncertain values – this means that a test will pass if Expected Value = Actual Value (to within whatever threshold is specified) OR Actual Value = uncertain.</li> </ul>

### Ignore results

You can flag an outcome so that any actual value for the outcome will be ignored when the test case is run. This will result in the expected outcome always passing. To do this, select the outcome attribute in the test case editor, right-click and select **Ignore Result**.

### Delete invalid outcomes

To bulk delete attributes that are no longer used in your rulebase, right-click anywhere in the outcome set editor and select **Delete Invalid Outcomes...**

NOTE: If an entity no longer exists in the rulebase then all attributes belonging to that entity will be flagged as invalid.

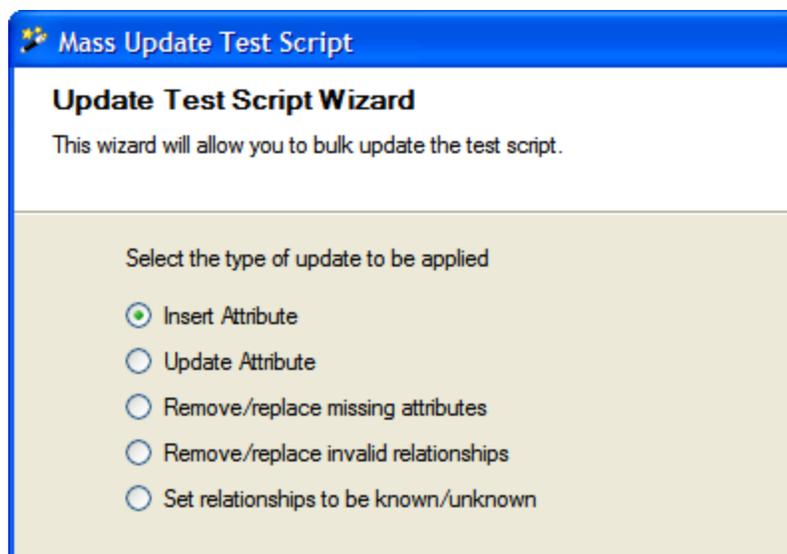
## Modify a test script

Test cases often need to be reviewed or modified to allow for changes in the rulebase. Changes can be made to individual test cases in the test case editor, or across multiple test scripts and test cases with the Update Test Script Wizard.

To make changes across multiple test scripts and test cases:

1. In Oracle Policy Modeling, right-click on a test script, or on a folder that contains test scripts, and select **Update Test Script Wizard**.

The **Mass Update Test Script** dialog is shown.



2. Select from one of the following four options which are explained further below:
  - a. Insert Attribute
  - b. Update Attribute
  - c. Remove/replace missing attributes
  - d. Remove/replace invalid relationships
  - e. Set relationships to be known/unknown

### Insert attribute

This option allows you to insert a value for an attribute which hasn't yet been added to your test cases. This is usually where a new attribute has been added to the rulebase since the last time the test cases were updated.

To insert an attribute:

1. Select the **Insert Attribute** option on the first screen of the wizard and click **Next**.
2. Select the test cases to which the attribute should be added. Use the browse button to select the attribute to be added, and enter the value which you wish to insert for the attribute, if any. Click **Next**.
3. Review your changes on the **Summary of Changes** screen. Click **Back** to amend your changes if necessary, then click **Next** to apply the changes.
4. After the wizard has applied the changes, select the **Yes** option to make another change, otherwise select the **No** option and click **Finish**.

## Update attribute

This option allows you to update the value for an attribute which already exists in your test cases.

To update the value for attribute:

1. Select the **Update Attribute** option on the first screen of the wizard and click **Next**.
2. Select the test cases to which the attribute should be added. Use the browse button to select the attribute to be added, and enter the new value which you wish to set for the attribute. Click **Next**.
3. Review your changes on the **Summary of Changes** screen. Click **Back** to amend your changes if necessary, then click **Next** to apply the changes.
4. After the wizard has applied the changes, select the **Yes** option to make another change, otherwise select the **No** option and click **Finish**.

## Remove/replace missing attributes

This option allows you to remove an attribute which still exists in your test cases, but has been removed from the rulebase. Alternatively, you can specify an attribute value which should replace it.

To remove or replace missing attributes:

1. Select the **Remove/replace missing attributes** option on the first screen of the wizard and click **Next**.
2. The wizard will detect whether any attributes exist in your test cases which are no longer present in the rulebase. Select the attribute you wish to change from the **Attributes With Errors** list. Leave the **Remove Only** checkbox selected if you just want to remove the attribute value from your test cases, or uncheck it and use the browse button to select an attribute to replace it with, and enter the value for the new attribute.
3. Review your changes on the **Summary of Changes** screen. Click **Back** to amend your changes if necessary, then click **Next** to apply the changes.
4. After the wizard has applied the changes, select the **Yes** option to make another change, otherwise select the **No** option and click **Finish**.

## Remove/replace invalid relationships

This option allows you to remove or replace any relationships in your test cases which no longer exist in the rulebase.

To remove or replace invalid relationships:

1. Select the **Remove/replace invalid relationships** option on the first screen of the wizard and click **Next**.
2. The wizard will detect whether any relationships exist in your test cases which are no longer present in the rulebase. For each **Invalid** relationship it detects, you can either remove it from the test case by selecting the **Delete** checkbox, or you can select a **Valid** relationship from the drop down list to replace it with. Once you have done this for each invalid relationship, then click **Next**.
3. Review your changes on the **Summary of Changes** screen. Click **Back** to amend your changes if necessary, then click **Next** to apply the changes.
4. After the wizard has applied the changes, select the **Yes** option to make another change, otherwise select the **No** option and click **Finish**.

## Set relationships to be known/unknown

This option allows you to set relationships to known or unknown.

To set the new state of a relationship:

1. Select the **Set relationships to be known/unknown** option on the first screen of the wizard and click **Next**.
2. In the left hand pane, select the test cases that the change is to apply to (or tick the **Check all items** checkbox of you want all test cases to be affected by the update).
3. In the right hand pane, select the **Entity, Relationship** and **Current State**. Then select the **New Relationship State** and the **Affected Instances**.

The screenshot shows a wizard interface with three main sections:

- Entity/Relationship:**
  - Entity:
  - Relationship:
  - Current State:  <Any>  Known  Unknown
- New Relationship State:**
  - New State:  Known  Unknown
- Affected Instances:**
  - All relationship instances
  - Specific relationship instance
    -
  - Empty relationship instances
  - Non-empty relationship instances

4. Click **Next**.
5. Review your changes on the **Summary of Changes** screen. Click **Back** to amend your changes if necessary, then click **Next** to apply the changes.
6. After the wizard has applied the changes, select the **Yes** option to make another change, otherwise select the **No** option and click **Finish**.

### Validate a test script

You have the option to validate a test script when it is opened and show a warning message if:

- A test case has no defined outcomes - this will show a warning for each test case that contains no outcomes. A test case with no outcomes is usually caused when the existing outcomes in a test case are removed from the test script. Consider either adding the relevant outcomes to the test script or moving the test case to a test script with the relevant outcomes defined.
- A test case has no expected value for an outcome – this will show a warning if an outcome defined in the test script does not have an expected value defined in a test case. This warning is useful when a new outcome has been added to the test script to identify which test cases have not been updated. If you wish to define a lot of outcomes in your test script which are mutually exclusive then it may be convenient to turn off this warning.

To change or view these settings, go to **File | Project Properties | Regression Tester Properties | General**.

## View the details of a test script

The Test Specification report allows you to view the details of all of your test cases at once. To view the Test Specification report for one or more test scripts:

1. In the Project Explorer, right-click on your test script or folder containing test scripts, and select **View Test Script Specification**.
2. In the View Test Script Specification dialog, select the test scripts that you want included in the report. If you want the selected test scripts included in the same report, select the **Combine all test scripts into one report** option.
3. Click **View**. The Test Specification/s will be displayed in the right hand pane. You can save a copy of the Test Specification by clicking the **Save** button.

## Remove a test script

To remove a test script from a project:

1. In the Project Explorer in Oracle Policy Modeling, right-click the test script file that you want to remove and select **Remove from Project**.

NOTE: The file remains in your file system but has been removed from your Oracle Policy Modeling project. To permanently delete a file from both your file system and from your project, right-click it in Oracle Policy Modeling and select Delete.

## Change the platform that the regression tester runs on

To change the runtime platform for the regression tester:

1. In Oracle Policy Modeling, go to **File | Project Properties | Common Properties | Platform**.
2. Select a different option from the **Target Platform** drop down list. (The options are **.NET** and **Java**, with **.NET** being the default platform.)
3. Click **OK**.

Note that this setting also determines which platform the test script coverage analyzer and the what-if analyzer run on.

## Create a test case from within an interview

You can create a new test case from the data in an interview in the debugger or Oracle Web Determinations. You can export the data directly into a new test case in the debugger, or you can export interview data to an XDS file and then import it into a new test case.

## What do you want to do?

[Export interview data directly into a new test case from the debugger](#)

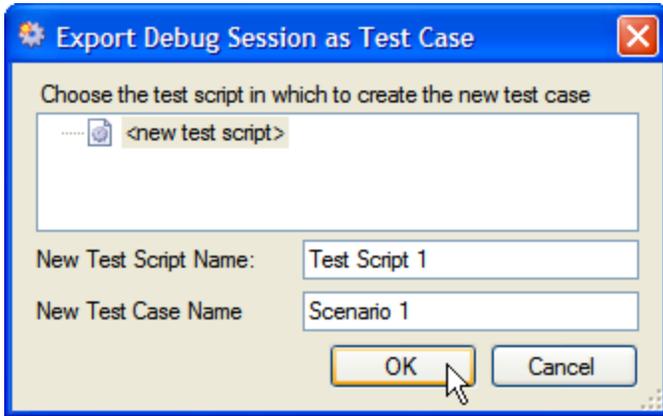
[Export interview data to an XDS file and import into a new test case](#)

### Export interview data directly into a new test case from the debugger

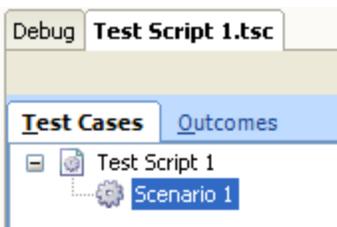
To create a new test case directly from within an interview in the debugger:

1. Open the debugger by selecting **Build | Build and Debug**.
2. Start an investigation and enter the desired values. (For more information on using the debugger, see [Debug a rulebase](#).)
3. Click on the arrow on the right side of the **Export** button (located in the top right of the **Debug** view), and select the **Export as Test Case** option.

4. Select a test script to save the new test case to, or enter the name for a new test script.
5. Enter the name for the new test case and click **OK**.



6. The new test case is created in the test script.



## Export interview data to an XDS file and import into a new test case

You can export the data from an interview in the debugger or Oracle Web Determinations, and then import this as a test case into a test script.

### Export the interview data from the debugger

To create and export the interview data from the debugger:

1. Open the debugger by selecting **Build | Build and Debug**.
2. Start an investigation and enter the desired values. (For more information on using the debugger, see [Debug a rulebase.](#))
3. Click the **Export** button (located in the top right of the **Debug** view).
4. In the **Save As** dialog box, enter a file name and destination folder for the XDS file.

### Export the interview data from Oracle Web Determinations

To create and export the interview data from Oracle Web Determinations:

1. Start Oracle Web Determinations by selecting **Build | Build and Run**.
2. In the Oracle Web Determinations interview, enter the desired values. (For more information on using Oracle Web Determinations, see [Test an interview or screen flow.](#))
3. On the summary screen, click **Save As** on the horizontal menu bar.

# ORACLE' Web Determinations

Summary

Save | Save As | Load | Restart | Close

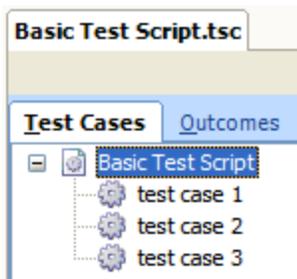
Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

4. In the **Save as case ID** field enter a name for the test case. Click **Save**. The file will be saved as an XDS file in \Release\web-determinations\data\<>project name>.

## Import the data into a new test case

To import data from a file created in the debugger or Oracle Web Determinations:

1. In Oracle Policy Modeling, open your test script file.
2. Select the test script file in the Test Cases tab in the right hand pane in Oracle Policy Modeling.
3. Right-click and select **Import Test Cases** from the pop-up menu.
4. In the **Open** dialog box, browse to select the XDS file/s you want to import. Click **Open**.
5. New test cases will be created for each of the imported XDS files.



Once you have imported your input data as test cases, you can add to it, edit it and delete it.

## Import test cases from another project

You can import test cases from another Oracle Policy Modeling project using XDS files. To do so you firstly need to export the input data from the other project.

To export input data to an XDS file:

1. In Oracle Policy Modeling, open the test script file.
2. Open the test case that you want to export and click the **Export** button.
3. Specify a name and destination for your XDS file. Click **Save**.

To import input data from an XDS file:

1. In the Test Case Editor, select the **Import** button.
2. Browse to select the XDS file/s you want to import. Click **Open**.

Once you have imported your input data into a test case, you can add to it, edit it and delete it.

## Create test scripts from existing data

Using the batch processor it is possible to create test scripts. The batch processor can be configured to read from a database connection, or a directory of CSV files, so that it can generate a large number of test cases based on real-world data. The steps to create test scripts from existing data are:

1. [Ensure data is in appropriate format for the batch processor](#)
2. [Run the batch processor to generate the test script](#)
3. [Add the test script file to OPM](#)

Ensure data is in appropriate format for the batch processor

### Format CSV files

Data can be read from a directory of files containing data formatted as comma-separated values (CSV). The files containing the data to be read must end with the .csv extension.

The batch processor supports a configuration-free option for CSV files as long as they follow the conventions below:

- the CSV files need to be using headers (ie the first row of the CSV contains column names)
- the name of a document needs to correspond to the public name of an entity (unless it is representing a many-to-many relationship)
- column names are either:
  - '#' for the primary key column (values must be an integer)
  - the public name of an attribute
  - the public name of an attribute enclosed in parenthesis - this represents an output attribute
  - the public name of the 'to-one' side of a relationship - the column value is the primary key of the target entity instance
  - the public name of the parent entity if the entity exists in a containment relationship

A special case is where a CSV file represents a many-to-many relationship. In that case the name of the document needs to correspond to the public name of one of the directions of the relationship. The CSV document is then required to have two columns with the first column having foreign key references to source entity instances and the second column with foreign key references to target entity instances (source and target are from the perspective of the side of the relationship used for the name of the document).

The expected format for attribute values is:

Value Type	Format Description	Blank Value
Number	<p>Numeric values must adhere to the following conditions:</p> <ul style="list-style-type: none"><li>• The '.' character is the decimal separator</li><li>• Thousands separators are not supported</li><li>• Currency symbols are not supported</li><li>• For scientific notation, the '+' is not supported in the exponent</li></ul>	Blank values are considered UNCERTAIN

Value Type	Format Description	Blank Value
String	String values will be read as-is	Blank string values are considered to be a blank value
Boolean	Boolean values must adhere to the following conditions: <ul style="list-style-type: none"> <li>• String values are case insensitive, so "YES" is the same as "yes" and "Yes"</li> <li>• Leading zeros are not truncated from numeric values</li> <li>• "True", "Yes" and 1 will be parsed as TRUE</li> <li>• "False", "No" and 0 will be parsed as FALSE</li> </ul>	Blank values are considered UNCERTAIN
Date	Date values must adhere to the format "yyyy-MM-dd" where: <ul style="list-style-type: none"> <li>• yyyy is the four-digit year</li> <li>• MM is the two-digit month, including leading zero for values below 10</li> <li>• dd is the two-digit day, including leading zero for values below 10</li> </ul>	Blank values are considered UNCERTAIN
Datetime	Datetime values must adhere to the format "yyyy-MM-dd HH:mm:ss" where: <ul style="list-style-type: none"> <li>• yyyy is the four-digit year</li> <li>• MM is the two-digit month, including leading zero for values below 10</li> <li>• dd is the two-digit day, including leading zero for values below 10</li> <li>• HH is the two-digit 24-hour hour value, including leading zero for values below 10</li> <li>• mm is the two-digit minute value, including leading zero for values below 10</li> <li>• ss is the two-digit seconds value, including leading zero for values below 10</li> </ul>	Blank values are considered UNCERTAIN
Time	Time values must adhere to the format "HH:mm:ss" where: <ul style="list-style-type: none"> <li>• HH is the two-digit 24-hour hour value, including leading zero for values below 10</li> <li>• mm is the two-digit minute value, including leading zero for values below 10</li> <li>• ss is the two-digit seconds value, including leading zero for values below 10</li> </ul>	Blank values are considered UNCERTAIN

If there is only a single CSV file in the folder, or one of the CSV files is "global.csv" then it is presumed to be the 'base' table, otherwise the base table needs to be specified. For example, a folder may contain "parent.csv" and "child.csv" - without specifying which is the base table, the batch processor won't know whether it should be processing cases for parents or children. Note that once a base table is established, other tables will be brought in as required by containment or reference relationships.

### Convert Excel data to CSV format

You can use Excel to perform batch generation of test scripts for Oracle Policy Modeling. You just need to convert the data into CSV format first. To do this:

1. Set up the data in Excel.

For example, to set up 101 cases where the "income" attribute is stepped from 0 to 100,000.

On a blank worksheet you would enter the following and then use 'fill down' to replicate the last row down to row 1002.

#	income
1	0
=a2+1	=b2+1000

2. Make sure that the data is formatted according to the conventions defined earlier.
3. Create a directory in Windows Explorer and save the file in CSV format as "global.csv".

### Run the batch processor to generate the test script

The batch processor is invoked from the command line. To generate a test script you will need to specify at least the following parameters:

- the rulebase location (--rulebase <rulebase path>)
- the data source (--csv <folder> or --database <db-connection-string>)
- the test script location (--exporttsc <path>)

Note that the data source can be specified in the XML configuration file to be used by the batch processor instead of as a command line parameter.

For more details, see the Batch Processor section of the [Oracle Policy Automation Developer's Guide](#).

### Add the test script file to OPM

In the Oracle Policy Modeling project, add the generated tsc file from the previous step (**File | Add | Add Existing File...**). You can then use the regression tester in OPM to customize the test script. See [Define, modify or remove test scripts](#) for more information.

See also:

- [Analyze the outcomes of a large number of test cases](#)

### Compare test case results with expected results

To compare the test case results with the expected results you need to run the test scripts for a rulebase. A report will be generated which shows the results.

## What do you want to do?

[Run a single test script](#)

Run multiple test scripts

View the test results

Customize the test report

Save the test report

Run a single test script

To run a single test script:

1. Ensure that you have created your test case/s and outcome set.
2. Click the **Execute** button on your test script tab. NOTE: This will just run the currently active test script. The test script will run and the Test Report will be displayed on a new tab.

Run multiple test scripts

To run multiple scripts for a rulebase:

1. In Oracle Policy Modeling, select **Reports | Run Multiple Test Scripts...**  
The **Run Multiple Test Scripts** dialog box will open.
2. Select the scripts in your rulebase that you would like to run. Click **Run Test Scripts**. The selected test scripts will run and the Test Report will be displayed.

NOTE: You should re-run your test script/s whenever the rulebase changes to guarantee that the results are still correct.

View the test results

After a test script has run a tab will open in the top right hand pane in Oracle Policy Modeling which shows the Test Report.

An example of a Test Report for an individual test script is shown below. The report contains two sections: a summary of the report and the test case comparison results. There will also be an additional section for Errors if any are encountered during the running of the script.

Test cases that pass are highlighted in green and test cases that fail are highlighted in red.

## Test Report



Test Script Result Detail - Tests



Save

### Regression Tester Report

Generated 28/05/2012 9:04 AM

#### Summary

Test cases		Attributes	
<b>Number of test cases:</b>	2	<b>Number of outcomes:</b>	10 (10 significant)
<b>Test cases passed:</b>	1 (50%)	<b>Matching outcomes:</b>	9 (90% of total). Of these, 9 (100 % of matching) matched exactly and 0 (0 % of matching) matched after allowance for defined thresholds.
<b>Test cases failed:</b>	1 (50%)	<b>Different outcomes:</b>	1 (10%)
<b>Test cases ignored:</b>	0 (0%)	<b>Ignored outcomes:</b>	0 (0%)
<b>Test cases with errors:</b>	0 (0%)		

#### Test case comparison results

Cases	Entities	Outcomes	Expected	Actual
all true 2 children Fail (1 out of 5 items)	global [global 1]	eligible_low_income_allowance (global)	True	True
		low_income_allowance_payment (global)	80.0	70.0
		eligible_teenage_allowance (global)	True	True
	child[child 1]	(child)	True	True
	child[child 2]	(child)	False	False
all false 2 children Pass (5 exact, 0 threshold and 0 ignored)	global [global 1]	eligible_low_income_allowance (global)	False	False
		low_income_allowance_payment (global)	0.0	0.0
		eligible_teenage_allowance (global)	False	False
	child[child 1]	(child)	False	False
	child[child 2]	(child)	False	False

If you have selected multiple test scripts to be run, the Test Report will open to a Test Script Result Summary. This show the Total Statistics for all the test scripts at the top of the report, and individual reports can be viewed by clicking on the links below this.

## Test Report



Back

Test Script Result Summary

### Regression Tester Report

Generated 28/05/2012 9:09 AM

#### Total Statistics

##### Test Scripts

**Number of test scripts:** 4

**Test scripts passed:** 4 (100%)

**Test scripts failed:** 0 (%)

Test cases		Attributes	
<b>Number of test cases:</b>	152	<b>Number of outcomes:</b>	1004 ( <b>1004</b> significant)
<b>Test cases passed:</b>	152 (100%)	<b>Matching outcomes:</b>	<b>1004</b> (100% of total). Of these, <b>1004</b> (100 % of matching) matched exactly and <b>0</b> (0 % of matching) matched after allowance for defined thresholds.
<b>Test cases failed:</b>	0 (0%)	<b>Different outcomes:</b>	0 (0%)
<b>Test cases ignored:</b>	0 (0%)	<b>Ignored outcomes:</b>	0 (0%)
<b>Test cases with errors:</b>	0 (0%)		

AllBenefits

Passed (

LIHEAP

Passed (

SNAP

Passed (

TANF

Passed (

To navigate from individual reports back to the summary view, you click the **Back** button at the top left of the Test Report tab.

Customize the test report

Reports can be customized by changing the report options in **File | Project Properties | Regression Tester Properties | Report Options**.

**Regression Tester - Report Properties**

**Report Type**

Sequential HTML Layout

Tabular HTML layout

Use Custom XSLT Template:

**Report heading styles**

Outcome ID only

Outcome Display Text Only

Both Outcome ID and Display Text

**Omit from report**

Values that Match

Test cases that pass

The options in this dialog box are explained below:

Setting	Options
Report type	<p>The Test Report can be rendered in two distinct layouts – sequential or tabular.</p> <ul style="list-style-type: none"> <li>the sequential layout lists results for cases down the page</li> <li>the tabular layout presents results in a grid (cases rows and attribute columns)</li> </ul> <p>Alternatively, you can specify a custom XSLT template for the regression tester to use when generating the Test Report.</p>
Report heading styles	<p>There are three options for report headings:</p> <ul style="list-style-type: none"> <li>Outcome ID only - this will cause reports to display with attribute IDs (either model ID or public name) as headings.</li> <li>Outcome display text only - this will cause reports to display with the value of the Display Text specified for the attribute in the outcome set.</li> <li>Both outcome ID and display text - will display both the attribute IDs and the Display Text.</li> </ul>
Omit from report	<p>You have the option to omit from the Test Report:</p> <ul style="list-style-type: none"> <li>Values that match - this excludes attributes with outcome values that match the test case, and/or</li> <li>Test cases that pass - this excludes test cases that have passed.</li> </ul>

### Save the test report

You can save a test report by clicking the **Save** button at the top right of the Test Report tab.

If the Test Report is for an individual test script, the report will be saved as a HTML file.

If the Test Report contains multiple test script reports, you have the option to save the report in XML or HTML format. You need to specify a folder where the summary and individual report files will be saved to.

## Debug a failing test case

In the tester report which is generated when you run your test scripts, failed test cases are highlighted in red.

To debug a failed test case you should:

- check that the input data is correct (open the test case to view and/or update the base level attribute values)
- check that the expected results are correct (in the Test Report, click on the expected result to view and/or update the expected results for that attribute)
- check whether outcomes are set correctly (in the Test Report, click on the outcome to view and/or update the outcome for the test script)
- check the decision report (in the Test Report, click on the actual result to view the decision report for that attribute)

See also

- [Find the cause of a logic error](#)

## Create test cases with temporal data or outcomes

To create test cases with temporal data or outcomes you need to specify change points for the attributes.

### What do you want to do?

[Create change points in input data](#)

[Create change points in expected results](#)

[Create change points in outcome data](#)

### Create change points in input data

To create change points in input data:

1. In Oracle Policy Modeling, open your test script file by double-clicking it in the Project Explorer.
2. Double-click the test case to open it for editing.
3. Double click the base-level attribute in the Data view to open the **Set Attribute Value** dialog box.
4. Click the **Change Points** button. This expands the dialog box so that you can add change points for the attribute.
5. Click the **Add** button to add a new change point. A change point will be added. (By default this will have today's date and a value of unknown.)
6. From the **Date** field, select the desired date (or type a new date). Then select the check box for the **Value** that applies from that date.
7. To add additional change points, repeat steps 5 and 6.
8. When you have created all the change points, click **OK**. In the Data view you can now see the values you set for the attribute.

### Create change points in expected results

To create change points in expected results:

1. In Oracle Policy Modeling, open your test script file by double-clicking it in the Project Explorer.
2. Double-click the test case to open it for editing.
3. Select the inferred attribute in the Data view, right-click and select **Set Value**.
4. In the **Set Attribute Value** dialog box, click the **Change Points** button. This expands the dialog box so that you can add change points for the attribute.
5. Click the **Add** button to add a new change point. A change point will be added. (By default this will have today's date and a value of unknown.)
6. From the **Date** field, select the desired date (or type a new date). Then select the check box for the **Value** that applies from that date.
7. To add additional change points, repeat steps 5 and 6.
8. When you have created all the change points, click **OK**. In the Data view you can now see the values you set for the attribute.

### Create change points in outcome data

To create change points in outcome data:

1. In Oracle Policy Modeling, open your test script file by double-clicking it in the Project Explorer.
2. Select the **Outcomes** tab.
3. Double-click on the outcome attribute to open the **Edit Outcome** dialog box.
4. Click the **Change Points** button. This expands the dialog box so that you can add change points for the attribute.
5. Click the **Add** button to add a new change point. A change point will be added. (By default this will have today's date and a value of unknown.)
6. From the **Date** field, select the desired date (or type a new date). Then select the check box for the **Value** that applies from that date.
7. To add additional change points, repeat steps 5 and 6.
8. When you have created all the change points, click **OK**. In the outcome editor you can now see the values you set for the attribute.

### Measure the coverage of a test suite

It is important to ensure that a test suite has a high level of coverage of the rules in that rulebase. The Test Script Coverage report is used to measure the amount of coverage that an existing test suite provides by charting all of the logical paths through the rulebase and ensuring that they are all taken at some point. With this information you can then improve the quality of your test cases.

### What do you want to do?

[Generate a Test Script Coverage report](#)

[View the coverage for a rule](#)

[Change the coverage threshold](#)

[Change the goals used in the analysis](#)

[Save the coverage report as an XML file](#)

[Understand how coverage is measured](#)

Analyze an existing coverage file

Change the platform used by the analyzer

Analyze test script coverage using the command line tools

Generate a Test Script Coverage report

To generate a Test Script Coverage report, go to **Reports | Test Script Coverage**. The test scripts in the project will be analyzed and the report will be displayed in the right hand pane.

The percentage attached at each level of the report shows the percentage of conditions at that level which have relevant values. See [Understand how coverage is measured](#) for more information on this.

**Test Script Coverage**

Organize By Document Coverage Threshold: 80 % Analysis based on 16 goals

SocialServicesScreening

- Rules - (1044/3995)
  - Source - (881/3426)
    - Supplemental Nutrition Assistance Program.doc - (41/43)
    - Temporary Assistance for Needy Families.doc - (45/45)
    - Low Income Home Energy Assistance Program.doc - (10/10)
    - School Meals.doc - (14/18)
    - Women Infants Children.doc - (11/11)
    - Earned Income Tax Credit.doc - (44/45)
    - Child and Dependent Care Credits.doc - (20/20)
    - Child Care Assistance.doc - (32/32)
    - Thresholds.xls - (228/686)
      - the maximum monthly allotment for SNAP for the household size - (12/18)
      - the SNAP monthly utility allowance - (26/226)
      - the TANF gross monthly income standard for the household size - (9/42)
      - the TANF monthly benefit standard - (9/42)
      - the TANF monthly payment standard - (9/42)
      - the LIHEAP income threshold for the household size - (68/102)
      - the annual EITC income limit for the household size - (27/32)
      - the percentage of child and dependent care expenses which the applicant can claim for the Federal Child and Dependent Care Credit - (5/102)
      - the percentage of the Federal Child and Dependent Care Credit which the applicant can claim for the applicant's State Child and Depend...
      - the 2-letter abbreviation for the State - (0/102)
    - Poverty Guidelines.xls - (270/1524)
    - State Median Income.xls - (166/992)

### View the report organized by document

By default, the report is organized by document. Colored icons give a visual indication of the coverage of each rule:

- a green 'tick' icon indicates a rule that has adequate coverage at the specified threshold
- a red 'arrow' icon indicates a rule that has inadequate coverage at the specified threshold
- a yellow 'arrow' icon indicates a document/folder that contains a rule that does not meet the threshold

### View the report organized by goal

Alternatively, you can view the report organized by goal. To do this, select **Organize By Goal** in the drop down box at the top of the report.

**Test Script Coverage**

Organize By Goal Coverage Threshold: 80 % Save Regenerate

- [-] SocialServicesScreening
  - [+] ✓ the applicant may be eligible for Child and Dependent Care Credit - (29/29)
  - [+] ⚠ the applicant may be eligible for child care assistance - (67/266)
  - [+] ✓ the applicant may be eligible for EITC - (70/76)
  - [+] ⚠ the children in the household may be eligible for free school meals - (34/229)
  - [+] ⚠ the children in the household may be eligible for reduced price school meals - (32/229)
  - [+] ✓ the estimated amount of the Federal Child and Dependent Care Credit - (82/92)
  - [+] ✓ the estimated amount of the State Child and Dependent Care Credit - (100/112)
  - [-] ⚠ the household may be eligible for SNAP - (427/941)
    - [-] ⚠ the total gross income minus monthly child support is less than or equal to 130% of the poverty level for the household size - (120/216)
      - [+] ✓ the household's total gross monthly income - (3/3)
      - [+] ✓ the household's gross monthly earned income - (1/1)
      - [+] ✓ the household's gross monthly unearned income - (1/1)
      - [-] ⚠ 130% of the monthly poverty level for the household size - (115/211)
        - [-] ⚠ the household's geographical area for the purpose of the Poverty Guidelines - (55/102)
        - [+] ✓ the number of people in the household - (1/1)
    - [+] ✓ the household includes a household member who is aged 60 years or older or who is disabled or blind - (14/14)
    - [+] ⚠ the household's total gross monthly income is equal to or below 200% of the poverty level for the household size - (116/214)
    - [+] ⚠ the household's monthly SNAP net income is equal to or below 100% of the poverty level - (163/483)
  - [+] ⚠ the household may be eligible for TANF - (79/161)
  - [+] ⚠ the household may be eligible for the LIHEAP Funded Cooling Component - (1034/2918)
  - [+] ⚠ the household may be eligible for the Low Income Home Energy Assistance Program - (1101/2914)
  - [+] ⚠ the household's estimated monthly allotment for SNAP - (445/963)
  - [+] ⚠ the household's estimated monthly benefit amount for TANF - (94/209)
  - [+] ✓ the maximum amount of the Federal Earned Income Credit - (87/93)
  - [+] ⚠ the maximum amount of the State Earned Income Credit - (93/105)
  - [+] ⚠ there is a household member who may be eligible for WIC - (59/259)

The report is broken down by each goal mentioned in any test case. Selecting any attribute in the report will show the rule that proves that attribute. Colored icons give a visual indication of the coverage of each rule:

- a green 'tick' icon indicates a rule that has adequate coverage
- a red 'arrow' icon indicates a rule that has inadequate coverage
- a yellow 'arrow' icon indicates a rule that is itself adequately covered but contains a rule that is not.

#### View the coverage for a rule

When you click on a rule in the report, the bottom pane shows the rule and how the coverage has been determined:

**Test Script Coverage**

Organize By Document Coverage Threshold: 80 % Analysis based on 16 goals Save Regenerate

- SocialServicesScreening
  - Rules - (1044/3995)
    - Source - (881/3426)
      - Supplemental Nutrition Assistance Program.doc - (41/43)
      - Temporary Assistance for Needy Families.doc - (45/45)
      - Low Income Home Energy Assistance Program.doc - (10/10)
      - School Meals.doc - (14/18)
      - Women Infants Children.doc - (11/11)
      - Earned Income Tax Credit.doc - (44/45)
      - Child and Dependent Care Credits.doc - (20/20)
      - Child Care Assistance.doc - (32/32)
      - Thresholds.xls - (228/686)
        - the maximum monthly allotment for SNAP for the household size - (12/18)
        - the SNAP monthly utility allowance - (26/226)
        - the TANF gross monthly income standard for the household size - (9/42)
        - the TANF monthly benefit standard - (9/42)

Coverage for this rule: 12/18

```

the maximum monthly allotment for SNAP for the household size
200
true false the number of people in the household = 1
367
true false the number of people in the household = 2
526
true false the number of people in the household = 3
668
true false the number of people in the household = 4
793
true false the number of people in the household = 5
952
true false the number of people in the household = 6
1052
true false the number of people in the household = 7
1202
true false the number of people in the household = 8
1202 + 150 * (the number of people in the household - 8)
true false the number of people in the household >= 9
0
otherwise
  
```

Next to each condition in the rule is 'true' and 'false' - these values are shown in bold when that value is covered by the test suite (and are not bolded when that value is not covered).

For rules without any conditions, if the conclusion is relevant to anything, the rule is considered 100% covered and is marked as 'used' in the rule coverage browser:

Coverage for this rule: 1/1

**used** the monthly child/dependent care costs = **Minimum**(200 \* the number of children under age 2 in the household + 175 \* the number of people in the household who are age 2 or older, the sum of the reported monthly child care and dependent care expenses)

### Change the coverage threshold

Adequate coverage is initially defined to be 80%, but this can be changed by altering the percentage in the **Coverage Threshold** field at the top of the report (and then pressing **Enter** or clicking the **Regenerate** button).

### Change the goals used in the analysis

By default, the analysis is based on all of the goals (outcomes) defined in the test script. You can remove goals used in the analysis by clicking on the **Analysis based on X goals** link at the top of the report. In the **Coverage Goals** dialog, unselect any goals that you do not want included in the analysis and then click **OK**. You will notice that all of the rules used by the goals that are no longer included in the analysis now have zero coverage.

### Save the coverage report as an XML file

To save the coverage report as an XML file (so that it can be opened outside Oracle Policy Modeling), click on the **Save** button at the top of the report.

### Understand how coverage is measured

Coverage is measured by:

1. Taking every condition that appears in a rule (for example "the person's income < \$100,000" is a condition), and
2. Running all test cases, and
3. Testing the relevancy of every condition in the rulebase with respect to all nominated outcomes of each test case:
  - i. If a condition was never relevant to any nominated outcome then it is not covered at all.
  - ii. If a condition was relevant on at least one occasion, but only ever with a true value then it has partial coverage.
  - iii. If a condition was relevant on at least one occasion, but only ever with a false value then it has partial coverage.
  - iv. If a condition is relevant with both true and false values in at least one occasion, then the condition has full coverage.

NOTE: Relevancy means that it would have appeared in a decision report. For more information, see [Definition of 'relevant' in decision reports](#).

It is important to note that attributes are not the basis for this report, and are in fact not even recognized by the test script analyzer. The analyzer only cares that at some point a condition was true and on another occasion it was false.

## Analyze an existing coverage file

Every time you analyze coverage of a project's test cases, Oracle Policy Modeling automatically produces a \*.coverage file for the master project (and any modules it uses) which you can reopen later. Coverage files that have been generated [using the batch processor](#) can also be opened and analyzed in Oracle Policy Modeling.

To analyze an existing coverage file:

1. Go to **Reports | Analyze Coverage File...**
2. Choose a .coverage file to analyze, then click **Open**.

The coverage report will be displayed in the right hand pane.

## Coverage of projects that include modules

The coverage data includes the data about the rulebase plus any modules it uses, all in the same file. This means you can analyze that same file in the main project, or you can open up one of the modules that was used and analyze it there. You will still only see the rules that belong to that project, but effectively this lets you see which rules in your module are being used in the master rulebase. Since the module doesn't recognize any of the attributes from the master rulebase, the goals will be displayed with only their name, not their text, but the analysis is basically the same.

## Change the platform used by the analyzer

There is both a .NET and Java coverage analyzer, so it will work if you have .NET or Java [custom functions](#). To change the platform used by the analyzer (the default is .NET), see [Change the platform that the regression tester runs on](#).

## Analyze test script coverage using the command line tools

To build a rulebase and analyze its test script coverage using the command line tools, follow these steps:

1. Use the [command line build tool](#) to build the project.
2. Use the [command line regression tester tool](#) to run tests and produce a \*.coverage file.
3. Use the command line build tool a second time to transform the \*.coverage file into a \*.xml coverage report.

See also:

- [Analyze the outcomes of a large number of test cases](#)
- [Improve test script coverage](#)

## Improve test script coverage

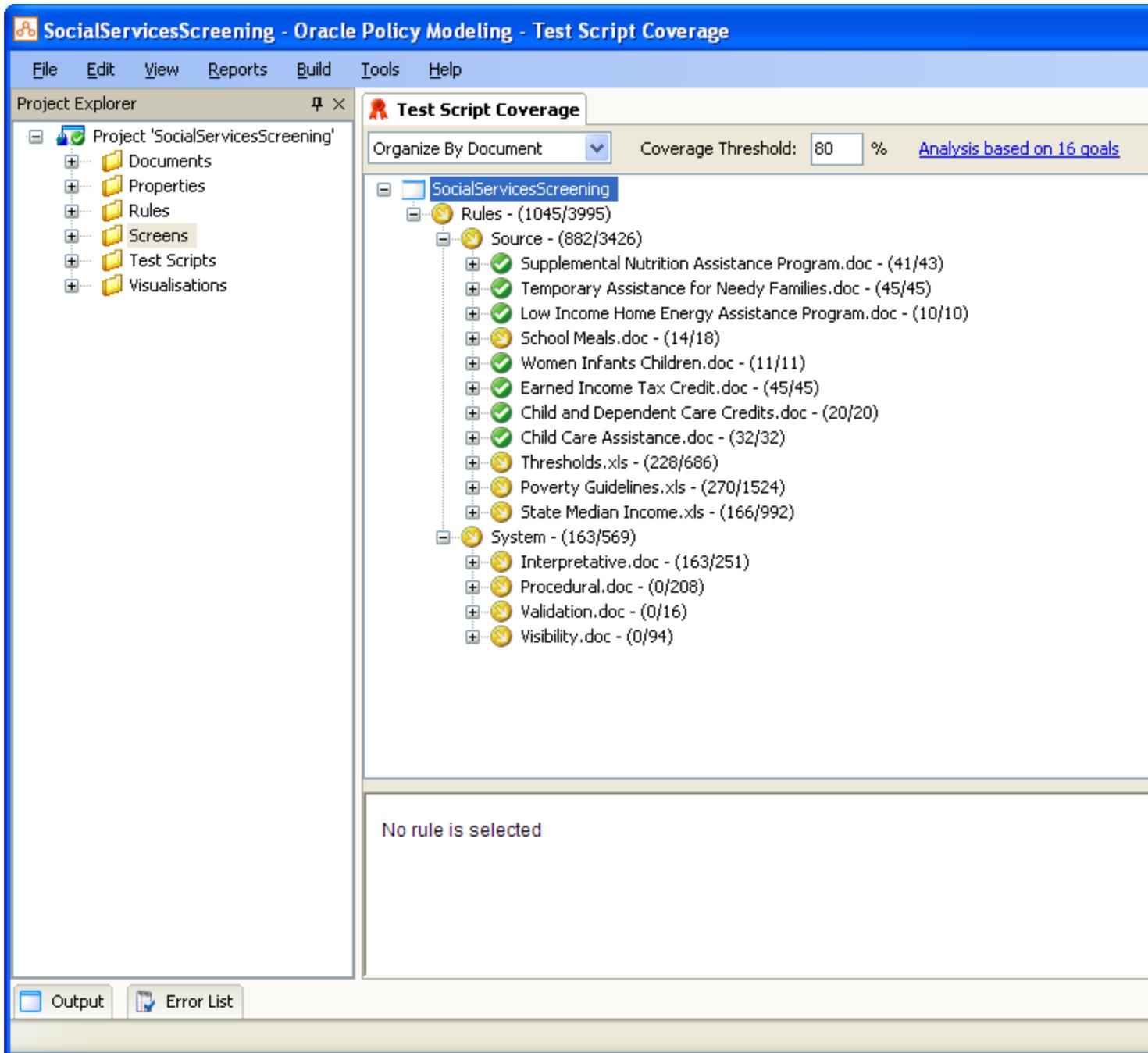
For a suite of test cases to provide any confidence in the correct workings of a rulebase, it is important that the majority of the rules in the project are meaningfully exercised by test cases.

For example, if a rulebase calculates an income threshold, but there is no test case that fails to meet that threshold, then a change to that threshold (for example, by inadvertently bypassing it) may not be detected until the rulebase is put into production. Only by exercising the threshold in both the positive and negative situation (by having test cases that meet the threshold and test cases that do not meet the threshold) can there be any confidence that a change to the rules has not had unintended side effects.

The [Test Script Coverage report](#) in Oracle Policy Modeling can be used to find areas of a rulebase that are not well-exercised. The next step is to construct a plausible test case that does exercise those areas.

The following example walks through this process:

1. Open the Social Services Screening example project.
2. Generate a Test Script Coverage report by going to **Reports | Test Script Coverage**.

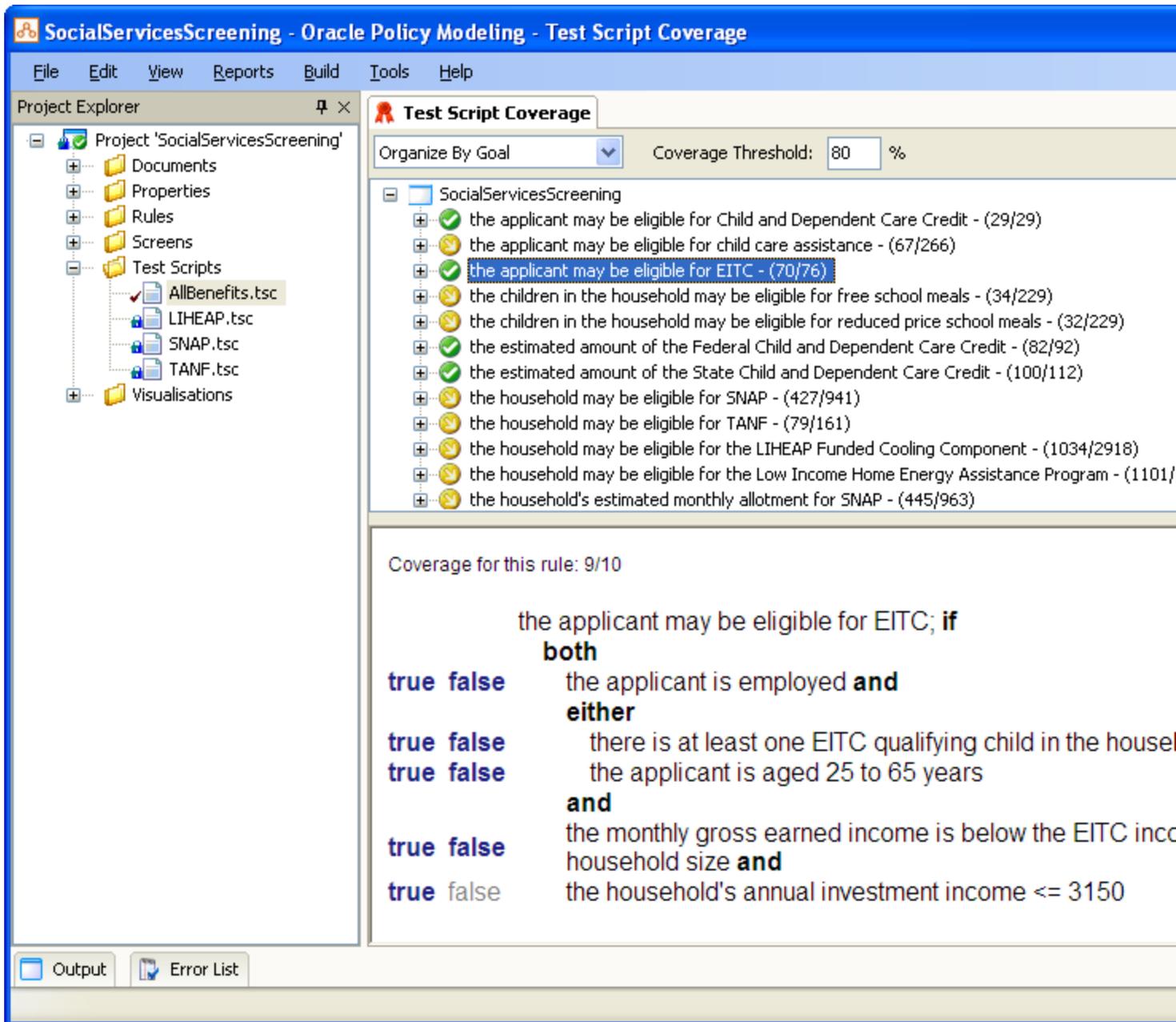


By default, this report will organize the report by document, which allows you to see how well tested the rules are on a per-document basis. The colored icons represent whether a document contains any rules that do not meet the default coverage threshold of 80%.

- Green tick: Indicates the rules meet the threshold.
- Red arrow: Used for a rule that does not meet the threshold.
- Yellow arrow: Used when a document/folder contains a rule that does not meet the threshold.

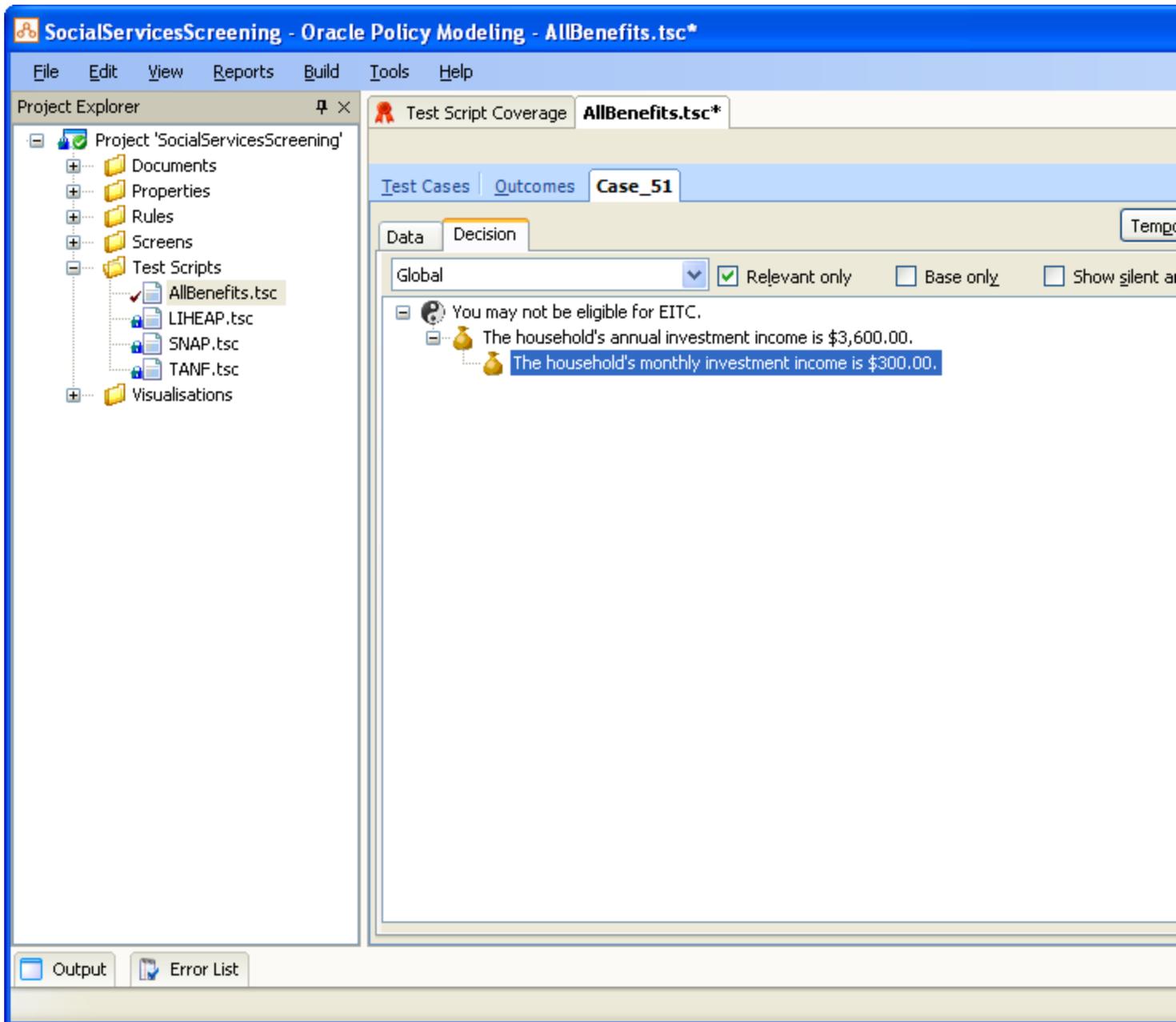
Note that the System rules mostly have no coverage at all. This is because these rules are mostly only used during a guided interactive interview, which is not verified by the test cases. The rules in those documents may, of course, be triggered when the test cases run, but their results are never compared with expected values, so the rules have no coverage.

3. Change **Organize By Document** to **Organize By Goal**. This shows all of the top-level goals that are tested in all the test scripts of the project, and shows how well-tested the rules of each goal are. In general, it is most important that top level rules are well covered, as these tend to indicate broad areas of functionality within the rulebase that should absolutely be verified. Lower level rules need coverage also, but exercising all combinations may not be practical for large tables whose only purpose is to reinterpret base-data provided by the user. An example of this can be seen by following the yellow coverage icons from "the applicant may be eligible for child care assistance" down to the rule "the household's geographical area for the purpose of the Poverty Guidelines" which just maps 50 possible locations into one of three broad categories. Testing all 102 possibilities of this rule would not be as useful as ensuring that all the high level eligibility conditions are tested.
4. Select "the applicant may be eligible for EITC", which displays the high-level rule for this attribute.



The bold 'true' and 'false' values indicate that most conditions in the rule have been tested in both a true situation and a false situation. The rule meets the coverage threshold by having 9/10 coverage, however the final condition "the household's annual investment income <= 3150" has only been applied when it was 'true'.

- To fix this, open the AllBenefits.tsc test script file and make a copy of Case\_50, call it Case\_51 and then open it. Change the value of the attribute "the household's monthly investment income" from 0 to 300. This will change the yearly investment income to \$3,600, and the household will no longer be eligible for EITC. The expected results in this test case will need to be updated for the test to pass, but this can be done later.



6. Return to the Test Script Coverage report and click the **Regenerate** button. The coverage for the rule "the applicant may be eligible for EITC" is now complete.

This example was simple because the change to the test case was immediately significant to the conclusion, therefore the change contributed immediately to the coverage score. Other changes may be more subtle.

For example:

1. In the Test Script Coverage report, expand "the applicant may be eligible for EITC", then "the monthly gross earned income is below the EITC income limit for the household size" then "the annual EITC income limit for the household size".

SocialServicesScreening - Oracle Policy Modeling - Test Script Coverage

File Edit View Reports Build Tools Help

Project Explorer

- Project 'SocialServicesScreening'
  - Documents
  - Properties
  - Rules
  - Screens
  - Test Scripts
    - AllBenefits.tsc
    - LIHEAP.tsc
    - SNAP.tsc
    - TANF.tsc
  - Visualisations

Test Script Coverage AllBenefits.tsc\*

Organize By Goal Coverage Threshold: 80 %

- SocialServicesScreening
  - the applicant may be eligible for Child and Dependent Care Credit - (29/29)
  - the applicant may be eligible for child care assistance - (67/266)
  - the applicant may be eligible for EITC - (70/76)
    - the applicant is employed - (2/2)
    - there is at least one EITC qualifying child in the household - (19/19)
    - the applicant is aged 25 to 65 years - (4/4)
    - the monthly gross earned income is below the EITC income limit for the household size - (1/1)
    - the household's gross monthly earned income - (1/1)
    - the annual EITC income limit for the household size - (32/37)
    - the household's annual investment income - (1/1)

Coverage for this rule: 27/32

	the annual EITC income limit for the household size	50270
	<b>both</b>	
true false	the number of EITC qualifying children in the household	
true false	the applicant is married	45060
	<b>both</b>	
true false	the number of EITC qualifying children in the household	
true false	~the applicant is not married	47162
	<b>both</b>	
true false	the number of EITC qualifying children in the household	
true false	the applicant is married	41952
	<b>both</b>	
true false	the number of EITC qualifying children in the household	
true false	~the applicant is not married	42130
	<b>both</b>	
true false	the number of EITC qualifying children in the household	

Output Error List

This rule is mostly tested, however, there is no test case for an unmarried applicant with 2 EITC qualifying children. The conditions for that situation are tested when they are false (in cases when a later row in the table is triggered instead) but

never when they are true. We can also deduce from this that the income limit for that situation is not being tested.

The screenshot shows the Oracle Policy Modeling Test Script Coverage interface. The main window title is "SocialServicesScreening - Oracle Policy Modeling - Test Script Coverage". The menu bar includes File, Edit, View, Reports, Build, Tools, and Help. The Project Explorer on the left shows a tree structure for "Project 'SocialServicesScreening'" with folders for Documents, Properties, Rules, Screens, Test Scripts, and Visualisations. Under Test Scripts, "AllBenefits.tsc" is selected. The main area displays "Test Script Coverage" for "AllBenefits.tsc\*" with a coverage threshold of 80%. A tree view shows various goals, with "the annual EITC income limit for the household size - (32/37)" highlighted. Below this, a detailed view shows the coverage for this rule as 27/32. The detailed view lists several test cases with their outcomes and the number of children in the household. A red oval highlights the test case where the number of children is 42130 and the applicant is not married.

Project Explorer

- Project 'SocialServicesScreening'
  - Documents
  - Properties
  - Rules
  - Screens
  - Test Scripts
    - AllBenefits.tsc
    - LIHEAP.tsc
    - SNAP.tsc
    - TANF.tsc
  - Visualisations

Test Script Coverage AllBenefits.tsc\*

Organize By Goal Coverage Threshold: 80 %

- SocialServicesScreening
  - the applicant may be eligible for Child and Dependent Care Credit - (29/29)
  - the applicant may be eligible for child care assistance - (67/266)
  - the applicant may be eligible for EITC - (70/76)
    - the applicant is employed - (2/2)
    - there is at least one EITC qualifying child in the household - (19/19)
    - the applicant is aged 25 to 65 years - (4/4)
    - the monthly gross earned income is below the EITC income limit for the household size - (1/1)
      - the household's gross monthly earned income - (1/1)
      - the annual EITC income limit for the household size - (32/37)
      - the household's annual investment income - (1/1)

Coverage for this rule: 27/32

		the annual EITC income limit for the household size	50270	both	
true	false	the number of EITC qualifying children in the household			
true	false	the applicant is married	45060	both	
true	false	the number of EITC qualifying children in the household			
true	false	~the applicant is not married	47162	both	
true	false	the number of EITC qualifying children in the household			
true	false	the applicant is married	41952	both	
true	false	the number of EITC qualifying children in the household			
true	false	~the applicant is not married	42130	both	
true	false	the number of EITC qualifying children in the household			

Output Error List

2. Return to the AllBenefits.tsc file and copy Case\_51 to make another new case Case\_52. Open this new case. This case already has 2 EITC qualifying children, but the applicant Marge is considered to be married because her spouse Homer is in the household. Remove Homer from the household.

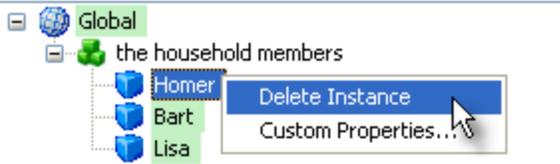
Project Explorer

- Project 'SocialServicesScreening'
  - Documents
  - Properties
  - Rules
  - Screens
  - Test Scripts
    - AllBenefits.tsc
    - LIHEAP.tsc
    - SNAP.tsc
    - TANF.tsc
  - Visualisations

Test Script Coverage AllBenefits.tsc\*

Test Cases Outcomes Case\_51 Case\_52

Data Decision



Show All

Name

Base Attributes:

hhm\_disabled

hhm\_student

hhm\_name

123 hhm\_age

hhm\_relationship

hhm\_gender

Inferable Attributes:

b20@SNAP

b22@SNAP

b6@TANF

b7@TANF

b8@TANF

b15@TANF

b11@SchoolMeals

b2@EITC

b3@EITC

b6@EITC

b6@CCDC

b7@CCDC

b8@CCDC

b4@CCA

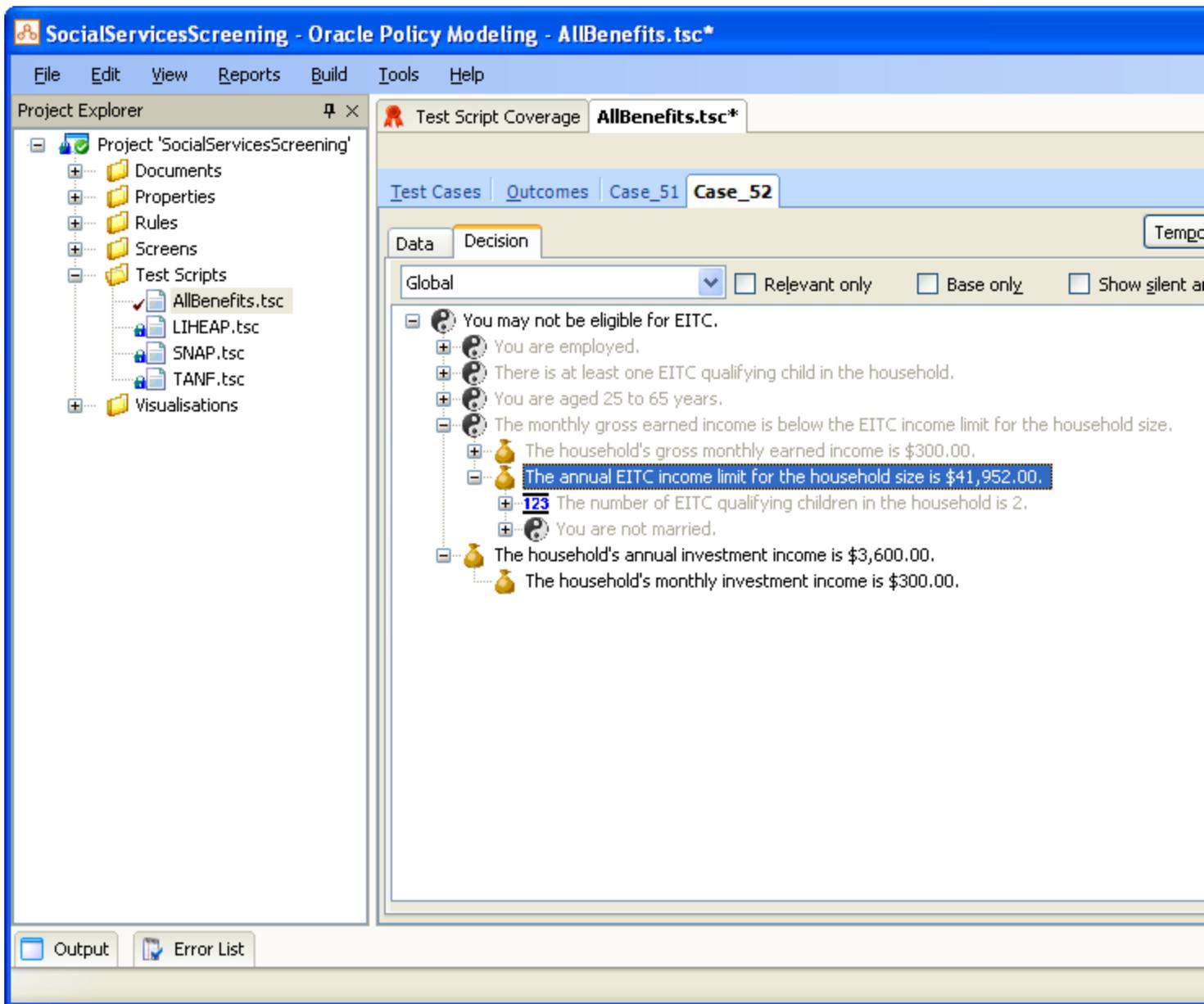
< [ ] >

Notes

Output

Error List

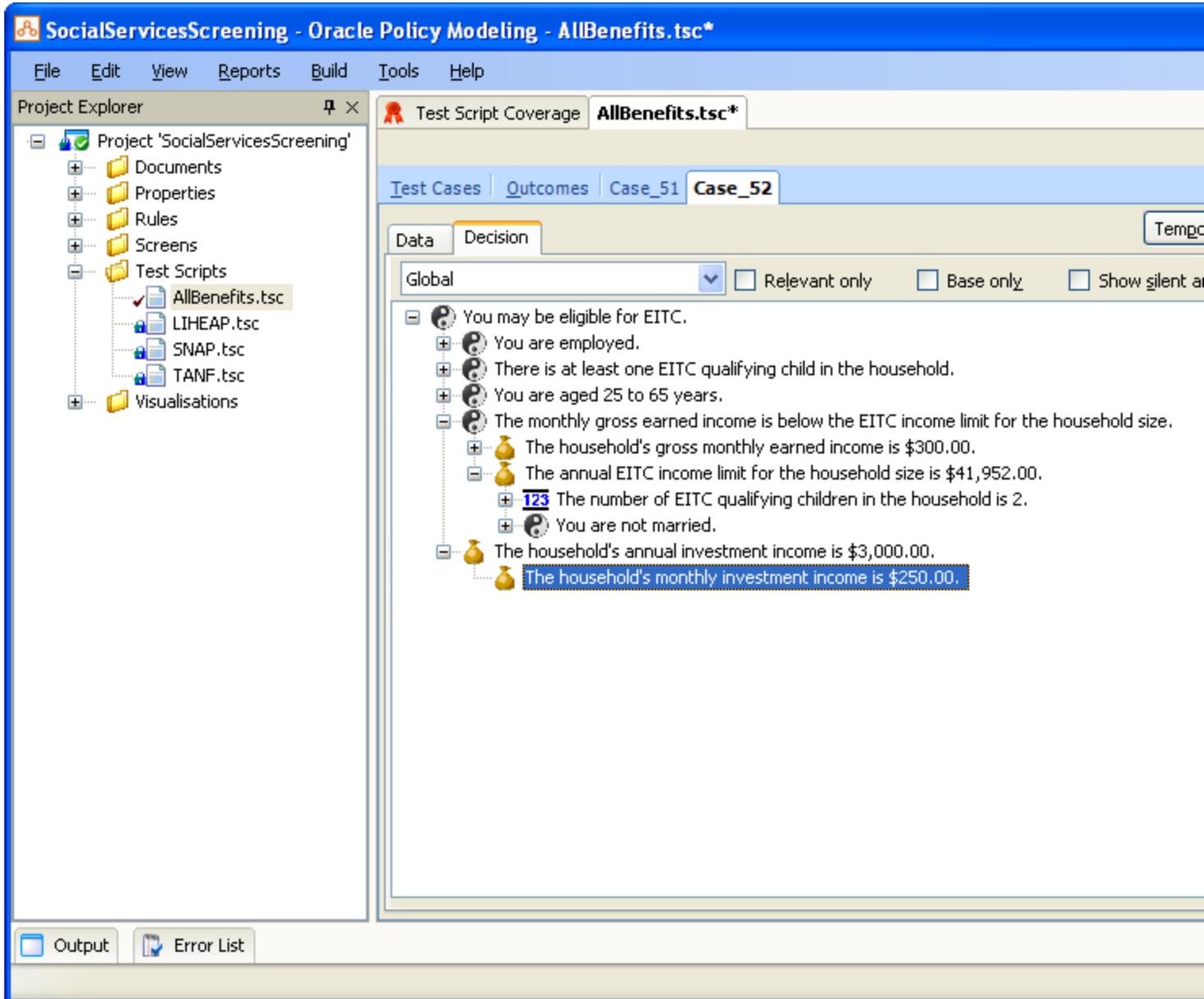
3. Regenerate the Test Script Coverage report and review the coverage for the rule for "the annual EITC income limit for the household size". Note that the coverage score has not improved, and the conditions for an unmarried applicant with 2 children is still not considered covered. To find out why, the test case must be examined more closely.
4. Return to Case\_52 and locate the attribute "You may not be eligible for EITC" (note the use of second person). Right click and choose **Show Decision**. Uncheck **Relevant Only** and expand the decision tree to find "the annual EITC income limit for the household size".



The gray values are those that not relevant to the decision. This explains why the change has not contributed to the coverage score: the change described in the previous example (which made the household not eligible for EITC) has made

the income limit irrelevant. Even though the income limit was calculated for an unmarried applicant in a 2 child household, the value was ultimately irrelevant and so those conditions are still considered to have no coverage.

5. Change the household's monthly investment income back to 250. The income limit is now relevant again.



6. Regenerate the Test Script Coverage report and see that the rule for "the annual EITC income limit for the household size" now has a higher coverage score.

SocialServicesScreening - Oracle Policy Modeling - Test Script Coverage

File Edit View Reports Build Tools Help

Project Explorer

- Project 'SocialServicesScreening'
  - Documents
  - Properties
  - Rules
  - Screens
  - Test Scripts
    - AllBenefits.tsc
    - LIHEAP.tsc
    - SNAP.tsc
    - TANF.tsc
  - Visualisations

Test Script Coverage AllBenefits.tsc\*

Organize By Goal Coverage Threshold: 80 %

- SocialServicesScreening
  - the applicant may be eligible for Child and Dependent Care Credit - (29/29)
  - the applicant may be eligible for child care assistance - (69/266)
  - the applicant may be eligible for EITC - (73/76)
    - the applicant is employed - (2/2)
    - there is at least one EITC qualifying child in the household - (19/19)
    - the applicant is aged 25 to 65 years - (4/4)
    - the monthly gross earned income is below the EITC income limit for the household size - (1/1)
    - the household's gross monthly earned income - (1/1)
    - the annual EITC income limit for the household size - (34/37)
    - the household's annual investment income - (1/1)

Coverage for this rule: 29/32

	the annual EITC income limit for the household size	50270
	both	
true false	the number of EITC qualifying children in the household	
true false	the applicant is married	45060
	both	
true false	the number of EITC qualifying children in the household	
true false	~the applicant is not married	47162
	both	
true false	the number of EITC qualifying children in the household	
true false	the applicant is married	41952
	both	
true false	the number of EITC qualifying children in the household	
true false	~the applicant is not married	42120

Output Error List

The other conditions in this rule cannot be fully covered because they use a defensive style of rule authoring and test for conditions that should never occur, or should always be true (such as testing that there are 0 children when all other numbers of children have been accounted for). For this reason, 100% coverage will rarely be possible in practice.

## Use the regression tester from the command line

The Oracle Policy Modeling Command Line Regression Tester provides a means of executing a rulebase project's text scripts using the command line.

### Syntax

This tool can be used in two different modes. The syntax for these modes is given below.

1. Oracle.Policy.Modeling.RegressionTester.CmdLine *rulebase-file testscript-file [options]*  
This is the default mode. The tool takes the supplied compiled rulebase file (.xml) and tests it using the supplied test script file (.tsc).
2. Oracle.Policy.Modeling.RegressionTester.CmdLine -project *rulebase-project-file [options]*  
Project mode. This mode takes the supplied rulebase project, and tests it using all test scripts that are associated with the project.

### Options

The following options can be used:

- --javaengine  
Indicates that the java version of the rule engine should be used to run the regression test. By default the .NET version of the rule engine is used.
- --outputfile  
Specifies that output should be written to the supplied file. By default output is written to the console.
- --verbose  
Provides verbose output. When this option is used, the result of every test case outcome is reported. By default, only those outcomes that fail are reported on.
- --xml  
Results are output in JUnit XML format.
- --coverage <coverage file>  
Generates a .coverage file that can be imported into Oracle Policy Modeling using the [Analyze Coverage File feature](#).

### Formatting

For the tool's standard (non-XML) output, the [region setting](#) for the rulebase provided is used to determine the formatting used for data types such as date, datetime and timeofday.

# Debugging

## Topics in "Debugging"

- [Debug a rulebase](#)
- [Define data to use in a test case or a debug session](#)
- [Test a portion of a rulebase](#)
- [View the attributes inferred in a test case or debug session](#)
- [Debug temporal rules and data](#)
- [Find the cause of a logic error](#)
- [Change a rule while debugging](#)
- [Save or reload a debugger session](#)

## See also:

- [Set the time period to use for calculations](#)
- [Test an interview or screen flow](#)
- [Check rule structure and dependencies](#)
- [Debug a failing test case](#)

## Debug a rulebase

When writing rules it is important that they are thoroughly tested to ensure they operate in the intended way. The Oracle Policy Modeling debugger is a tool that can be used to perform this testing function.

### What do you want to do?

[Use the integrated debugger to test the rules](#)

[Use the standalone debugger to test the rules](#)

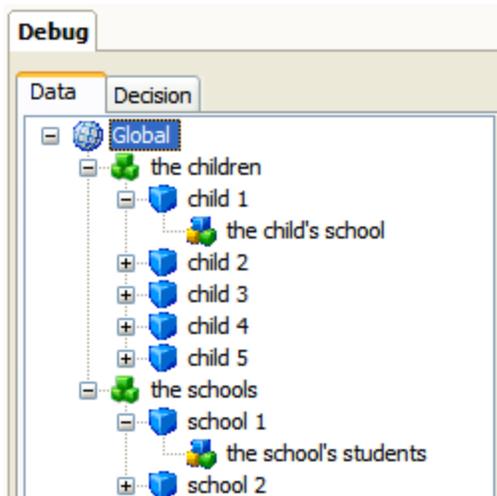
### Use the integrated debugger to test the rules

To start debugging your rules using the integrated debugger:

1. In Oracle Policy Modeling, select **Build | Build and Debug**.
2. In the **Debug Options** dialog box, select the option to debug **Without screens**. (NOTE: This option will just test the logic of your rules. Testing with screens is covered separately in [Test an interview or screen flow](#).) Click **OK**.

The debugger will open with the Debug view in the top right hand pane in Oracle Policy Modeling.

On the left hand side of the Data view is a tree view of the entity instances and relationships in the build model.



The icons used in this pane, and what they represent, are given below:

	Global entity
	Containment relationship
	Entity instance
	Reference relationship
	Inferred relationship

The right hand side of the Data view shows either an attribute list, or a relationship editor, depending on what item you have selected in the left hand pane.

You can also start a debugger session from within the Build Model. To do this:

1. In Oracle Policy Modeling, select **View | Build Model**.
2. Select the attribute that you would like to investigate, right-click and select **Investigate in Debugger**.
3. In the **Debug Options** dialog box, select the option to debug **Without screens**. Click **OK**.

The Debugger will open to the Decision view which will show all the relevant paths for a goal simultaneously.

### Using the standalone debugger to test the rules

To start debugging your rules using the standalone debugger:

1. Go to **Start | All Programs | Oracle | Tools | Oracle Policy Modeling Debugger**.  
The standalone Debugger will open.
2. Select **File | Debug Compiled Rulebase**.
3. Browse to select your compiled rulebase (XML) file (this is typically located in the output folder of your rulebase project).  
Click **Open**.

On the right hand side of the Data view in the Debugger you will see the attributes in your rulebase.

See also:

- [Define data to use in a test case or a debug session](#)
- [Test a portion of a rulebase](#)
- [View the attributes inferred in a test case or debug session](#)

## Define data to use in a test case or a debug session

In order to run a test case or debug session, you need to firstly set up the data to use.

### What do you want to do?

[Set the value for a base level attribute](#)

[Set up entities and containment relationships](#)

[Set reference relationships](#)

#### Set the value for a base level attribute

When you are investigating the inferences that are made by setting particular attribute values, you need to set values for the base level attributes directly.

To set the value of a base level attribute directly:

1. Select the attribute in the Data view. It is handy to filter the attributes list by **Base Level** attributes to ensure you are selecting a base level attribute.
2. Right-click and select from any of the following Set options from the menu:
  - Set Value** - this opens the **Set Attribute Value** dialog box where you can enter a value or set the value to 'uncertain' or 'unknown'. When setting variable values directly in the Data view, values must be entered in the correct format: see [Formatting of variable values](#). You can also specify change points for the attribute.
  - Set to True** - this option is only available for boolean attributes
  - Set to False** - this option is only available for boolean attributes
  - Set to <value>** - this option is only available for non-boolean text attributes. The values that appear here will be the values used in the rules or on screens.
  - Set to Unknown** - this option is used to clear the value of the attribute
  - Set to Uncertain**

Alternatively, you can double-click the selected attribute to open the **Set Attribute Value** dialog box and then select or set the appropriate value, ensuring that it is entered in [the correct format](#).

3. The Data view will be updated to show the new attribute value you have set, and any attribute values inferred as a result. TIP: You can sort the attributes in each grouping in the Data view by clicking on any of the column headings (Name, Value or Text).

#### Set up entities and containment relationships

If you have entities in your rulebase, you will need to create entity and relationship instances in order to investigate any rules which use those entities/relationships. For example, if you have a rulebase containing the entity "the child" and you are assessing a family with 3 children then you will need to create 3 instances of "the child" in the debugger. It is easiest to set these up before you start to investigate goals or to observe the effects of setting values for attributes.

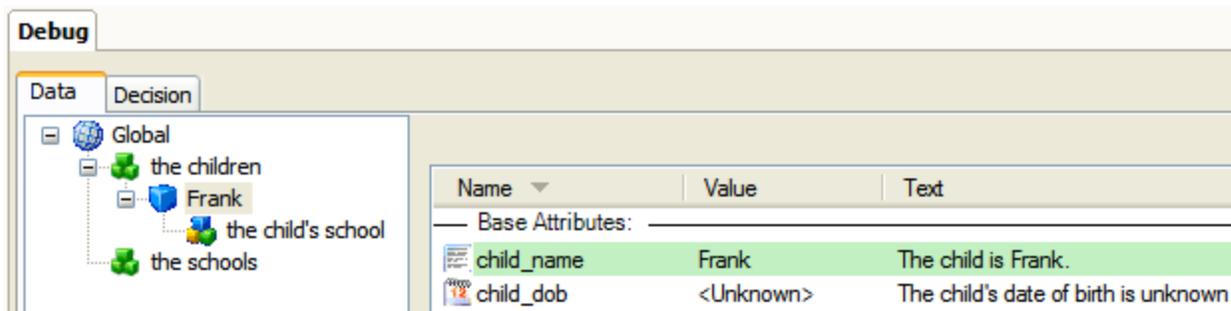
## Add entity instances

Entity instances are added via their containment relationships. To create an entity instance for an entity in the Data view:

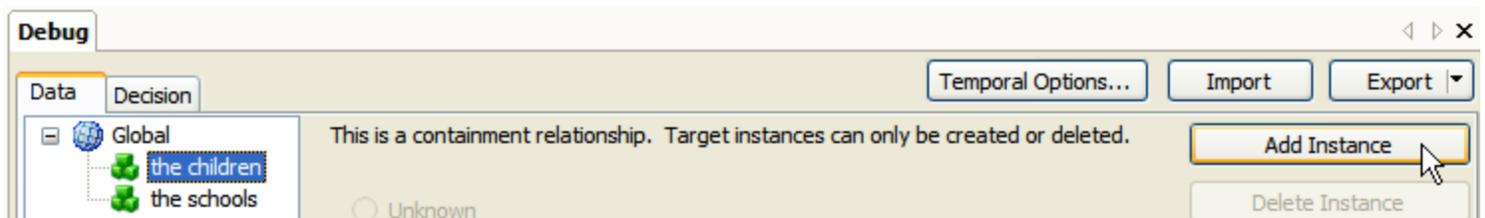
1. Select the containment relationship for the entity in the left hand pane. TIP: Containment relationships are indicated by a green multi-cube icon.
2. Right-click and select **Add Instance**:  
An entity instance (eg child 1) will appear below the containment relationship. The containing relationship for that entity has now also been set. Any containment or reference relationships that are associated with that entity are also shown under the entity instance (eg the child's school).



TIP: At this point it can be useful to provide a value for the **identifying attribute** for each of the entity instances. This will make it easier to distinguish between the entity instances when debugging. In the example above, the child's name attribute is the identifying attribute for the child entity. Following the steps above for setting the value for a base level attribute, you would set the value of the child's name attribute for each of the entity instances you have added. This value (eg Frank) then replaces the generic entity label (eg child 1) in the Data view:



You can also add a new entity instance by selecting the containment relationship and using the **Add Instance** button in the relationship editor:



For entities contained within other entities, instances are created in the same way as above, using the containment relationships within the existing instances.

### Delete entity instances

To delete an entity instance:

1. Select the entity instance in the Data view.
2. Right-click and select **Delete Instance**:

The selected entity instance will be removed from the list of entity instances for that entity.

TIP: You can also delete an entity instance by selecting the containment relationship in the Data view, and then selecting the entity instance to be deleted in the relationship editor and using the **Delete Instance** button:



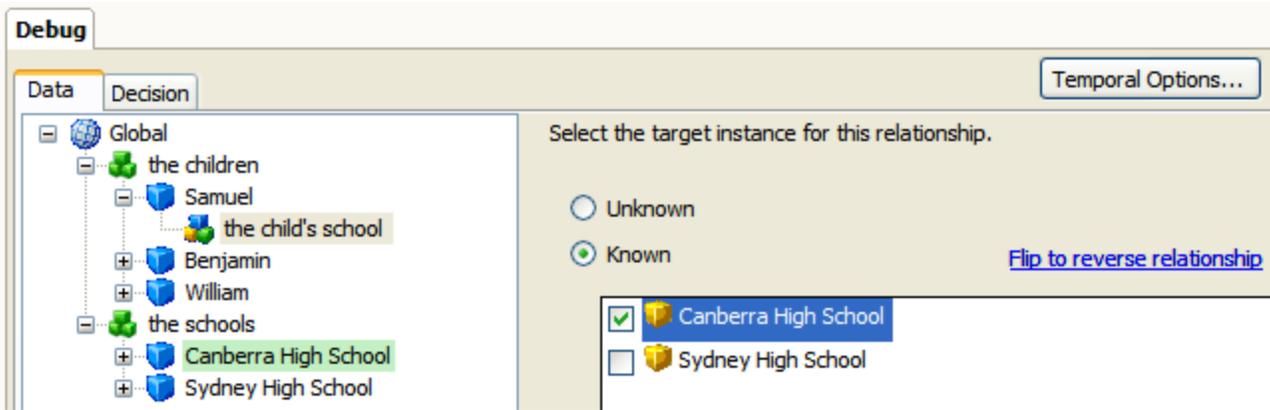
### Set reference relationships

Once [entity instances have been created](#) within their containment relationships, you may set up any reference relationships between the entity instances.

### Set reference relationships between entity instances

Reference relationships are shown underneath the entity instance in the Data view, and can be set once the relevant entity instances have been created via their containment relationships. For example, having created three children Samuel, Benjamin and William, and an instance of "the school" entity, Canberra High School, you might set that one of the children attends the school. To do this:

1. Select the relationship in the left hand pane of the Data view (in this case, "the child's school" under the child Samuel). The relationship is currently unknown.
2. In the relationship editor, check the check box for the existing entity instance Canberra High School, to set the child's school for Samuel. The relationship now becomes known.



NOTE: When you set targets for static relationships, the relationship will become known - it is not possible to leave the relationship as unknown.

### Remove the association between a target instance and the relationship

To remove the association between a target entity instance and the relationship:

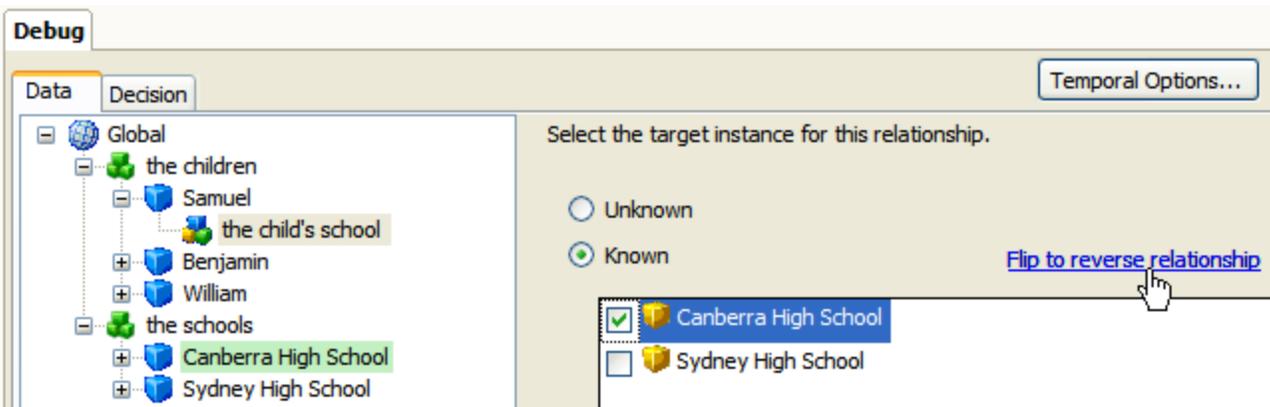
1. Select the relationship in the left hand pane of the Data view.
2. In the relationship editor in the right hand pane, deselect the check box for any entity instances that you no longer want associated with that relationship.

### Navigate reverse relationships

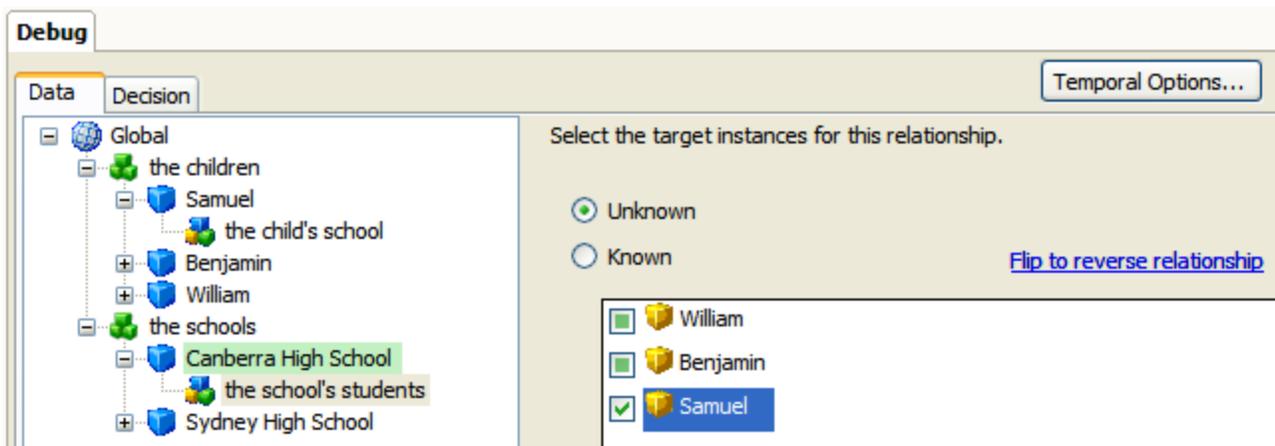
Using the relationship editor in the right hand pane of the Data view, you can switch from viewing a relationship (from the direction of the source entity) to viewing the reverse relationship (from the direction of the target entity). Note here that the 'source' and 'target' entities of a relationship are relative, and the entities referred to by these terms depend on which relationship direction is being considered.

For example, you may have a many-to-one relationship between child and school entities called 'the child's school', with a reverse relationship 'the school's students'. If you have already set the school for one child, you could easily navigate between these entity instances to view and set the reverse relationship, for other children who attend the school. To do this:

1. Click on the child's school relationship to display it in the relationship editor, then click on the **Flip to reverse relationship** link to edit the reverse relationship.



- The Data view now shows the reverse relationship 'the school's students', for the relevant instance of the school (Canberra High School). Additional child instances can now be set as targets for this relationship as appropriate.



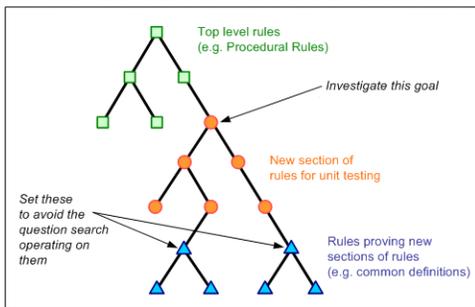
See also:

- Investigate a goal

## Test a portion of a rulebase

Unit testing the rulebase is the process of interactively testing a discrete section of rules to ensure that every sub-section operates as it should. The debugger enables you to perform targeted testing of your rules by manually setting attributes and investigating custom goals.

A section of the rules can be unit tested using the goal specific to just those rules without assessing other related or unrelated rules. This approach can be visualized in the following diagram:



In the diagram, the circles represent the attributes of the rulebase that are being tested. The top orange circle is the "custom" goal - that is, the goal chosen to be investigated for the purpose of unit testing. The squares represent the higher-level attributes that don't get tested because the goal investigated is not proved by them. The triangles represent attributes which have been "closed off" by setting them manually with the debugger.

## What do you want to do?

Test a portion of a rulebase

Test data validations

## Test a portion of a rulebase

To test a portion of the rulebase follow these steps:

1. Examine your rules and identify which branch of the rulebase you want to test. Identify the attribute which heads the branch to be assessed (the goal attribute).
2. Determine which attributes you need to set to close off the other branches of the rules. Using the debugger, [set values for these attributes](#).
3. [Investigate the goal attribute](#) by answering the required questions until a conclusion is reached for the goal.
5. Check the validity of the conclusion. Change the rules if errors are identified.
6. Go back and change the answers until all of the sub-branches have been fully tested.

A similar process is used for unit testing smaller and larger branches of the rulebase. The smaller the branch the more detailed the assessment of all the different possible combinations of sub-branches.

## Set the value for a base level attribute

To set the value of a base level attribute in the Decision view either:

- Right-click and select from any of the following Set options in the menu:
  - Set Value** - this opens the **Set Attribute Value** dialog box where you can enter a value or set the value to 'uncertain' or 'unknown'. When setting variable values directly in the Decision view, values must be entered in the correct format: see [Formatting of variable values](#) for details.
  - Set to True** - this option is only available for boolean attributes
  - Set to False** - this option is only available for boolean attributes
  - Set to <value>** - this option is only available for variables. The values that appear here will be the values used in the rules or on screens.
  - Set to Unknown** - this option is used to clear the value of the attribute
  - Set to Uncertain**
- Double-click the selected attribute to open the **Set Attribute Value** dialog box. Select or set the appropriate value, ensuring that it is entered in [the correct format](#).

After setting a value, the list of attribute values in the Decision view (and the Data view) will be updated with the value you have specified.

## Investigate a goal

After you have [set up any entities and relationships](#) in the debugger, you can investigate a goal. To do this:

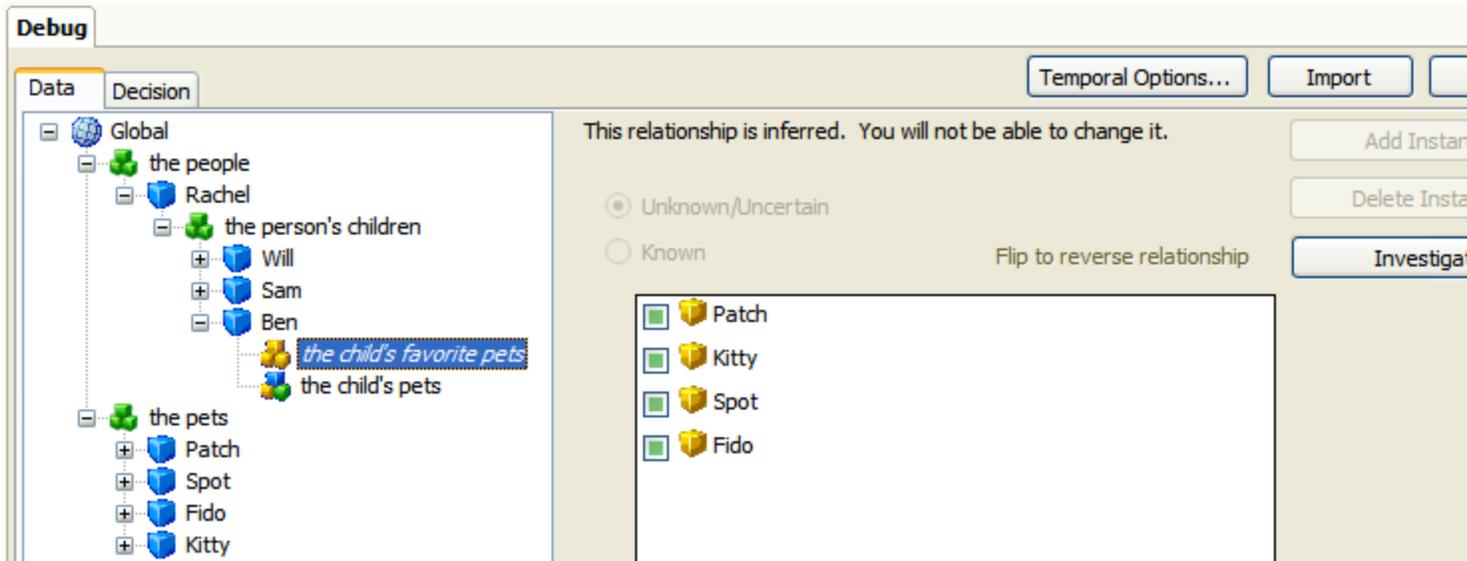
1. In the Data view select the goal you want to investigate. It is handy to filter the attributes list by **Top Level** attributes (or by **All** to see the list of Inferrable attributes) to ensure you are selecting a non-base level attribute.
2. Right-click and select **Investigate**. This will open the Decision view with the attribute you have selected as the goal. The value of this attribute is unknown and all of the relevant paths to the goal are shown in the text box below. Entities for which no instances have been created yet will be shown just by the relationship text.
3. Work your way through the list of questions, setting answers (see above). In order to investigate any attributes which belong to an entity, you will need to add instances of that entity. Add your entity instances and continue investigating attributes until a value for the goal is known.

TIP: You can filter the list of relevant attributes using [the checkboxes at the top of the Decision view](#). The **Show rule** checkbox displays the rule proving the selected attribute in the lower part of the Decision view - this can be useful to see which rule conditions are evaluating to what result to help understand how your rules are working.

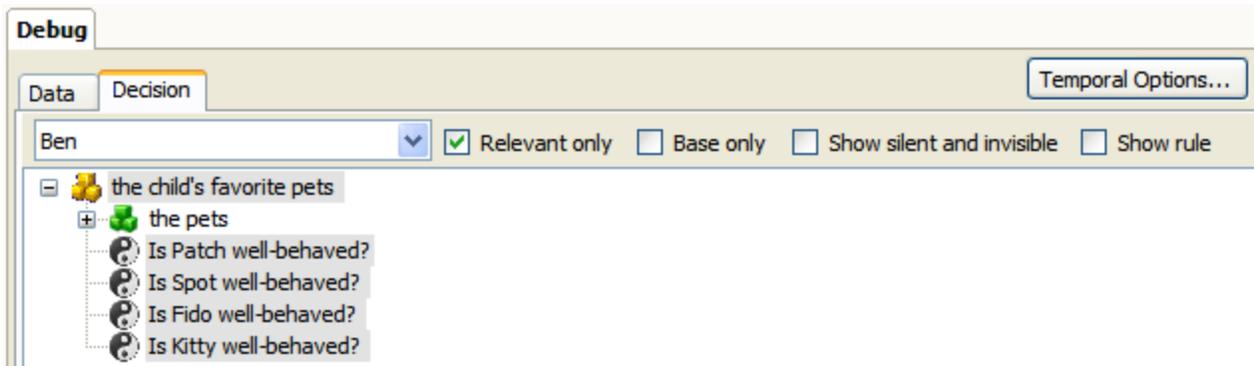
## Investigate an inferred relationship

After you have [added any entity instances](#) in the debugger, you can investigate an inferred relationship. To do this:

1. In the Data view select the inferred relationship that you want to investigate.



2. In the right hand pane, click the **Investigate** button. This will switch to the Decision view.



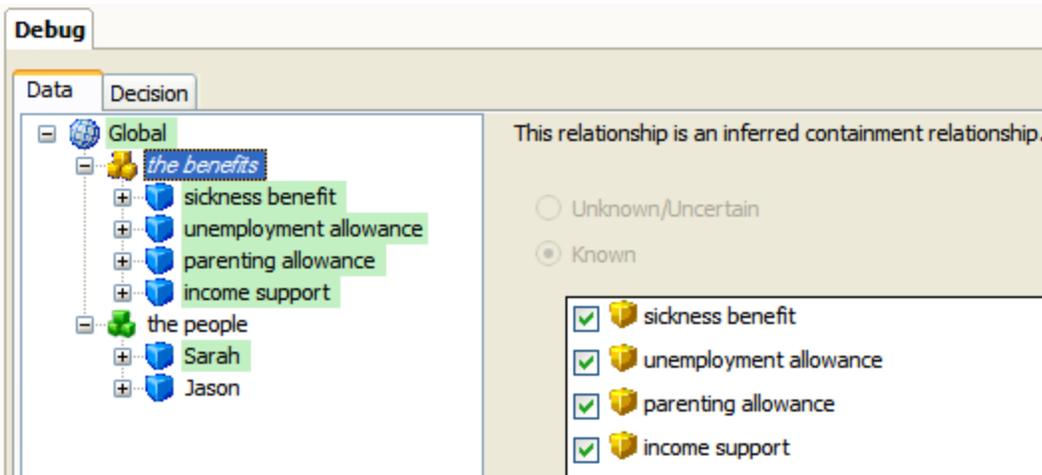
3. [Set the values for any base level attributes](#). The Decision view will be updated as you go to show which entity instances have been inferred for this relationship, and the attributes contributing to this conclusion.



In the case of existing entity instances that have been inferred as members of a relationship (ie using [IsMemberOf rules](#)), these will be shown as selected items in the right hand pane of the Data view. (These entity instances will not be shown under the inferred relationship in the left hand pane as they have not inferred a containment relationship).



In the case of entity instances that have been created as members of a relationship (ie using [InferInstance rules](#)), these are also shown in the left hand pane of the Data view under the containment relationship that they have inferred.

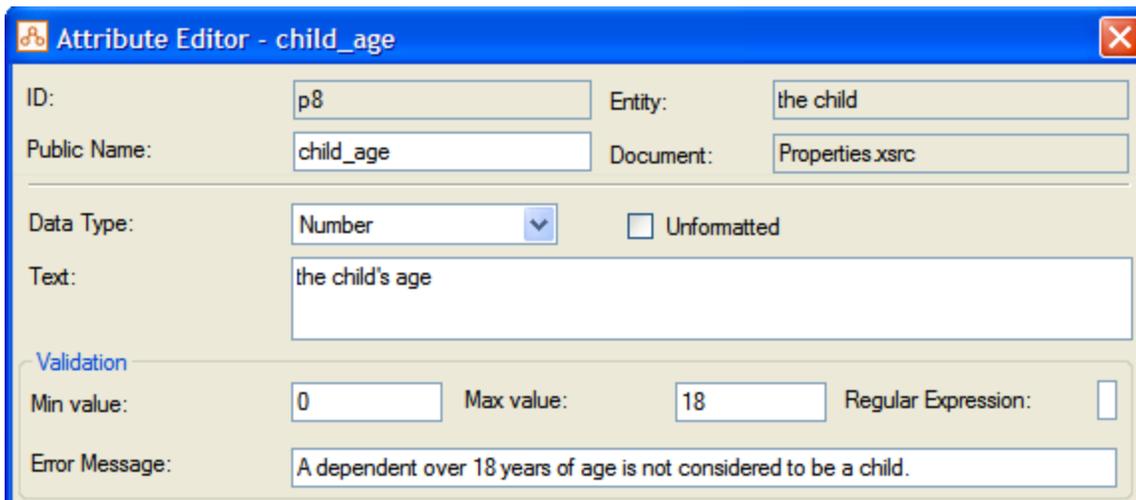


### Test data validations

The validation of data input (minimum values, maximum values, regular expressions, warnings and errors) can be tested using the debugger.

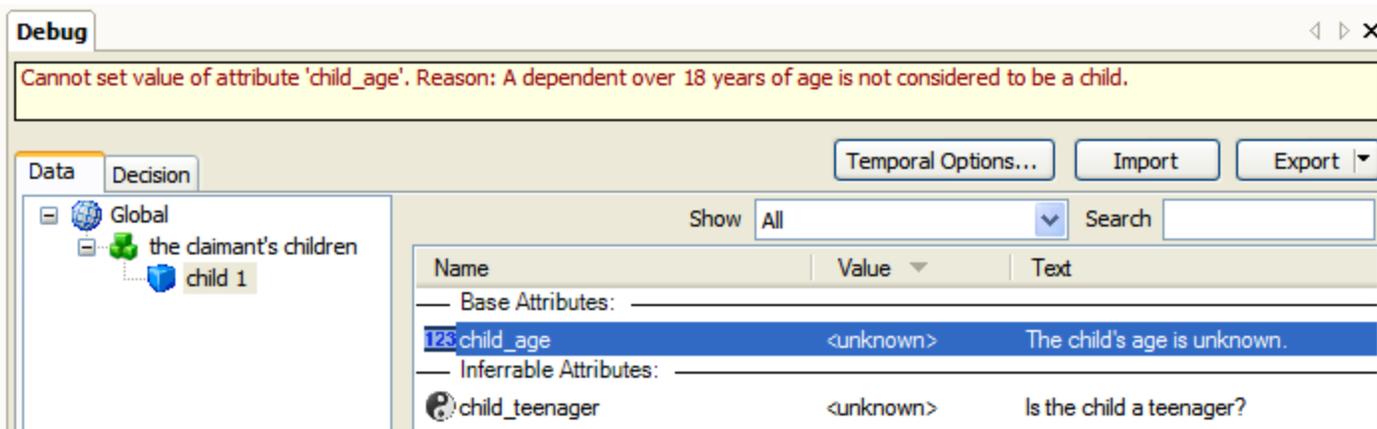
### Test minimum and maximum values

In the example below the data input (the child's age) has a **maximum value** of 18 specified in the Attribute Editor:



To check the functionality of this data validation:

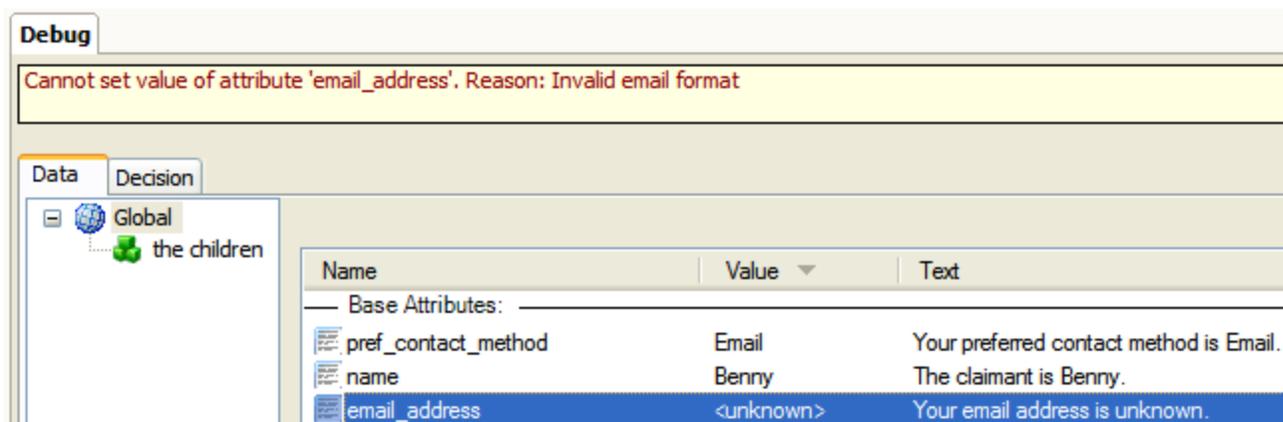
1. Select **Build | Build and Debug**, and then select the **Without screens** option.
2. Right-click the attribute in the **Data** view and select **Set Value**:
3. Enter a value of 20 (ie outside the valid range of 0-18). Note that the configured error message will appear at the top of the Debug view:



The invalid value is not set. A value will not be set until you enter a value within the specified range.

### Test regular expressions

Similarly in the debugger, if you enter invalid data for a variable with a [regular expression](#) attached (ie data which does not comply with the format specified by the regular expression) you will be presented with the configured error message at the top of the Debug view:



As with validation by minimum value and maximum value, the invalid data is not set.

### Test errors and warnings

Unlike validation with minimum value, maximum value and regular expressions (see above), the invalid data which triggers [errors](#) and [warnings](#) will still be set in the debugger. The error message appears in the Output window in Oracle Policy Modeling, not in the **Debug** view.

In the screenshot below, the rules included an error event which triggered if the date of application is in the future.



TIP: If the Output window is not visible (eg because it has been closed or is being hidden by another window), you can view it by selecting **View | Output Window** from the main menu.

## View the attributes inferred in a test case or debug session

Debugging can be undertaken using a 'bottom up' approach. The bottom up approach is where you [set the base level attributes](#) and then view what is inferred from those.

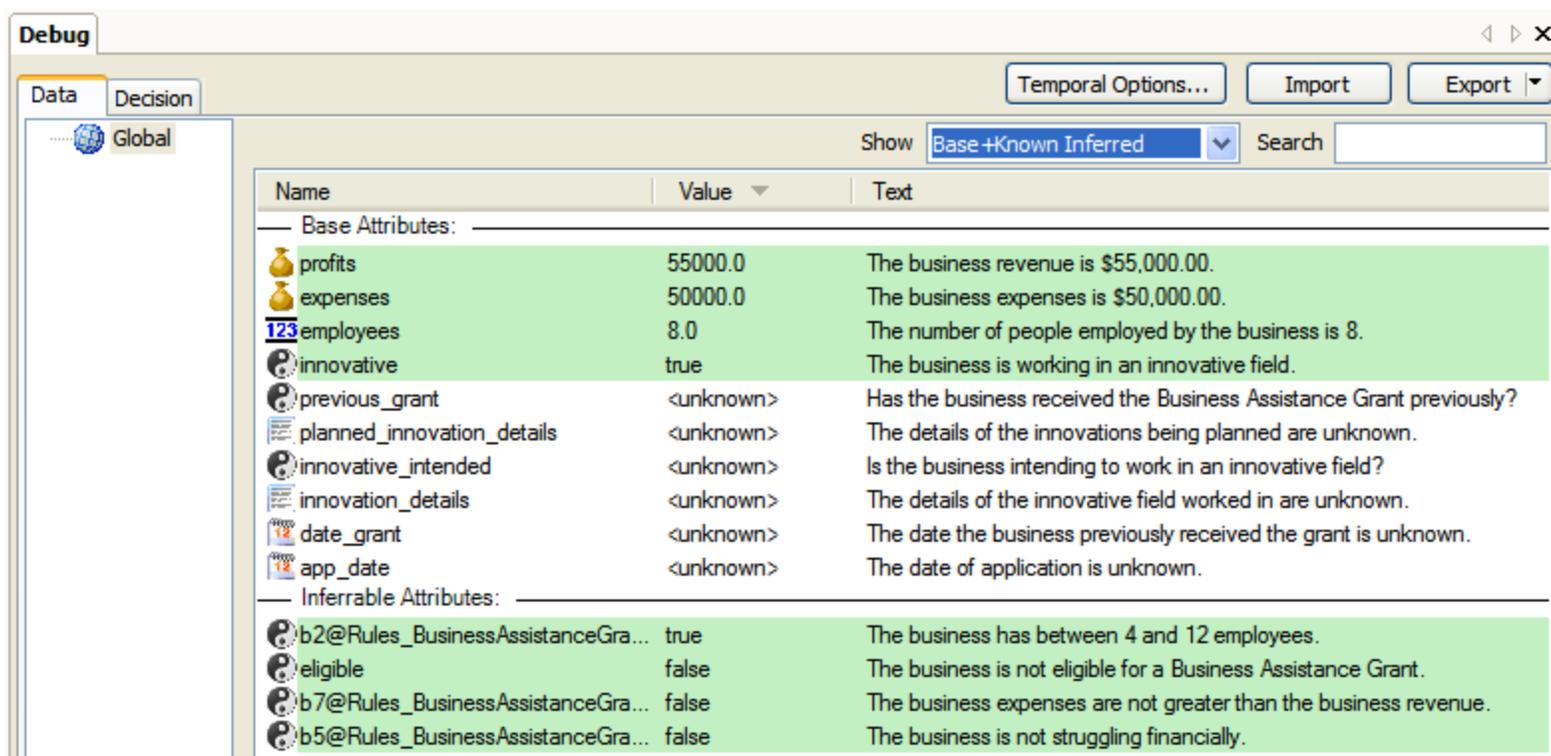
## What do you want to do?

[View the inferred attributes](#)

[View the decision for a known attribute](#)

## View the inferred attributes

After setting the value for a base level attribute, the list of attribute values in the Data view will be updated with the value you specified, as well as the values for any other attributes which have been inferred as a result. Inferred attributes will be highlighted in green in the Data view. The best way to view the attributes you have set and those that have been inferred is to filter the attribute list to show **Base and Known Inferred** attributes:



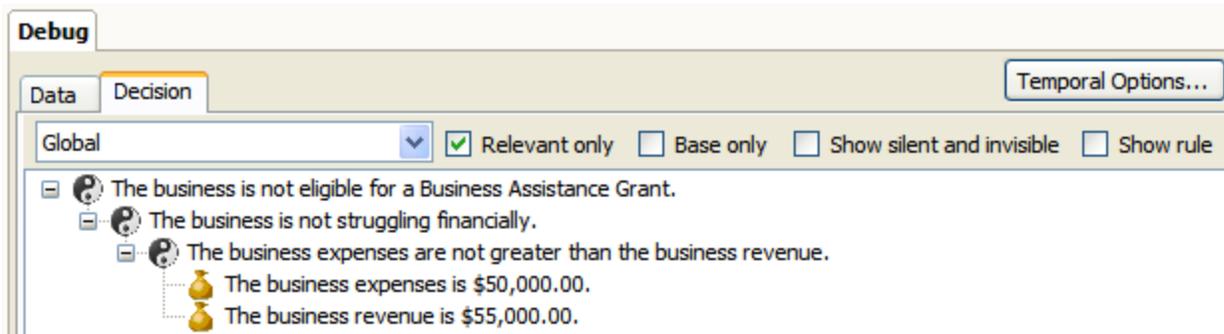
The screenshot shows the Debug window with the Data view selected. The 'Show' dropdown is set to 'Base +Known Inferred'. The table below shows the attributes and their values.

Name	Value	Text
Base Attributes:		
profits	55000.0	The business revenue is \$55,000.00.
expenses	50000.0	The business expenses is \$50,000.00.
employees	8.0	The number of people employed by the business is 8.
innovative	true	The business is working in an innovative field.
previous_grant	<unknown>	Has the business received the Business Assistance Grant previously?
planned_innovation_details	<unknown>	The details of the innovations being planned are unknown.
innovative_intended	<unknown>	Is the business intending to work in an innovative field?
innovation_details	<unknown>	The details of the innovative field worked in are unknown.
date_grant	<unknown>	The date the business previously received the grant is unknown.
app_date	<unknown>	The date of application is unknown.
Inferable Attributes:		
b2@Rules_BusinessAssistanceGra...	true	The business has between 4 and 12 employees.
eligible	false	The business is not eligible for a Business Assistance Grant.
b7@Rules_BusinessAssistanceGra...	false	The business expenses are not greater than the business revenue.
b5@Rules_BusinessAssistanceGra...	false	The business is not struggling financially.

You can sort the attributes in each grouping in the Data view by clicking on any of the column headings (Name, Value or Text). Note that Values are grouped by their type, ie booleans are sorted together, text values are sorted together, etc. Unknowns/uncertains, however, are sorted separately regardless of type.

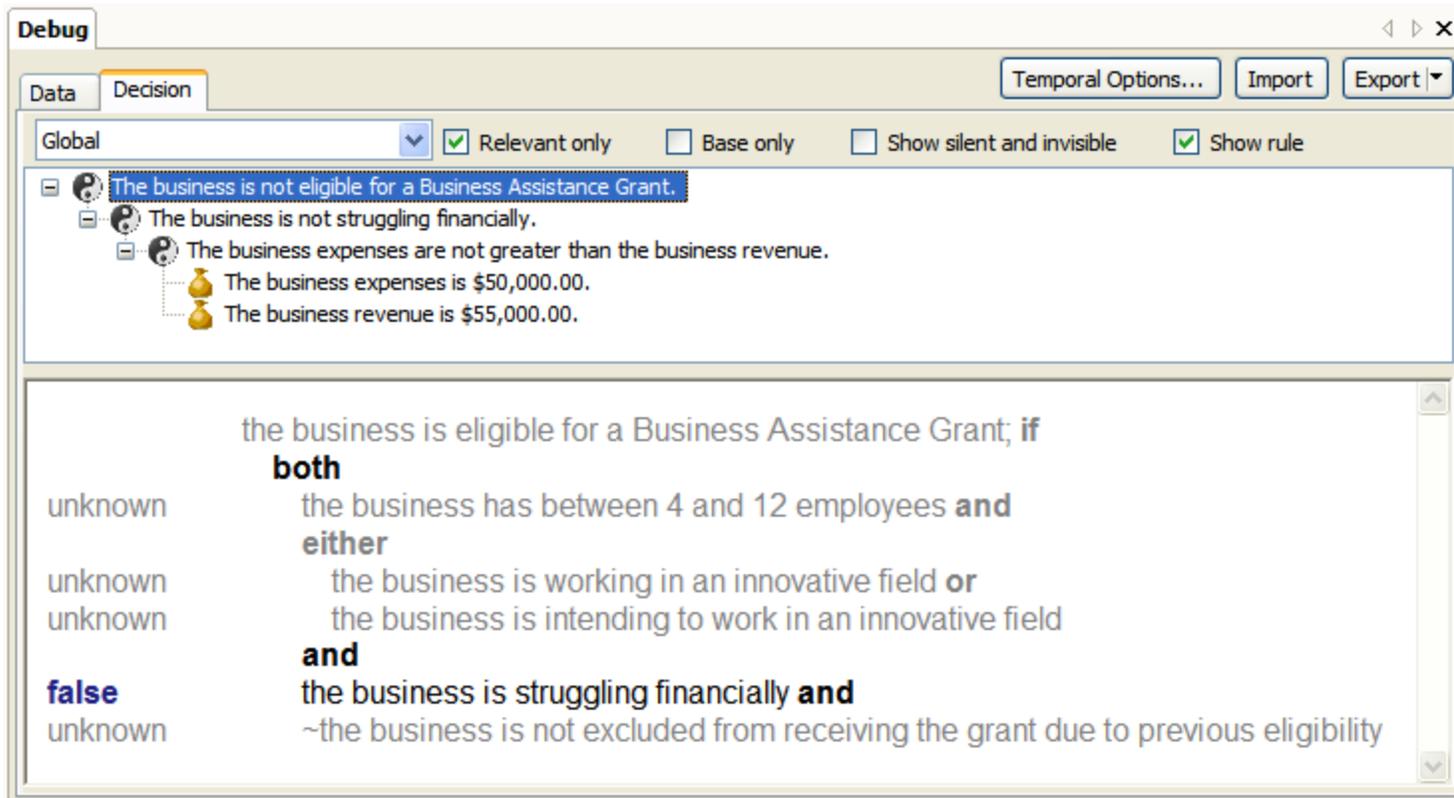
## View the decision for a known attribute

To view the decision for an attribute with a known value, you can right-click the attribute in the Data view and select **Show Decision**. This will open the Decision view with the attribute you have selected in the **Attribute** field. The decision view appears like a decision report showing all the relevant paths that contributed to the goal attribute's value.



The following options are available to alter the behavior of the decision view:

- **Relevant only** - when selected this hides irrelevant paths through the rulebase to the selected goal. This is selected by default. (When this option is not selected, irrelevant attributes are grayed out.)
- **Base only** - when selected this hides intermediate attributes and only shows base level attributes that require an answer in a flat list.
- **Show silent and invisible** - when selected this shows attributes in the decision report that would normally be hidden as a result of silent or invisible operators added to the rules.
- **Show rule** - when selected this displays a pane in the lower part of the Decision tab which shows the rule proving the attribute selected in the decision report. The true/false value of each premise in the rule is shown, with premises irrelevant to the decision grayed out.



If your goal attribute belongs to an entity of which there are multiple instances, you can switch to view the decision tree for different entity instances using the drop-down list of entity instances at the top of the decision view.

## Debug temporal rules and data

You can use the debugger to test temporal rules and data.

### What do you want to do?

[Enter temporal data in the debugger](#)

[Visualize temporal data](#)

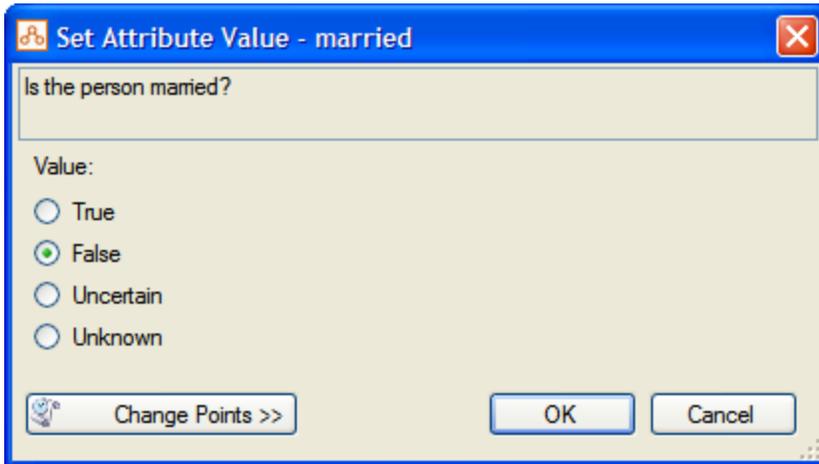
[Understand temporal outcomes](#)

### Enter temporal data in the debugger

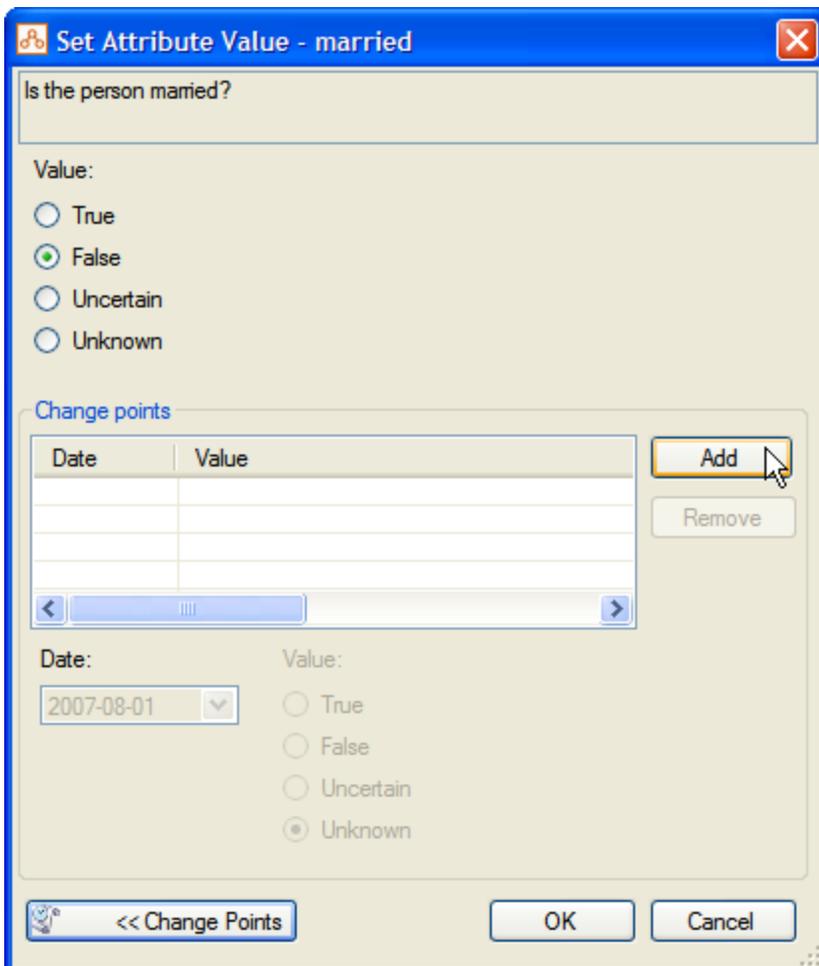
You enter temporal data in the debugger by specifying change points for the base level attributes that you setting values for. A change point represents a value for an attribute applying from a specified date until the next change point (if there is one). To add a change point for an attribute in the debugger:

1. Select the attribute in the Data view or Decision view, right-click and select **Set Value...** (Alternatively, if the attribute is a base level attribute, you can just double-click the attribute.)
2. In the **Set Attribute Value** dialog box, specify the initial value for the attribute. This is the value that the attribute takes up until the first change point. (For the correct format to use when setting variable values, see [Formatting of](#)

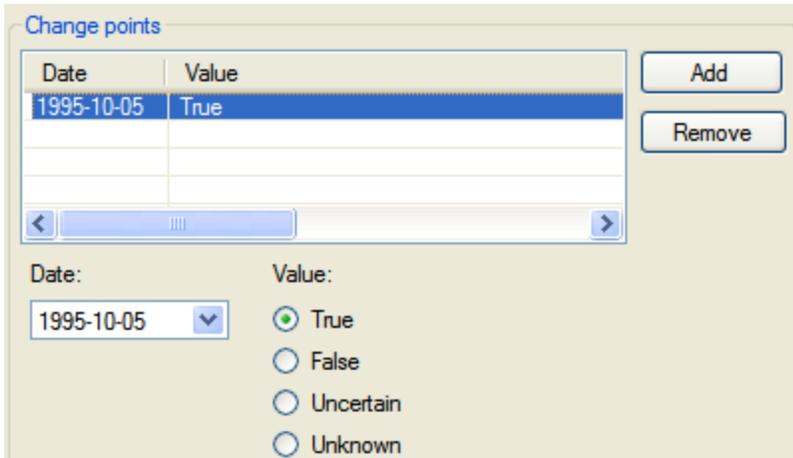
variable values.)



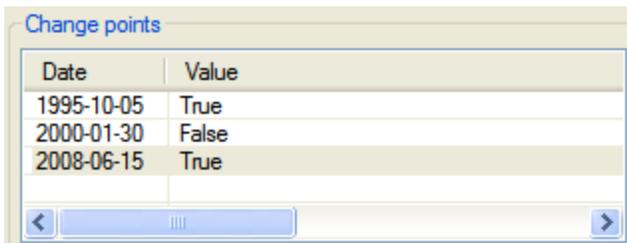
3. Select the **Change Points** button. This expands the dialog box so that you can add Change Points for the attribute.



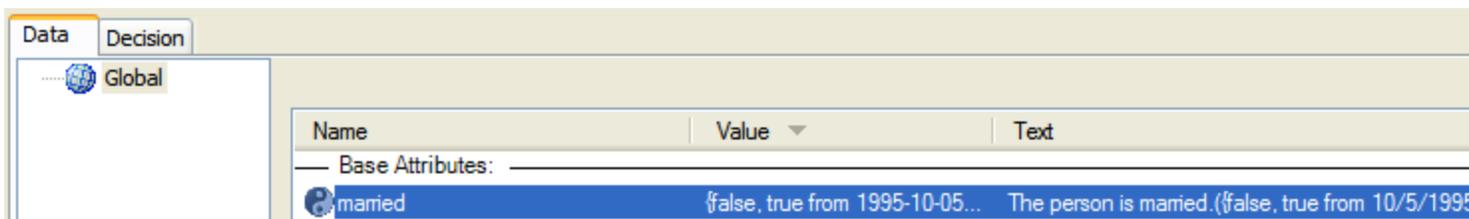
- Click the **Add** button to add a new change point. A change point will be added. (By default this will have today's date and a value of unknown.)
- From the **Date** field, select the desired date (or type a new date). Then select the check box for the **Value** that applies from that date.



- To add additional change points, repeat steps 4 and 5.



- When you have created all the change points, click **OK**. In the Data view you can now see the values you set for the attribute.

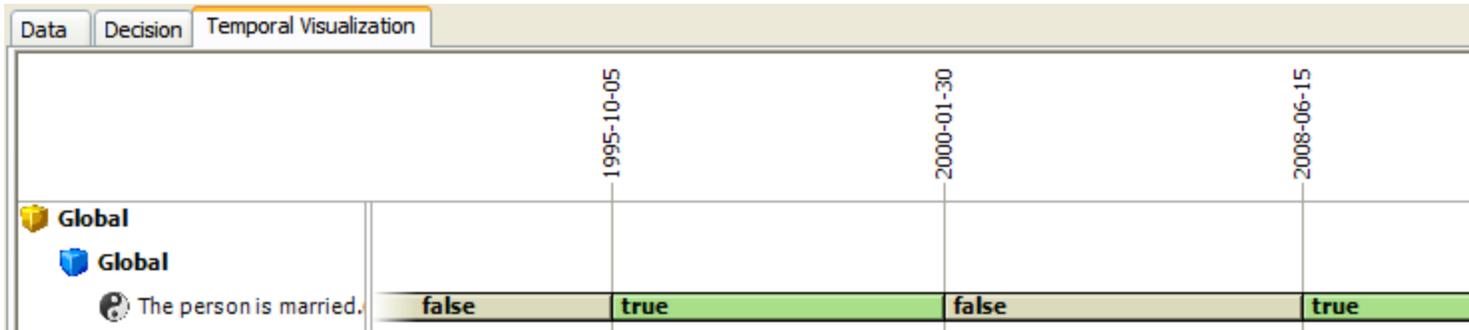


To delete change points, select the desired row or rows in the **Change points** table (from the **Set Attributes Value** dialog) and click the **Remove** button.

### Visualize temporal data

In the debugger, you may want to visualize on a timeline how an attribute's value changes, relative to other attribute's values. To do this:

1. Select the attributes you are interested in the **Data** view or **Decision** view, right-clicking and selecting **Show in Temporal Visualization**.
2. Select the **Temporal Visualization** tab.  
The attributes and their values are displayed in a timeline.



There are three panes to this view:

- The left hand pane shows the attributes, organized by entity. The attributes are labeled either by their name (public name if they have one, otherwise their model id) or by their text, as specified in the **Temporal Options** dialog box.
- The right hand pane shows a timeline for each attribute's values. Non-boolean attributes are indicated with a blue timeline. Boolean attributes have a gray timeline where they have a False value and a green timeline where they have a True value.
- The top pane has a date for every change point represented in the timelines below.

To remove an attribute from the Temporal Visualization view, select the attribute in the Data view, the Decision view or the Temporal Visualization view and deselect **Show in Temporal Visualization**.

The Temporal Visualization tab will remain visible, showing the attributes you have selected, even if you restart the debugger or Oracle Policy Modeling. It will be hidden again once you remove any attributes you have been viewing and restart the debugger.

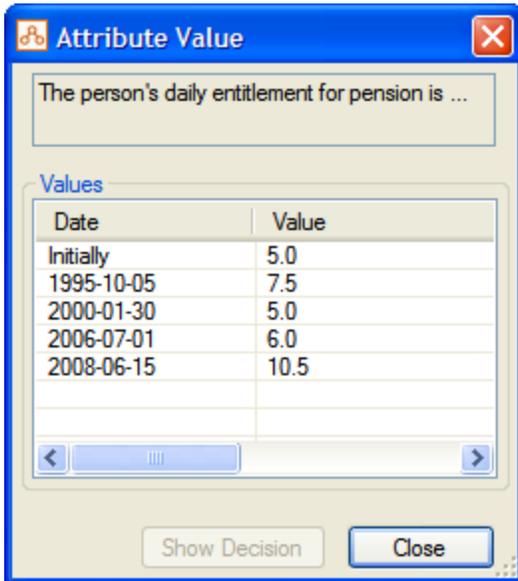
### Understand temporal outcomes

When an attribute takes multiple values over time it can be useful to view a list of the attribute's values and the dates that each of the values apply from.

To view the values for an attribute in the debugger:

1. Select the attribute in the Data view or Decision view, right-click and select **View Value...** (Alternatively, if the attribute is an intermediate level attribute, you can just double-click the attribute.)

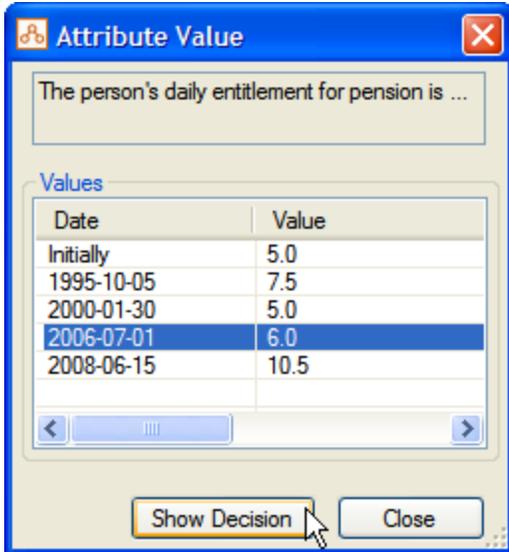
2. In the **Attribute Value** dialog box you can see all the values (change points) for the attribute.



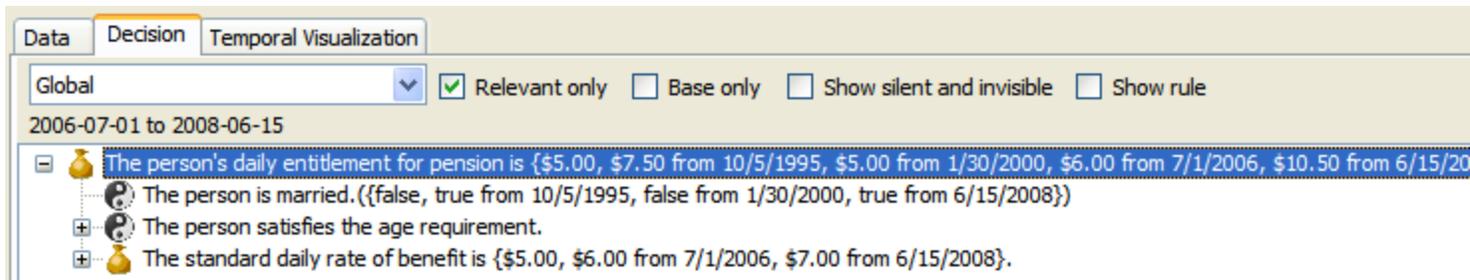
### Understand why an inferred attribute has a particular value on a particular date

You may want to understand why an inferred attribute has a particular value on a particular date. To investigate this:

1. Select the date/value in the **Attribute Value** dialog box and then click the **Show Decision** button.



2. The Decision view opens to show everything relevant to that particular value. You can then review the reason why the attribute has that value from that date.

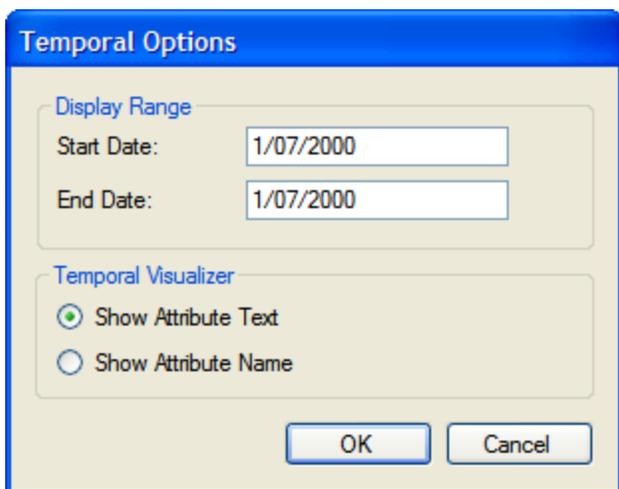


NOTE: The Decision view will limit the relevance period for the decision tree to the period from the current change point to the next change point for the attribute. In other words, only attributes relevant to proving the value of your chosen goal within the relevance period, will be displayed in the Decision Report. Note, however, that any attribute which is displayed in the Decision Report, will display its whole timeline in {curly brackets} after the attribute text, and not just the portion of the timeline which is relevant to proving the value of your chosen goal within the relevance period.

### Limit the display range of attribute value

You may want to limit the display range of attribute values so that you can focus on a particular date range that interests you. To do this:

1. Click on the **Temporal Options** button in the debugger.
2. In the **Temporal Options** dialog box, specify a **Start Date** and an **End Date**. Note that the Start Date is inclusive and the End Date is exclusive.



3. Click **OK**. In the Data view, the values in the Value column will be limited to the dates specified in the Display Range.

### Find the cause of a logic error

Sometimes when debugging, an inferred attribute may have a different value from what you expected based on the input data. In this case you will want to see the decision that led to the value. To understand a decision:

1. Select the attribute in the **Data** view, right-click and select **Show Decision**. (If there are multiple values for the attribute, you also have the option to show the decision for a particular value from that change point.)

2. The decision report for this attribute is shown in the **Decision** view, in which you can also [show the relevant rule](#). (If you have opted to show the decision for a particular value, the Decision view will limit the relevance period for the decision tree to the period from the current change point to the next change point for the attribute. In other words, only attributes relevant to proving the value of your chosen goal within the relevance period, will be displayed in the Decision Report. Note, however, that any attribute which is displayed in the Decision Report, will display its whole timeline in {curly brackets} after the attribute text, and not just the portion of the timeline which is relevant to proving the value of your chosen goal within the relevance period.)

When you review the reasons for a decision, you might uncover a logic error in your rules. To update your rule:

1. Firstly, save your debugger session so that you will be able to retest any changes that you make to the rulebase. (For more information, see [Change a rule while debugging](#).)
2. Find the rule in your rules document. (For more information, see [Find rules that use an attribute or relationship](#).)
3. Make the necessary changes to the rule.
4. Compile your rules.
5. Start the debugger and import your saved session data.
6. Check that the rules now operate as expected and produce the right decisions based on the test data.

You can also use the Rule Editor in Oracle Policy Modeling to check the underlying logic of a rule. To view a rule in the Rule Editor:

1. In the Project Explorer, right-click the rules document and select **Open Generated Rules**.
2. On the **Rules** tab, locate the rule that you want to investigate.
3. Double-click it to open the **Rule Editor** for that rule.
4. If you want to make a change to the rule, click the **View in Word** or **View in Excel** button.

## Change a rule while debugging

While the debugger session is active, you can navigate through Oracle Policy Modeling and edit rules without stopping the debugger session.

If you want to test your new rulebase against existing data you have entered in the debugger, then you can use the "Retain existing session data" option to do this:

1. With the debugger still running, make the necessary changes to the rules in Word or Excel documents. (See also [Find the cause of a logic error](#).)
2. Select **Build | Build and Restart Debugger**.
3. In the **Debug Options** dialog box, select the **Retain existing session data** checkbox.
4. Click **OK**.
5. Check that the rules now operate as expected and produce the right decisions based on the test data.

NOTE: If you have deleted or renamed any attributes, entities or relationships, the data associated with those items will be discarded when the debugger is restarted.

## Save or reload a debugger session

When testing rules with data in common it can be useful to save a session containing the base data so that it can be reused in future testing.

## Save a debugger session

To save a debugger session:

1. After you have [set up your baseline data in the debugger](#), click on the arrow on the right side of the Export button (located in the top right of the Debug view), and select the **Export as XML** option.
2. In the **Save As** dialog box, enter a file name and select the destination folder. Click **Save**.

The entity instances and user-set values in the test will be saved in the Data Set (XDS) XML file.

## Reload a debugger session

To reload a debugger session which you have previously saved:

1. Open the debugger and click the **Import** button.
2. In the **Open** dialog box, browse to select the XDS file which contains your saved data. Click **OK**.

The data and settings from your saved session will be added to the existing session. If you already have entity instances in your session, it will try to match up any entity instances in the data set with those that already exist.

**NOTE:** Only entity instances and user-set values are saved in a Data Set (XDS) file. Inferred attributes are not saved.

# Deployment

Topics in "Deployment"

- [Deploy an interview to Web Determinations](#)
- [Deploy a rulebase or interview to Determinations Server](#)
- [Deploy a rulebase to a custom application or mobile device](#)
- [Polish a rulebase for deployment](#)

See also:

- [Create and deploy a rulebase](#)
- [Validate user input using errors and warnings](#)
- [Use rules to trigger external software applications](#)

## Deploy an interview to Web Determinations

Oracle Web Determinations is one way in which a rulebase generated using Oracle Policy Modeling can be used. Web Determinations has the following features:

- [Intelligent navigation of questions](#) – Web Determinations makes use of the Oracle Determinations Engine's [inferencing capability](#), which provides intelligent rule interviews through its cyclical technique of querying rules and drawing inferences. By ignoring irrelevant rules the application asks only necessary questions in order to reach a conclusion.
- [Decision reports](#) – Web Determinations produces automatic visual representations of decision trees generated during interviews. These decision trees demonstrate how and why decisions have been reached by reference to rules and their underlying propositions.
- [Document generation](#) – Web Determinations can generate documents based on interview data and reasons for decisions.
- [Data review screen](#) – Web Determinations maintains lists of screens visited during interview sessions. This provides rapid access back into interviews to allow users to quickly access and change data.
- [Built-in help/commentary](#) – Web Determinations can include HTML pages which act as help or commentary for the application.
- [Interview saving](#) – Web Determinations has built-in support for saving and reloading interview sessions. Users are warned if they try to navigate away from a screen without submitting data, to prevent the lose of data.
- [Customizable user interface](#) – The web-based user interface in Web Determinations is customizable by rule developers.

To deploy an interview to Oracle Web Determinations:

1. In Oracle Policy Modeling, select **Build | Build and Run**.
2. In the **Build and Run** dialog box, select the option **Run with Oracle Web Determinations**.
3. If you want to completely replace the previously deployed version of the project (located in the Release folder), click the checkbox to **Replace deployed version for project**. TIP: This is useful in situations such as either using a new version of Oracle Policy Modeling or when using an updated customized version of Oracle Web Determinations (but note that this will discard any customizations made to the previously deployed version).
4. Click **Run**.

See also:

- [Test an interview or screen flow](#)

## Deploy a rulebase or interview to Determinations Server

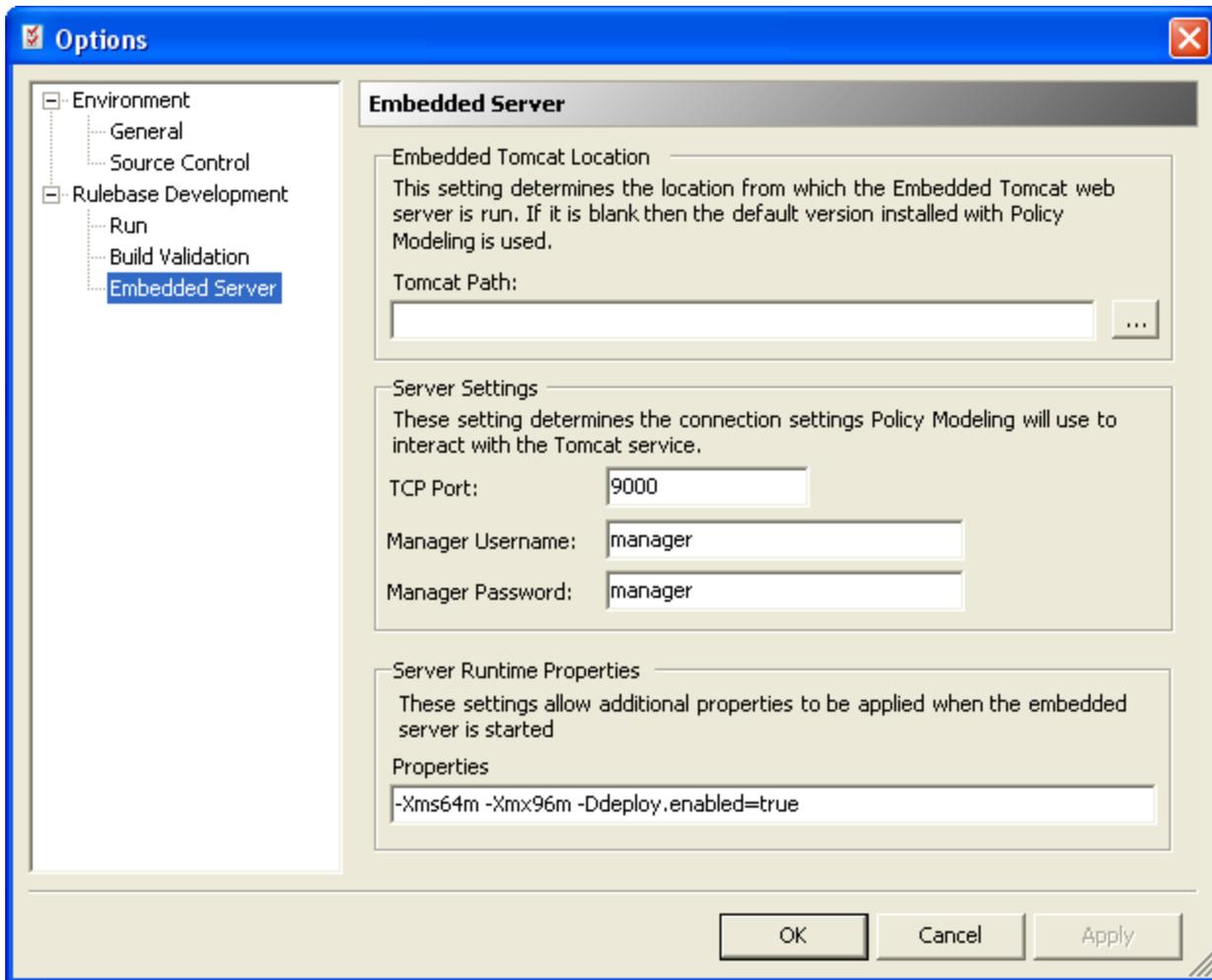
You can deploy a rulebase or interview to Oracle Determinations Server. This is handy for demonstration purposes, as well as a configuration-less testing platform for Determinations Server.

To deploy a rulebase or interview to Oracle Determinations Server:

1. In Oracle Policy Modeling, select **Build | Build and Run**.
2. In the **Build and Run** dialog box, select the option **Run with Oracle Determinations Server**.
3. If you want to completely replace the previously deployed version of the project (located in the Release folder), click the checkbox to **Replace deployed version for project**. TIP: This is useful in situations such as either using a new version of Oracle Policy Modeling or when using an updated customized version of Oracle Determinations Server (but note that this will discard any customizations made to the previously deployed version).
4. Click **Run**. (If the Check Determinations Server Compatibility option has not been selected in **Tools/Options/Rulebase Development/Build Validation**, then the **Enable Determinations Server Compatibility** dialog will be shown. Click **OK**.)

To modify the settings that are used to run the embedded server:

1. In Oracle Policy Modeling, go to **Tools/Options/Rulebase Development/Embedded Server**.



NOTE: An example of a *Tomcat Path* might be: `C:\Program Files\apache-tomcat-5.5.28-embed`

2. Change the settings as required, then click **OK**.

## Deploy a rulebase to a custom application or mobile device

For details on how to deploy a rulebase to a custom application or mobile device, please refer to the Oracle Policy Automation Help.

## Polish a rulebase for deployment

There are several ways in which you can polish your rulebase for deployment. Here are some ideas and suggestions with links to the relevant topics.

### What do you want to do?

[Personalize an interview](#)

[Configure the screens](#)

[Add default values and validate user input](#)

[Improve decision reports](#)

[Customize Oracle Web Determinations](#)

[Personalize an interview](#)

### **Using name substitution**

Name substitution personalizes the interview for a more user-friendly experience. You can collect a person's name at the start of an interview, and then the name will be automatically substituted in later questions, in decision reports and on the summary screen.

For more information see:

- [Substitute the actual value of a variable for its text](#)
- [Set up substitution](#)
- [Text substitution principles](#)

### **Using gender pronoun substitution**

Gender pronoun substitution can be used in combination with name substitution (or in isolation) to provide more natural language text.

For example, "the student avoided handing in the student's assignment" becomes (in combination with name substitution) "Matthew avoided handing in his assignment".

For more information see:

- [Substitute a gender pronoun for a text variable](#)
- [Collect the gender of a person](#)

### **Substituting names in headings and labels on screens**

Variable values such as the person's name and/or age can be substituted into screen names and labels. For example, you could have a screen name appear as "School Details – Bart, aged 10 years". For more information see:

- [Substitute an attribute value into the text on screens](#)

### **Using second person sentence generation**

Sentences and questions can be generated in second person rather than in third person in order to personalize the interview. For example, instead of "Does the applicant have health insurance?" the question is asked as "Do you have health insurance?". For more information see:

- [Display interview questions in second person form](#)

[Configure the screens](#)

### **Using screen labels**

Labels can be added to question screens and the summary screen to provide context. They can also be used as additional headings. Labels can include static text, as well as HTML. For more information see:

- [Add labels to question screens](#)
- [Add a label to the summary screen](#)

## Hiding and displaying summary screen elements

Using visibility attributes, you can control whether screen elements are displayed or hidden at various stages of the interview based on logic. For example, you might want to display a goal to investigate at the start of the interview, but then hide it at the end. For more information see:

- [Control the visibility of summary screen elements](#)
- [Tutorial: Hiding and displaying summary screen elements](#)

Add default values and validate user input

## Defaulting values on Oracle Web Determinations screens

You can set default values for any attribute on a question screen. Defaults can be a specific value, or can be dynamically determined based on data collected on previous screens. Providing defaults reduces the amount of typing/clicking required to complete an interview. For more information see:

- [Specify a dynamic default for an input](#)
- [Specify a default value for an input](#)

## Validating user input on Oracle Web Determinations screens

You can validate the input that a user enters to warn or prevent them from entering values which do not meet certain criteria when running the rulebase. Specific errors and warnings can be triggered by conditional logic in rules. For example, you could display the message "Please check the dates of birth as you have indicated that your date of birth is after your child's date of birth" if the applicant's date of birth > the child's date of birth. Defining maximum and minimum values, or using regular expressions, for an attribute are other ways to fire generic error messages when the input value falls outside the specified range or does not meet a specified format (eg an email address). For more information see:

- [Write an error event rule](#)
- [Write a warning event rule](#)
- [Specify minimum and maximum values](#)
- [Use regular expressions](#)

Improve decision reports

## Automatically generating structural elements

Decision reports can be improved by ensuring that structural elements in legislation (eg section, paragraph, article, etc.) and policy (eg chapter, guideline, etc.) are included. Oracle Policy Modeling can automatically generate these structural attributes. The default form is "section x is satisfied", but this can be configured. For more information see:

- [Use structural elements to model legislative structure](#)
- [Use keywords to customize automatic structural attributes](#)

## Using grouping connectors and intermediate attributes

Adding intermediate attributes to your rules can make decision reports more meaningful. For example, adding "the person satisfies the income test" as an intermediate attribute in between "the person is eligible for the benefit" and "the person's income < 3000" in the following rule results in a more useful decision report.

**the person is eligible for the benefit if**

the person satisfies the income test  
the person's income < 3000

For more information see:

- [Improve the wording of a rule](#)

### **Trimming the decision report**

Decision reports automatically generated based on logic can be extremely verbose in terms of language and the structure of the rules. You can 'trim' decision reports using the silent and invisible parameters making them much easier to follow ("silent" hides all the logic nested below the attribute, "invisible" hides the attribute only and can be applied locally or globally). For more information see:

- [Hide information in a decision report](#)

Customize Oracle Web Determinations

### **Defining a data review screen**

The data review screen in Oracle Web Determinations displays the questions asked during the interview and the answers provided. Question screens on the data review screen are listed in the order defined in the screen order in the screens file. If no screen order is defined in the screens file, the screens will appear in a random order in the default data review screen in Web Determinations, which is not very user-friendly. It is therefore recommended that you define a screen order. For more information see:

- [Define interview screen order](#)

### **Showing the progress stages**

Stages can be displayed at the top of interview screens (with the current stage/screen shown in bold) to show the user their progress through the interview. Progress stages are turned on by default if a Screen Order has been defined. For more information see:

- [Progress stages](#)

### **Configuring the Oracle Web Determinations labels**

Standard out-of-the-box Web Determinations label text, such as Yes, No, Submit, Load etc, can be modified in the messages.<locale>.properties file for the project. For more information see:

- [Configure the Oracle Web Determinations labels](#)

### **Changing the Oracle Web Determinations banner**

The Oracle Web Determinations banner, which by default is the Oracle logo and the text "Web Determinations", can be replaced with any other image and/or text (eg your application's logo and name). For more information see:

- [Change the Oracle Web Determinations banner](#)

### **Providing commentary/help text**

Commentary (context-sensitive help text) can be provided to help users understand the questions that they are being asked and the screens they are being presented. For more information see:

- [Create, update or delete interview help](#)

# Custom functions and programming

Topics in "Custom functions and programming"

- [Install a custom function](#)
- [Debug with a custom function](#)
- [Write a rule that uses a custom function](#)

See also:

- [Oracle Policy Automation Developer's Guide](#)

## Install a custom function

A custom function is a function written by a programmer in Java or .NET, conforming to a particular interface that makes them callable from a rulebase. Custom functions are a method of performing custom processing to return a calculated value, and are used when the required calculations cannot readily be performed with existing rulebase functions.

For a custom function to be available while writing and running your rulebase, it must be installed in a project folder at Development\Extensions\

For full details on the structure of the custom function files, see the [Oracle Policy Automation Developer's Guide](#).

## Debug with a custom function

In order to debug rules using a custom function, the [extension implementing the custom function must be installed](#) in the \Extensions\

- If you are [debugging with screens](#) then you must either have a Java implementation of the custom function available, or a custom version of Web Determinations which includes the custom function implementation.
- If you are [debugging without screens](#) then you must have a .NET implementation of the custom function available.

Further information on how to install a custom function is available in the [Oracle Policy Automation Developer's Guide](#).

## Write a rule that uses a custom function

### What is a Custom Function?

A function written in Java or .NET, conforming to a particular interface that makes it accessible from within a rulebase. Inputs are defined by the custom function, and are most often attributes whose values are used by the custom function.

### What can a Custom Function do?

Possible uses of custom functions are:

- To create a function that will be used repeatedly in rules (eg a function that counts days since the last reporting period).
- To perform a specific, non-standard calculation (eg where two accidents causing damage to 5% and 10% of the vehicle are considered to have damaged 12% of the vehicle in total).

### Call a Custom Function

To call a custom function, use the following syntax:

- `<conclusion attribute> = <custom function name>(<input 1>, ..., <input n>)`

The custom function inputs in brackets are defined by the custom function.

For example, a custom function "AccidentDamage" to calculate the total accident damage, where "the total accident damage", "the damage to car 1", and "the damage to car 2" are all number attributes (percentages), would be called in a rule as follows:

**the total accident damage = AccidentDamage(the damage to car 1, the damage to car 2)**

Note that a custom function is called in the same way as any of the other built-in functions in Oracle Policy Modeling, and similarly the custom function rule will take part in the question search and be included in decision reports in the same way as any other built-in function. Any of the input attribute values can be temporal, as can the custom function return value.

The custom function must be defined in the `\Extensions\<extension name>` project folder before any rules using it can be compiled. See [Install a custom function](#) for details. TIP: Create and debug your custom function in a small test project, and then copy it into your actual project.

For more information, refer to the *Write a Custom Function Extension* topic in the [Oracle Policy Automation Developer's Guide](#).

# Collaborating

## Topics in "Collaborating"

- [Work collaboratively on rule projects](#)
- [Include rules defined in a separate project](#)
- [Share rule documents across projects](#)
- [Use multiple properties files on a multi-developer project](#)
- [Import data model from another rule project](#)
- [Track rulebase changes on multi-developer projects](#)
- [Open a rulebase project from source control](#)
- [Connect or disconnect a project with source control](#)
- [Track versions of rule documents](#)
- [Get updates to rule documents from source control](#)
- [Retrieve a specific document version](#)
- [Create multiple rulebase versions](#)

## See also:

- [Manage legislation and other source material](#)
- [Export or import a data model](#)
- [See the results of a recent build or deploy operation](#)
- [Import test cases from another project](#)

## Work collaboratively on rule projects

When different individuals or teams need to work collaboratively on rule projects there are a couple of ways to approach this:

1. Separate Oracle Policy Modeling projects, each with the same data model but different rules documents, are developed for each part of the overall project and these modules are later combined. This is useful where you are able to sub-divide the work and work independently with no need to edit the same content. For more information on this method, see [Include rules defined in a separate project](#).
2. The rule and/or source documents themselves are shared between projects. This is suitable when the individuals or teams need to make changes to the same files. For more information on how to do this, see [Share rule documents across projects](#).

## Include rules defined in a separate project

You can share rules between rulebases with a common data model by building one rulebase as a module and then linking to this module from another rulebase.

This may be useful where there are:

- Multiple teams working on one main rulebase read more<sup>1</sup>
- Common rules that are used by several rulebases read more<sup>2</sup>
- Core rules that are deployed in different ways for distinct audiences read more<sup>3</sup>
- Different combinations of the same rules used by multiple rulebases read more<sup>4</sup>

TIP: To see an example of a complete rulebase with a module, open and run the [Income Support Benefit example rulebase project](#) provided in the Examples folder in the Oracle Policy Modeling installation folder.

## What do you want to do?

Add a link to a module

Use entities, attributes and relationships imported from a module

Include the rules from an imported module

Use the translations provided in an imported module

Build and load a rulebase containing modules

Debug a rulebase containing modules

Add a link to a module

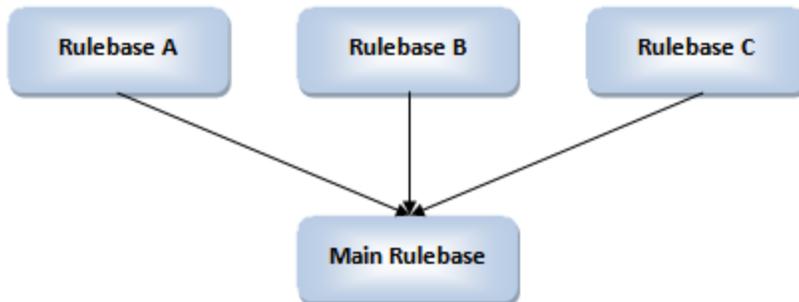
To create a link to a module from your rulebase:

1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the module link to be placed in (eg the Modules folder), right-click and select **Add Module Link...**

---

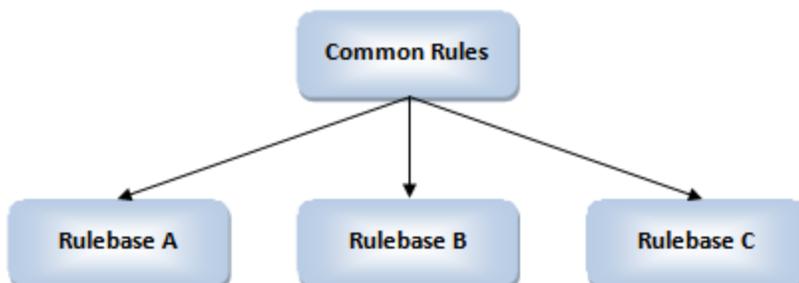
1

Modules allow teams to work independently on sub-components of a ruleset and combine their work into a single rulebase.



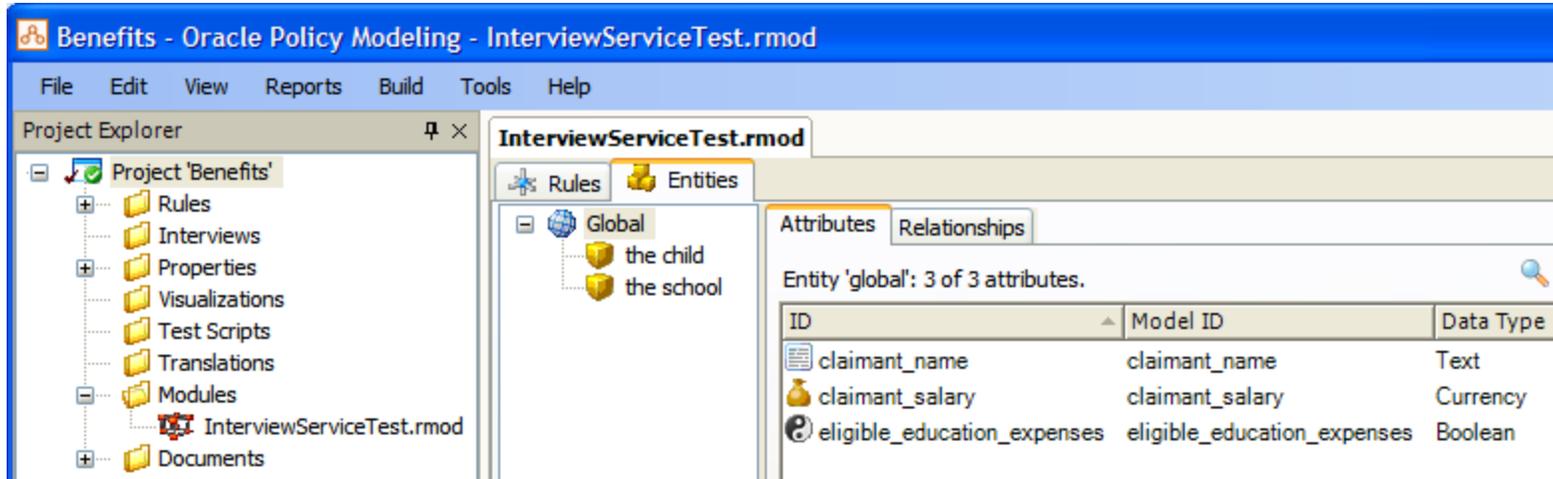
2

Modules allow rule developers to import a common set of rules and screens into other rulebases, avoiding the need to create and maintain common questions in multiple places.



- In the **Open** dialog box, browse to the module file (.rmod) that you want to link to. (The .rmod file is created when you [build a module](#) and is located in the output folder for the module's project.) Then click **Open**. The module's compatibility with the project will be checked (see [Validation of a module upon linking](#) below) and then, assuming there are no errors that need to be addressed, the module link will be added.

Once a module is imported, it will display in Oracle Policy Modeling as a read-only single source file, which will show the data model that was imported, including the complete definition of the entity, attribute or relationship as defined in the module. It will also show a set of rules that indicate the relationship between the base level and inferred entities, attributes and relationships and the data model items that prove it, but not any details of exactly how they are proved.



A key difference between adding modules, as opposed to other documents, is that adding a module doesn't copy it into that folder, but rather creates a link to the location of the module. The link location will be relative to the project's xprj file where possible, or absolute if necessary. This allows a module to be linked to without every person who works on that project having to have exactly the same file structure.

TIP: If you want to remove a link to a module from your project, you should right-click the rmod file in the Project Explorer and select **Remove from Project**. If you select **Delete** instead, the rmod file itself in the module's project will be deleted.

### Validation of a module upon linking

In the process of adding a link to a module, Oracle Policy Modeling validates that:

- The definition of any attributes/entities/relationships that already exist in the project match exactly, so that the corresponding attributes, entities and relationships can link together. If the data models match, it merges the rulebase and module. Any conflicts will result in build model errors which will need to be rectified before the project can be built.
- The rule language of the module is the same as the rule language of the parent project. If they are different the module will not be imported. (It is OK for the project region to be different in the module and the parent project.)
- A module with the same name does not already exist in the project.

### Use entities, attributes and relationships imported from a module

When a module is imported into a project all the entities, attributes and relationships that are defined in the module's public interface appear as if they were defined in a properties file in that project itself and can be used like any other item defined in a properties file. There is, however, one important restriction in that items imported from a module cannot be edited from within the parent project.

To make changes to the module you need to edit the module's project and rebuild it. Similarly, you cannot make changes in the parent project that override the module's attribute, entity and relationship definitions. Consequently, if you additionally define an imported entity, attribute or relationship in a parent project you cannot define anything other than the text of that item.

### **Text substitution**

Since you cannot alter the definition of an attribute imported from a module, substitution attributes will only apply up the chain and not down it. This means that any substitution attributes defined and exported from a module will apply to any other project that imports it. However, the reverse is not true, ie defining a substitution attribute in the parent project will not apply to any attributes defined in a module.

In order for second person substitution to work it has to be turned on in the main rulebase project and every module it includes. For example, say your rulebase consists of the following:

Module 1 -- (imported by) --> Module 2 -- (imported by) --> MyRulebase

Now if the attribute 'the person' is defined and exported from Module 1, then it will also need to be defined as a second person attribute in both Module 2 and MyRulebase.

NOTE: Defining attributes or relationships in multiple modules and then importing them into a project can lead to inconsistent text substitution (or sentence forms) being displayed.

### **Define attributes, entities and relationships in multiple modules**

If an attribute, entity or relationship is exported from multiple modules (or the module and its parent rulebase) then their definitions, including any metadata must match exactly. This means that you would need to change and redeploy multiple modules if you ever wanted to update the definition of the attribute/entity/relationship. (Having an attribute or relationship defined and exported from multiple modules in a project can also result in unexpected behavior in terms of text substitution and sentence forms.)

The recommended approach is therefore to define each entity/attribute/relationship only once and then import them into every other project that needs it.

For example, say a rulebase imports modules A and B, both of which need to use an attribute called 'the person'. Instead of defining that attribute in both modules A and B, it should be defined in another module which is then imported into both A and B.

### **Include the rules from an imported module**

Including a rule from an imported module in a project is simply a matter of using the publicly-named attribute (from the module) in a rule (in the parent project). Note that you don't have access to anything inside a module that has not been exported (eg intermediate attributes), either to prove or to use in rule premises. You can have attributes with the same text in the parent rulebase and the imported module but unless the one in the module has been exported, these will not be auto-aliased together and you will end up with two attributes with the same text at runtime.

### **Use the translations provided in an imported module**

Any translations provided in an imported module are exported. This means that a project using the module does not have to recreate existing translations for the contents of the module. These translations cannot be edited, however, from within the parent rulebase - any changes must be made in the module itself.

All modules imported into a project must also support all the languages the project does. (A module may support more languages than the project does but it cannot support less.)

TIP: The languages supported in a module can be viewed in the parent project by right-clicking the module link in the Project Explorer and selecting **Properties**.

## Build and load a rulebase containing modules

You build a rulebase that contains modules in the same way you build a normal rulebase (ie **Build | Build**). This process creates a build-time copy of the module's interface file in the rulebase, which is later validated against the runtime version of the module's interface file. (At runtime you can update an individual module without having to update all the rulebases (or modules) that rely on it provided that the changes do not alter the interface definition of the module. See [Deploy changes to a single module](#) for more information.)

When you load the rulebase (ie in the debugger, regression tester, Web Determinations or Determinations Server), all the rules from all the modules are combined to form the final rulebase that is then checked for consistency, logical loops, multiple proven attributes etc. This means that:

- you cannot have rule fragments that cross over module boundaries, and
- your data models must align, and
- logical loops and multiply proven attributes must be considered for the unified rulebase, not just for the individual modules.

## Validation at runtime of a rulebase containing modules

In addition to the build time checks, a rulebase that contains modules is also validated by the engine when it is loaded. The validation checks that are carried out mirror those done by Oracle Policy Modeling at build time, with the following notes:

- i. Maximum/minimum/regular expression validations get compiled out as custom properties so they will be reported as mismatches in custom properties if they are out of sync.
- ii. None of the properties relating to text substitution or text overrides get validated at runtime. Consequently, it is inadvisable to define attributes in more than one module or rulebase (see above). The engine does additional validation to ensure that the module's interface has not been modified since all the rulebases and modules that rely on it were compiled.

Since the engine, like all runtime validation, only reports the first error it encounters it is advisable to re-compile your head rulebase project after updating a module to detect any errors.

## Data matching at runtime

Entities, attributes and relationships are defined by their ID and are compared according to the following criteria:

- For an attribute to be considered the same it must have the same ID, base text (including parse) and data type.
- For an entity to be considered the same it must have the same ID and text.
- For a relationship to be considered the same it must have the same ID, reverse ID, text, reverse text, type, source entity and target entity. Additionally, the concept of primary direction is important such that a relationship defined as:

ID	Source	Text	Type	Reverse text	Target	Reverse ID
children	the child	the applicant's children	One-to-Many	the applicant of the child	the applicant	childsapplicant

will not be considered the same as a relationship defined as:

ID	Source	Text	Type	Reverse text	Target	Reverse ID
childsapplicant	the applicant	the applicant of the child	Many-to-One	the child	the applicant's children	children

- Additionally, attributes, entities and relationships cannot change its inferencing type from base to inferred or vice versa.

## Redeployment

It is possible to make changes to a module and deploy it without having to redeploy everything that uses it, if and only if the changes do not alter the external data model. In other words, the changes must be limited to changing the internal logic of the module. If the change affects the external data model, then you will need to rebuild and redeploy every rulebase/module that depends on it. (For more information see [Deploy changes to a single module.](#))

### Debug a rulebase containing modules

A rulebase containing modules is considered a single rulebase and you therefore debug it as you would any other rulebase (see [Debug a rulebase](#) for more information).

When you run the rulebase in the debugger, any attributes, entities and relationships that were not exported from your modules are shown with an ID of the form <module name>.<module document id>. For example, SimpleBenefits.b7@Rules\_BenefitRules\_doc is the attribute 'b7@Rules\_BenefitRules\_doc' that was not exported from the Simple Benefits module.

Name	Value	Text
<b>Base Attributes:</b>		
123 claimant_age	<Unknown>	The claimant's age is unknown.
123 claimant_employment_length	<Unknown>	The number of years that the claimant has worked for the company is unknown.
claimant_gender	<Unknown>	The claimant's gender is unknown.
claimant_income	<Unknown>	The claimant's annual income is unknown.
claimant_name	<Unknown>	The claimant is unknown.
claimant_pregnant	<Unknown>	Is the claimant pregnant?
claimant_public_housing_client	<Unknown>	Is the claimant a public housing client?
eligible_education_expenses	<Unknown>	Is the claimant eligible for education expenses?
leave_type_application	<Unknown>	The type of leave being applied for is unknown.
<b>Inferable Attributes:</b>		
eligible_low_income_allowance	<Unknown>	Is the claimant eligible for low income allowance?
eligible_maternity_allowance	<Unknown>	Is the claimant eligible for maternity leave?
eligible_teenage_allowance	<Unknown>	Is the claimant eligible for the teenage child allowance?
investigation_complete	<Unknown>	Is the investigation complete?
low_income_allowance_payment	<Unknown>	The claimant's low income allowance payment is unknown.
SimpleBenefits.b10@Rules_BenefitRules_doc	<Unknown>	Is the interview complete?
SimpleBenefits.b11@Rules_BenefitRules_doc	<Unknown>	Should the pregnancy question be mandatory?
SimpleBenefits.b12@Rules_BenefitRules_doc	<Unknown>	Should the pregnancy question be displayed?
SimpleBenefits.b13@Rules_BenefitRules_doc	<Unknown>	Is the claimant aged between 10 to 60 years?
SimpleBenefits.b14@Rules_BenefitRules_doc	<Unknown>	Is the claimant male?
SimpleBenefits.b15@Rules_BenefitRules_doc	<Unknown>	Should the long service leave option be displayed?
SimpleBenefits.b16@Rules_BenefitRules_doc	<Unknown>	Should the maternity leave option be displayed?
SimpleBenefits.b7@Rules_BenefitRules_doc	<Unknown>	Has the claimant worked for the company for more than 10 years?
SimpleBenefits.b8@Rules_BenefitRules_doc	<Unknown>	Is the claimant eligible for long service leave?
SimpleBenefits.b9@Rules_BenefitRules_doc	<Unknown>	Has the claimant worked for the company for more than 10 years?

These attributes are shown in the debugger for completeness (eg so that you can see a proper decision report).

## Share rule documents across projects

There are three ways to share Oracle Policy Modeling documents across projects:

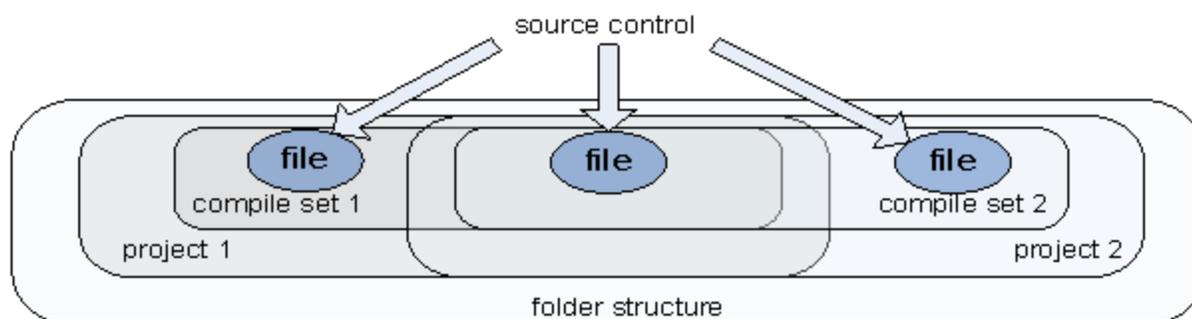
- have multiple Oracle Policy Modeling projects in one folder structure;
- share the files in source control; or
- duplicate the files.

### Multiple projects in folder structure

It is possible for more than one Oracle Policy Modeling project (and corresponding .xprj file) to co-exist in one location. Each project keeps track of which documents are to be included when compiling the application. Both .xprj files should sit side-by-side in the Development folder (which, in turn, contains the folders Rules, Test Cases, etc.).

With this set-up, only the project file is duplicated, and so any change to any rules or xsrc files will take effect in any project that incorporates them. Individual rules or xsrc files can be [removed from the project](#). Alternatively, files can remain in the project but be [excluded from the build](#).

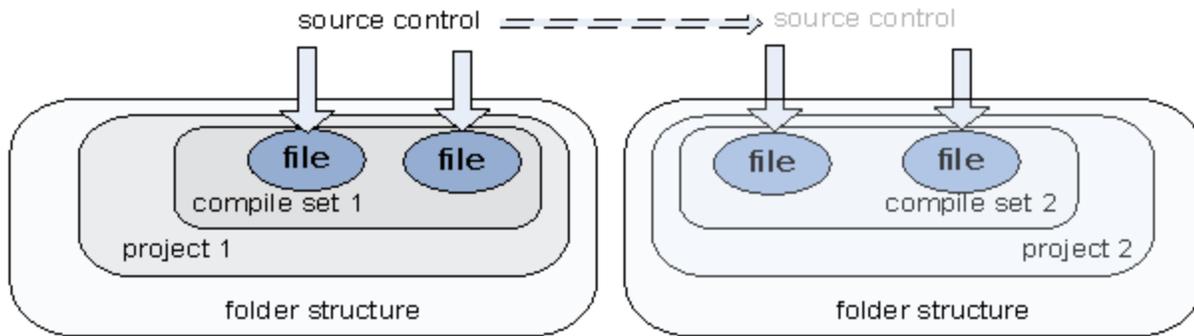
To be in the compile set, a file must be in the project, and to be in the project, a file must be in the folder structure. Source control operates as usual for a single application.



This method of sharing files between projects is particularly suitable where two versions of an application are required (ie a complete version and a "Lite" version). In these situations, the rules and xsrc files are organized into the same structure. The projects differ only by the inclusion or exclusion of the various source files they comprise.

### Share files in source control

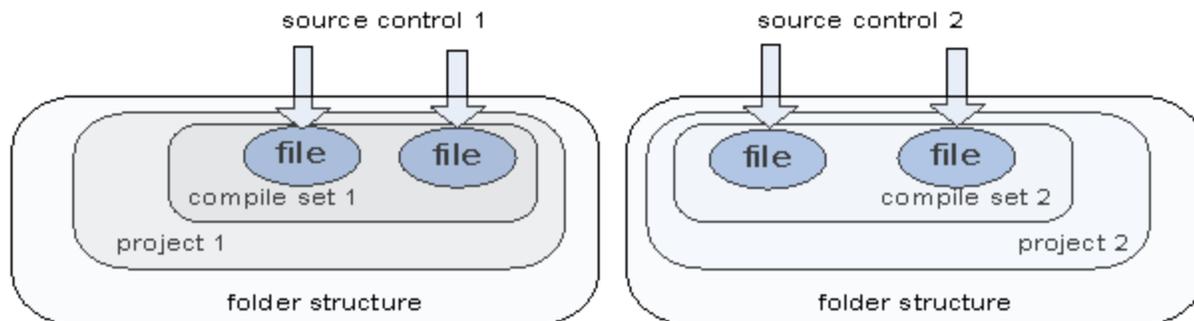
Another method of applying the rules and/or xsrc files in a Oracle Policy Modeling project is to duplicate the files in the file system but synchronize them in source control.



This method of sharing files between projects is suitable where selected rules and/or xsrc files in one main application can be isolated into particular file/s and introduced into another application which shares some of the same subject matter. It is possible for files to become out-of-sync on individual computers under this method, if the source control procedure is not strictly adhered to.

### Duplicate files in source control

Finally, the entire project can be duplicated and maintained separately in both the file system and in source control.



This method does not so much share files between applications, but rather involves a "branch" in the evolution of the project, and is suitable where the source files in one project are useful merely as a starting-point for the other project, for example, where the projects are (at least partly) based on the same source material, but require different analyses for their own purposes. All maintenance and source control occurs separately, since these projects are effectively separate applications once the branching has occurred.

See also

- [Add an existing file to a project](#)

### Use multiple properties files on a multi-developer project

To facilitate multi-developer authoring on a project, it can be helpful to create a more granular properties file structure. By having a separate properties file for each entity type (eg for Person, Income etc) it makes it easier for several developers to be working on the same project at once.

You can do this by using containment relationships. You just need to define the containment relationship in both the 'master' properties file as well as the lower level one. For example:

File	Containment Relationships	Attributes
Master Properties file	Global --> one-to-many --> the person --> one-to-many --> the person's income	Global attributes go in this file
Person Properties file	Global --> one-to-many --> the person	Person attributes go in this file
Income Properties file	Global --> one-to-many --> the person --> one-to-many --> the person's income	Person's income attributes go in this file

Note that while you can define the same attribute/entity/relationship in multiple properties files, only one of them can have non-default [metadata](#)<sup>1</sup>.

See also:

- [Include rules defined in a separate project](#)

## Import a data model from another rule project

You can import a data model from another rule project simply by [adding the properties file](#) from that project to your project. After you have added the existing properties file to your project you should:

1. Build your project straight away. This is because if there are any attributes, entities or relationships that are duplicated, these will be detected in the process and raised as build errors. You should rectify these errors before proceeding to make any further changes to the rulebase.
2. Check to see if there are any attributes or relationships that you don't need. You can use the clean up tool to remove any unused attributes and relationships. To access this tool, select **Tools | Clean Up Unused Attributes and Relationships**. (Unused attributes and relationships are those that have been defined in a properties file but that aren't used in a rule or screen.)

See also:

- [Include rules defined in a separate project](#)

## Track rulebase changes on multi-developer projects

The Oracle Policy Modeling project should be placed under source control whenever it is important to keep a history of changes to the rulebase. Source control should also be used on multi-developer projects so that only one developer may work on a rulebase file at any one time.

To add an existing Oracle Policy Modeling project to source control:

1. In Oracle Policy Modeling, select **File | Source Control | Add Project to <source control>** from the main menu, where <source control> is your installed source control system.
2. In **the Add Project to <source control>** dialog, browse to select the **URL** of the source control repository, and add a new folder name to this URL. Click **OK**.
3. Log in to source control using **Username** and **Password**, then click **OK**.

---

<sup>1</sup>Metadata here refers to custom properties (ie those defined in the custom properties tab of an attribute, entity or relationship) and intrinsic properties (ie non-custom properties that do not relate to the item's text, type or name)

The files in your Oracle Policy Modeling project will be checked into your new source control project, and the source control features in Oracle Policy Modeling will now be automatically available for you to use with your rulebase project files whenever you open the project.

## Subversion and other source control programs

Oracle Policy Modeling integrates with the Subversion source control program, and with any source control program that is accessible via the Source Code Control Application Program Interface (SCCAPI), such as Microsoft SourceSafe, Rational ClearCase, or Microsoft Team Foundation Server. Oracle Policy Modeling will automatically detect which programs you have installed. If you have both Subversion and a SCCAPI program installed, menu options for both systems will be available in Oracle Policy Modeling.

NOTE: If you do not have either of these source control systems installed, the **File | Source Control** menu options will not be available in Oracle Policy Modeling. If you wish to use source control for your project:

- select the Subversion component during the Oracle Policy Modeling installation, or
- install Subversion (note that a command line Subversion client package is required for direct integration with Oracle Policy Modeling), or
- install one of the SCCAPI source control programs. If you have selected Microsoft Team Foundation Server as your SCCAPI program, see [Install Microsoft Team Foundation Server](#) for more information.

See the *Oracle Policy Modeling Installation Guide* in your Oracle Policy Modeling installation folder for further details.

## Install Microsoft Team Foundation Server

To install Microsoft Team Foundation Server:

1. Download and install **Team Explorer for Microsoft Visual Studio 2012** from the Microsoft web site (<http://www.microsoft.com/en-au/download/details.aspx?id=30656>)
2. Download and install either:
  - for 32-bit systems: **Microsoft Visual Studio Team Foundation Server 2012 MSSCCI Provider 32-bit** (available from <http://visualstudiogallery.msdn.microsoft.com/b5b5053e-af34-4fa3-9098-aaa3f3f007cd>) or
  - for 64-bit systems: **Microsoft Visual Studio Team Foundation Server 2012 MSSCCI Provider 64-bit** (available from <http://visualstudiogallery.msdn.microsoft.com/3c7b6813-2617-4d5f-9a1d-5ad980cab5d2>)
3. Once the install has completed, open your project in Oracle Policy Modeling, open the **File** menu and select **Source Control**. Options for Team Foundation Server (such as **Open Existing Project from Team Foundation...** and **Bind Project to Team Foundation...**) should now be available on the Source Control sub-menu.

See also:

- [Track versions of rulebase documents](#)
- [Connect or disconnect a project with source control](#)

## Open a rulebase project from source control

To open an Oracle Policy Modeling project that already exists in a source control repository:

1. In Oracle Policy Modeling, select **File | Source Control | Open Existing Project from <source control>**, where <source control> is your installed source control system.
2. In the **Open Project from <source control>** dialog box, browse to select the **URL** of the project folder in the source control repository. Then browse to select the local **Folder** where you want to save the project. Click **OK**.
3. Log in to source control using **Username** and **Password**, then click **OK**.

4. In the **Open Project** dialog box, select the project file (.xprj) for the rulebase in your local files for the project, then click **Open**.

Once you have initially opened the project from source control, each time you open the project in Oracle Policy Modeling in future it will be automatically connected to source control without you having to repeat the steps above.

See also:

- [Track rulebase changes on multi-developer projects](#)
- [Subversion and other source control programs](#)

## Connect or disconnect a project with source control

Where you have project files on your machine, and that project also exists in source control but is not associated with your own project files, you can bind your Oracle Policy Modeling project to connect it to that existing source control project.

You can disconnect a project from source control by unbinding it.

Note that this option is not available when you are using Subversion as your sole source control system, as binding is managed automatically by Subversion.

### Connect a project

To bind your project to a source control project:

1. In Oracle Policy Modeling, select the project name in Project Explorer.
2. Select **File | Source Control | Bind project to <source control>**, where <source control> is your installed source control system.
3. Log in to source control using **Username** and **Password**, then click **OK**.
4. Select the project to which you want to bind the current project.

### Disconnect a project

To unbind your project from source control:

1. In Oracle Policy Modeling, select the project name in Project Explorer.
2. Select **File | Source Control | Unbind project from <source control>**, where <source control> is your installed source control system.

The project will no longer be managed under source control.

See also:

- [Track rulebase changes on multi-developer projects](#)
- [Subversion and other source control programs](#)

## Track versions of rulebase documents

Placing a rulebase project under source control allows you to track versions of the project documents, by checking files out before making changes, and then checking them back in again to commit those changes to the source-controlled project.

## What do you want to do?

Check out a document from source control

Check in a document to source control

See whether a document is checked in or out

View the version history of a document

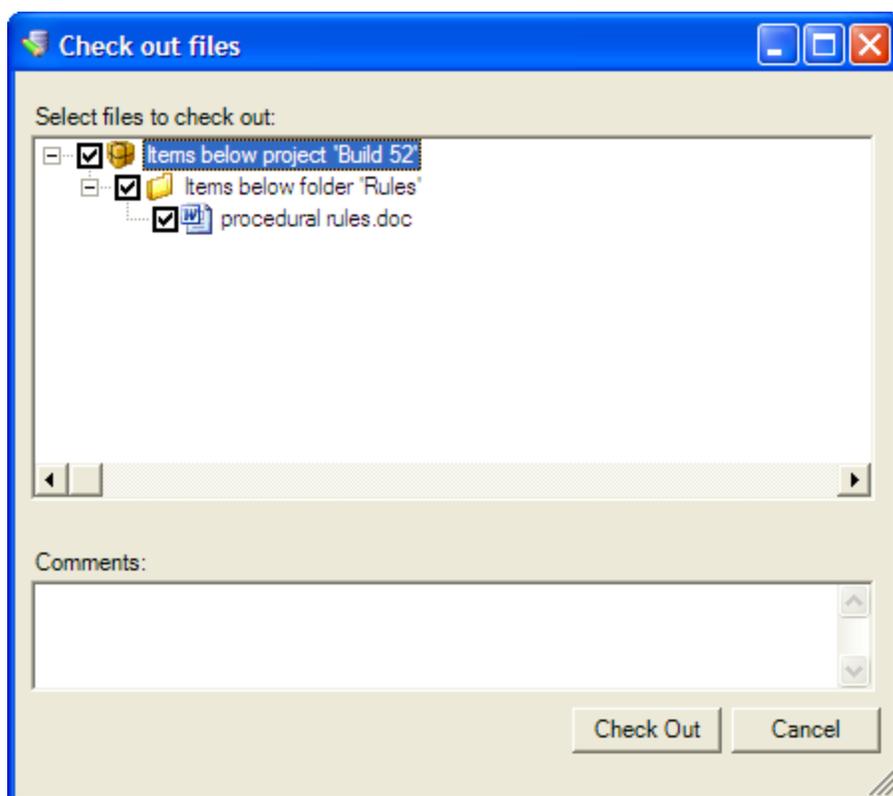
Ensure all documents are checked in when the project is closed

See which documents have not been added to source control

Check out a document from source control

To check out a file:

1. In Oracle Policy Modeling, right-click on the file name in the Project Explorer and select **Check Out**.
2. Select the files to check out and, if required, add a comment.



3. Click the **Check Out** button to check the file out.

4. Log in to source control using **Username** and **Password**, then click **OK**.

You can undo the check out by right-clicking the file name in Project Explorer and clicking the **Undo Check Out** menu option in the pop-up menu.

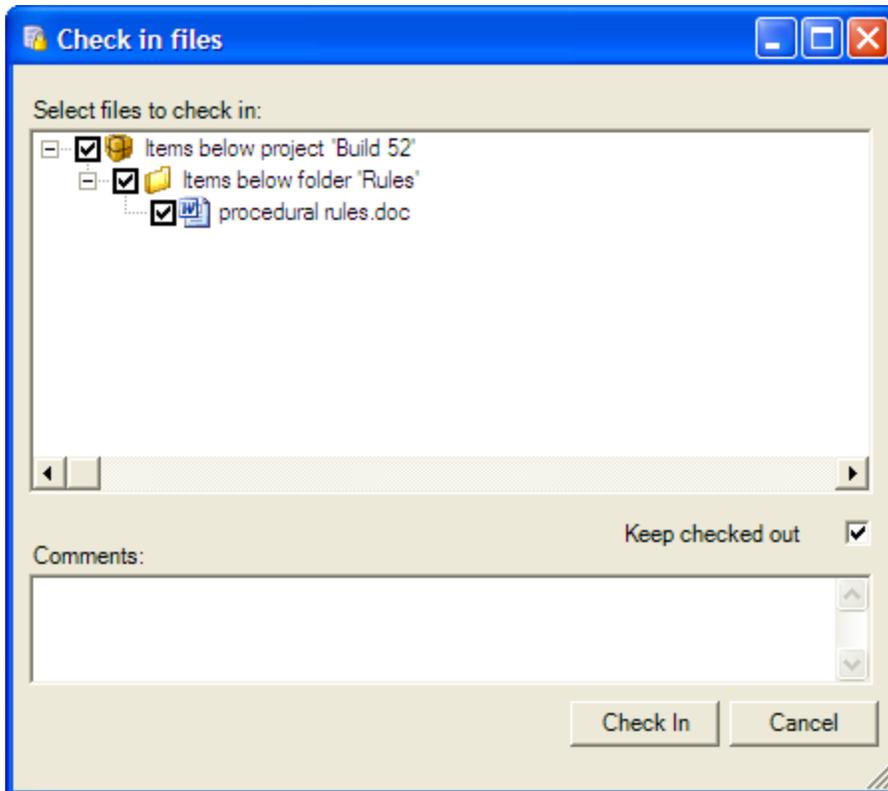
NOTE: Attempting to edit checked-in screen (\*.xint) and properties files (\*.xsrc) will automatically prompt you to check out the file.

TIP: To check out a file for editing, select **Check Out and Edit** from the popup menu in step 1 above. This will check out the file and open it ready for editing (eg in Word, Excel, Oracle Policy Modeling).

### Check in a document to source control

To check in a file:

1. In Oracle Policy Modeling, right-click on the file name in the Project Explorer and select **Check In**.
2. Select the files to check in and, if required, add a comment.



3. If you want to update the master copy of the file while keeping the file checked out, select the **Keep checked out** check box.
4. Click the **Check In** button to check the file in.

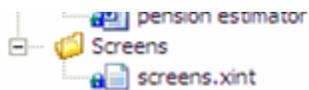
TIP: Files which cannot compile or build should be corrected before being checked into source control.

### See whether a document is checked in or out

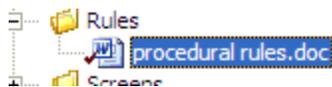
On shared projects, you may want to update the source control status of documents from time to time to see whether or not another team member is working on a project document.

- You can refresh the status of all project documents by right-clicking the project name in the Project Explorer, and selecting **Refresh Source Control Status**.
- You can refresh the status for individual project documents by right-clicking them individually in the Project Explorer, and selecting **Refresh Source Control Status**.

A small blue padlock on a project file icon shows that a file is checked in.



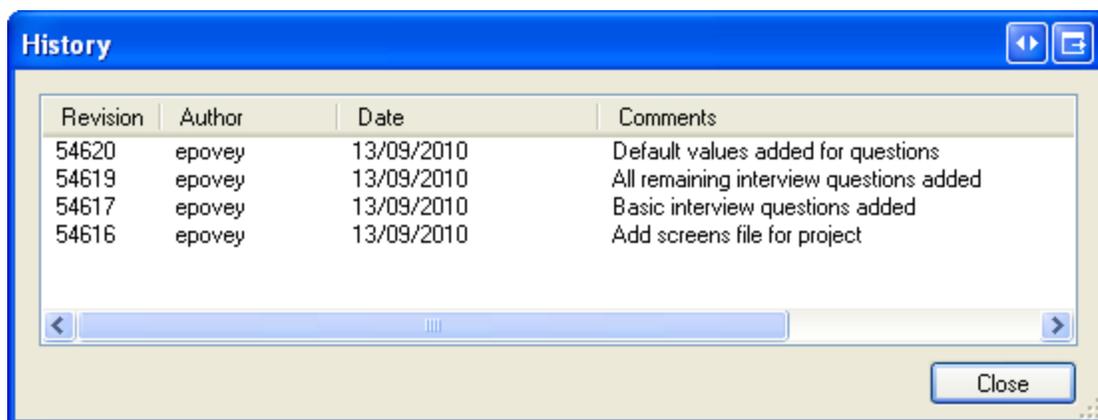
A tick on a project file icon shows that a file is checked out.



### View the version history of a document

Each time you check in a file under source control, a version of that file is kept along with details of the changes made. If you are using Subversion for your project source control, you can view the history of all changes made to a document from within Oracle Policy Modeling. To do this:

1. In Oracle Policy Modeling, right-click on the file name in the Project Explorer and select **Version History**.
2. All changes made to the document are shown, including the date the change was made and any comments entered when the document was checked in.



You can also compare versions of Word rule documents and [retrieve a specific version](#) from the change history. If you are not using Subversion for your project source control, you can access the version history for the document directly from the source control program.

### Ensure all documents are checked in when the project is closed

You can set a reminder for you to check in all of the files you have checked out whenever you close the project. To do this:

1. In Oracle Policy Modeling, select **Tools | Options | Environment | Source Control**.
2. Select the option **When a project is closed remind me to check in all of the files I have checked out**.

### See which documents have not been added to source control

You can also set an option to see all of the files which have not been added to source control whenever you close a source-controlled project. To do this:

1. In Oracle Policy Modeling, select **Tools | Options | Environment | Source Control**.
2. Select the option **When a project is closed show me all of the files that have not been added to source control**.

See also:

- [Track rulebase changes on multi-developer projects](#)

## Get updates to rule documents from source control

You can get updates from source control of single files, all files in a project, or the project file itself.

### Get updates to a single file

To get the latest version of a single file from source control, right-click on the file name in Project Explorer and select **Get Latest Version**.

### Get updates to all files in a project

To get the latest version of the entire project from source control, right-click on the project name in Project Explorer and select **Get Latest Version (Recursive)**.

You can set a reminder to get the latest version of the project files from source control whenever the project is opened. To do this:

1. In Oracle Policy Modeling, select **Tools | Options | Environment | Source Control**.
2. Select the option **When a project is opened ask me if I want to get the latest version of the project files from source control**.

### Get updates to the project file

All Oracle Policy Modeling projects are maintained using a master project file (\*.xprj), which records the file and folder structure of the project. To get this specific file from source control, select the project name in Project Explorer and select **File | Source Control | Get Latest Version of '<project\_name>.xprj'**.

NOTE: **Get Latest Version (Recursive)** will get the master project file as well as all project files from source control.

### Get updates to a file already checked out to you

If you attempt to get the latest version of a document from source control which is checked out to you already, you will be prompted to replace, merge, leave or cancel the operation.

- **Replace:** will replace your current document with the source controlled version.
- **Merge:** will merge differences between the source controlled version and the one on your machine, potentially resulting in unexpected document content. This will not work with Word documents as they are a binary format.
- **Leave:** will not get the file from source control and your current check out file will remain untouched.
- **Cancel:** will cancel the get operation.

## Retrieve a specific document version

Under source control, all historical versions of a document are held since it was first added to source control. This means that you can look at previous versions of a file, and if necessary, replace your working copy of the file with an older version. Oracle Policy Modeling allows you to directly access the document history if you are using the Subversion source control program, otherwise you can use your source control program to access the document history.

## What do you want to do?

[View historical versions of Word rule documents](#)

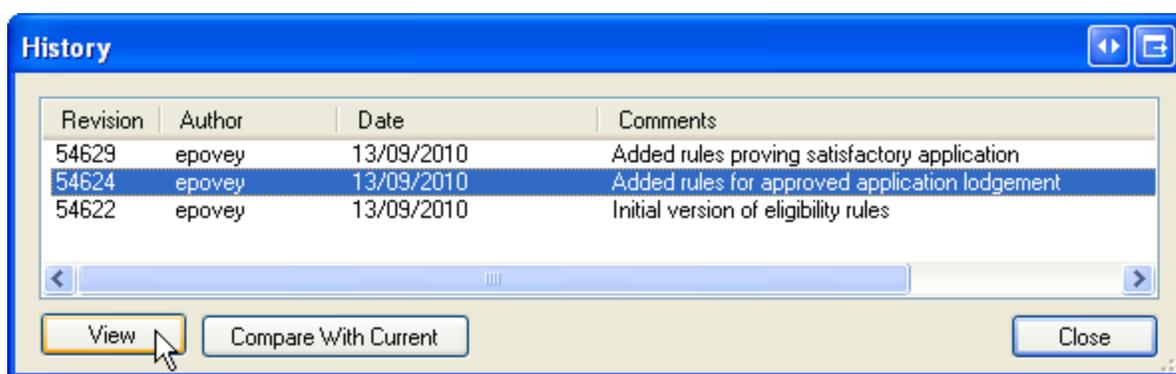
[Compare versions of Word rule documents](#)

[Retrieve versions of other rulebase documents](#)

[View historical versions of Word rule documents](#)

If you are using Subversion for your project source control, you can compare and view historical versions of Word rule documents in Oracle Policy Modeling:

1. In Oracle Policy Modeling, right-click on the file name in the Project Explorer and select **Version History**.
2. Select the version of the document you are interested in and click **View** to open that version.

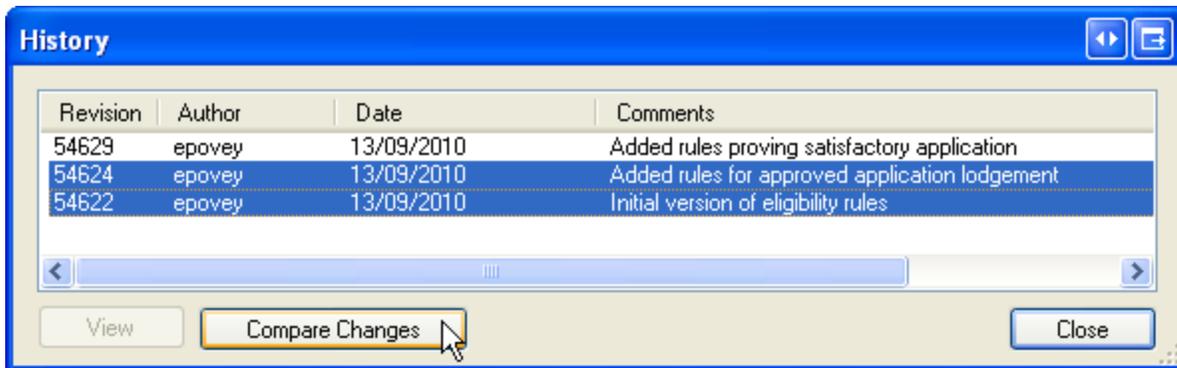


3. If you wish to replace your working copy of the document with the older version, use **Save As** in Word to replace it.

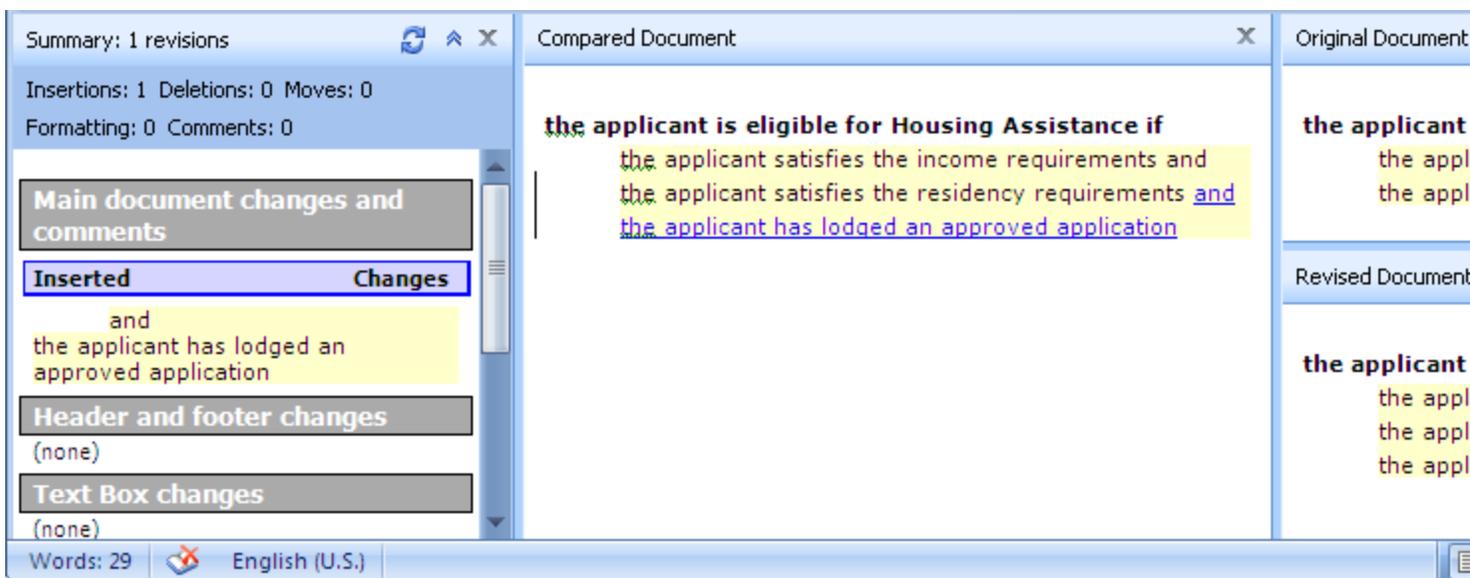
## Compare versions of Word rule documents

If you are using Subversion for your project source control, you can also compare any historical version of a rule document with the current document, or compare two historical versions.

1. In Oracle Policy Modeling, right-click on the file name in the Project Explorer and select **Version History**.
2. Select the version of the document you are interested in and click **Compare With Current** to view the differences between that version and the current version of the document. You can also select two versions of the document by holding down the Control key while clicking to select, then click **Compare Changes** to view the differences between the two selected versions.



3. Word is opened showing the two document versions and highlighting the changes between the two.



### Retrieve versions of other rulebase documents

To retrieve historical versions of project files other than Word rule documents, or if your project source control does not use Subversion, use your source control program to view the log or history for the file. When you have selected the particular version that you are interested in you can either:

- view that file, or
- roll back to that version of the file.

Refer to the Help material in your source control program for more information on these tasks.

### Create multiple rulebase versions

Generally, where you need to maintain an existing rulebase version and also continue development on newer versions (for example, adding new functionality), you need to share and branch the project to create new versions.

To do this, in your source control management tool share the project into a new project, and then branch the project to disconnect all files from the current one. After this, changes made to either project will be independent of one another.

# Integrating

## Topics in "Integrating"

- Set public identifiers for entities and attributes
- Augment the rulebase with metadata
- Build the rulebase from the command line
- Write rules to use in Siebel
- Import a Data Mapping from Siebel

## See also:

- Export or import a data model

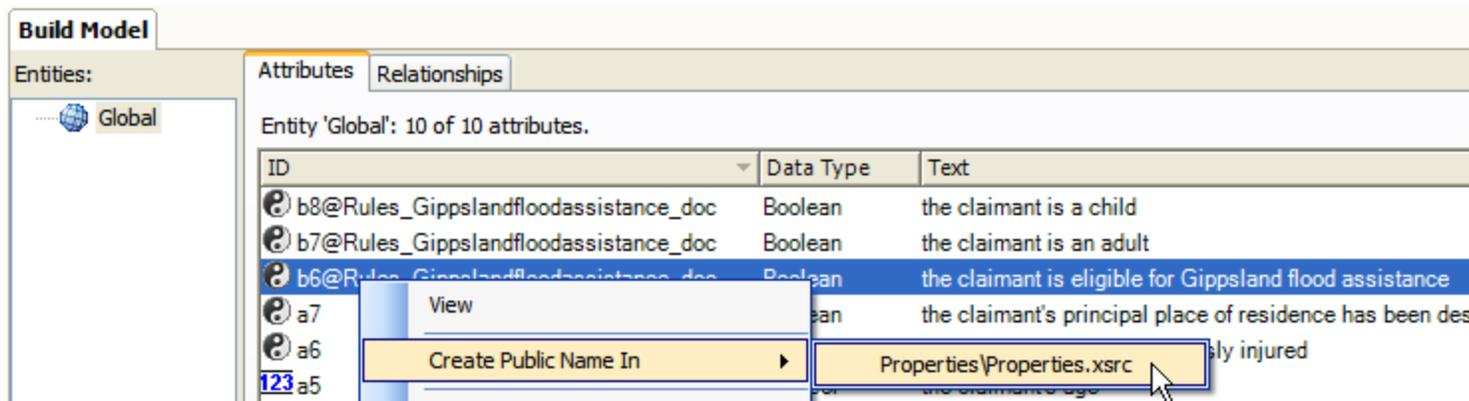
## Set public identifiers for entities and attributes

Important intermediate attributes need public names (user-defined attribute IDs) because this ensures that the attribute IDs for important attributes are reliable and static and can therefore be used by external applications.

Important intermediate attributes are those that may be used in their own right inside Oracle Policy Modeling (eg for regression testing purposes or as labels on screens) or called at runtime (eg called by the Determinations Server, used for document generation, saved with the session data etc).

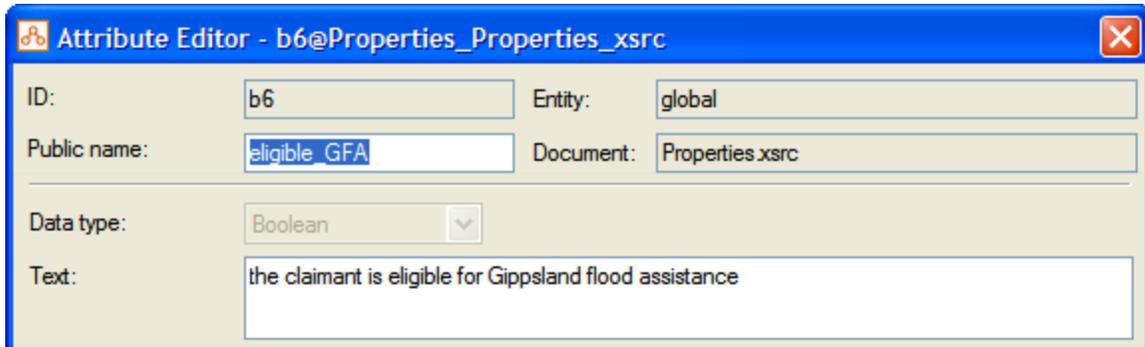
To manually create a public name for a non-base level attribute:

1. In Oracle Policy Modeling, select **View | Build Model**. In the attributes list, select the attribute, right-click and choose **Create Public Name In** and your properties file from the pop-up menu.



The properties file should open to the selected attribute in the **Attribute Editor** dialog.

2. Define a public name for the attribute.



3. Click **OK**. The attribute will now be listed in the properties file with its public name.

NOTE: It is important that the text of any manually created attributes is identical to the original attribute so that automatic linking of attributes can occur. This includes capitalization. In properties files the sentence capitalization remains as entered, whereas in Word and Excel, initial sentence capitals are decapitalized (unless the first two letters both have capitals indicating an acronym). So it is important that all attributes created in properties files start in the lower case (unless starting with an acronym) so that they will match what is compiled out of Word and Excel.

See also:

- [Define attribute names for use by external applications](#)

## Augment the rulebase with metadata

Custom properties are user-defined properties which provide a means of extending or customizing a rulebase by allowing metadata to be associated with any of the following elements: attributes, controls, entities, folders, relationships, rules, screens and the project.

To set up a custom property you need to:

1. Specify the custom property definition in Oracle Policy Modeling. Each property can be given a custom name, default value, data type and can be customized in a number of ways.
2. Assign a value for the custom property for the element.

Custom properties also require application support in order to work.

## What do you want to do?

[Specify a custom property definition](#)

[Assign a custom property to an attribute](#)

[Assign a custom property to an entity](#)

[Assign a custom property to a relationship](#)

[Assign a custom property to a rule](#)

[Assign a custom property to a screen](#)

[Assign a custom property to a control](#)

[Assign a custom property to an interview document](#)

[Assign a custom property to a folder](#)

Assign a custom property to the project

Implement a custom property using application support

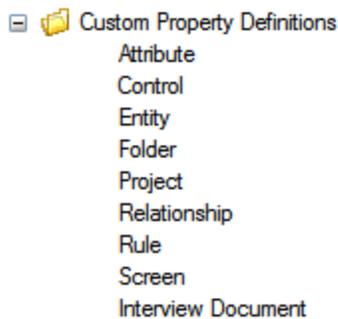
Generate a report of custom properties in a project

Specify a custom property definition

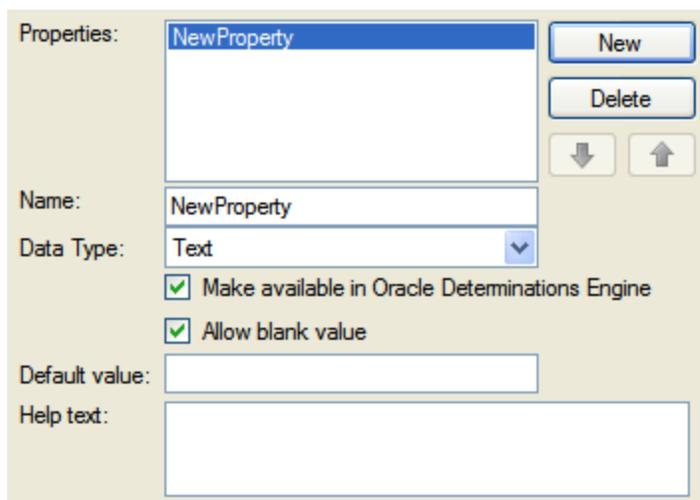
The first step towards setting up a custom property in your rulebase is to specify a custom property definition in Oracle Policy Modeling.

To specify a custom property definition:

1. Select **File | Project Properties** from the main menu.
2. From the list view, select the type of project element for which you wish to define a custom property definition:



3. You can define a new definition for the property by clicking the **New** button. A template will be created for the new custom property.

A screenshot of a form for defining a custom property. The form has a "Properties:" list containing "NewProperty". To the right of the list are "New", "Delete", and two arrow buttons. Below the list, there are fields for "Name:" (containing "NewProperty"), "Data Type:" (a dropdown menu set to "Text"), and two checked checkboxes: "Make available in Oracle Determinations Engine" and "Allow blank value". There are also text input fields for "Default value:" and "Help text:".

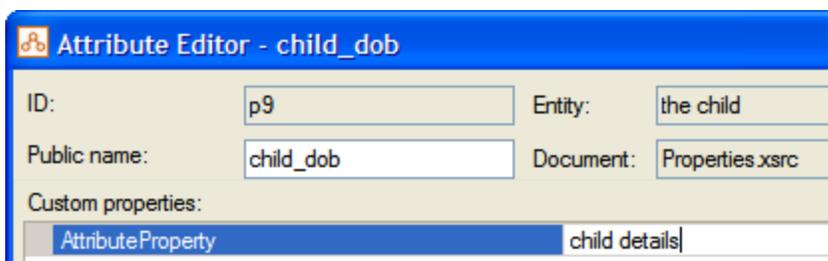
4. Specify a **Name** for the custom property.
5. Select the **Data Type** from the drop-down list (Text, Boolean, Number or List).
6. If you don't want the custom property to be included at runtime, unselect the **Make available in Oracle Determinations Engine** check box.
7. If you want to enforce properties to contain at least some value, unselect the **Allow blank value** check box.

8. Enter a **Default Value** for the custom property if required.
9. Enter **Help Text** for the custom property if required.
10. Save your project to ensure that your custom property definitions are saved.

### Assign a custom property to an attribute

To assign a custom property to an attribute:

1. Open the properties file for the project.
2. Right-click on the attribute and select **Edit Attribute** from the pop-up menu.
3. In the **Attribute Editor** dialog, click the **Custom Properties** tab.
4. Your defined attribute custom properties will be displayed in the **Custom properties** list. Select the property for which you wish to provide a value and enter that value in the right hand text box.



5. Click **OK** to apply the change and save the document.

NOTE: You cannot define a custom property directly on a generated attribute. In order to apply a custom property to an attribute, you must have an equivalent publicly named attribute in your properties file and assign the custom property to that attribute.

Some examples of how custom properties can be used on attributes are:

- to format an attribute or an attribute value. For example, to format an attribute value to be title case or an attribute to appear in bold.
- to identify attributes that are used in generated documents (for example, attributes used in a claim letter)
- to link equivalent attributes in different rulebases
- to map each attribute to a data item in an external application
- to group particular attributes together

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

### Assign a custom property to an entity

To assign a custom property to an entity:

1. Open the properties file for the project.
2. Double-click the entity and in the **Edit Entity** dialog box, click the **Custom Properties** tab.
3. Select the property for which you wish to provide a value and enter that value in the right hand text box.



4. Click **OK** to apply the change.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work. Also, in order for a custom property to be compiled out for an entity, the entity must have a public name.

### Assign a custom property to a relationship

To assign a custom property to a relationship:

1. Open the properties file for the project.
2. In the **Relationships** tab for the entity, right-click on the relationship and select **Edit Relationship...** from the pop-up menu.
3. In the **Relationship Editor** dialog box, click the **Custom Properties** tab.
4. Select the property for which you wish to provide a value and enter that value in the right hand text box.



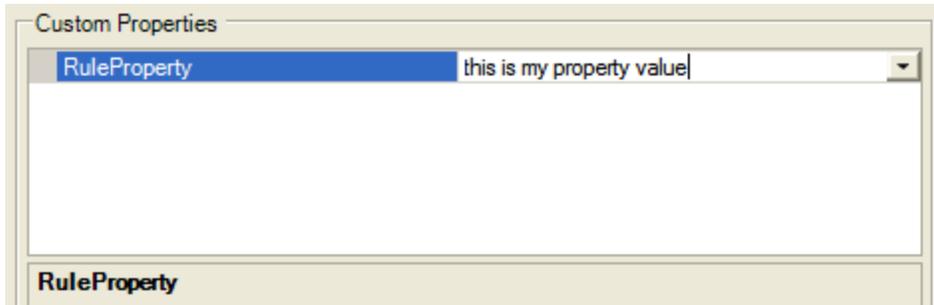
5. Click **OK** to apply the change.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work. Also, in order for a custom property to be compiled out for a relationship, the relationship must have a public name.

### Assign a custom property to a rule

To assign a custom property to a rule:

1. In the Project Explorer in Oracle Policy Modeling, double-click on the rules file to open it in Microsoft Word.
2. Place your cursor at some point in the rule and select the **Rule Properties Editor** button on the Oracle Policy Modeling toolbar.
3. Your defined rule custom properties will be displayed in the Custom Properties list in the Rule Properties dialog box.
4. Select the property for which you wish to provide a value and enter a value in the right-hand text box.



5. Click **OK** to apply the change. You will notice that the custom property now appears above your rule conclusion.

rule\_property[RuleProperty:this is my property value]

**the family is ready to go camping if**

the camping equipment has been packed and  
everyone is in the car

Some examples of how custom properties can be used on rules are:

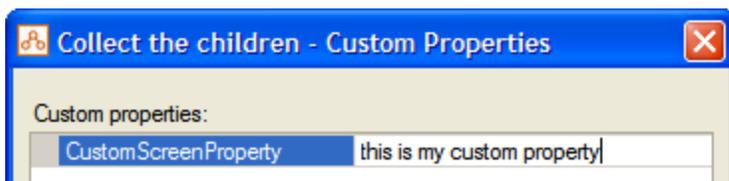
- identifying the owner of rules (eg rule\_property[Maintenance:Business Rule Team])
- identifying the date the rule was last amended (eg rule\_property[Updated:2006-06-17])

Note that Oracle Policy Modeling also provides various [predefined metadata items for rules](#).

**Assign a custom property to a screen**

To assign a custom property to a screen:

1. Open your screens file and right-click the name of the screen for which you wish to add a custom property value.
2. Select **Custom Properties** from the pop-up menu.
3. In the **Custom Properties** dialog box, select the appropriate custom property and enter a value into the right-hand text box.



4. Click **OK** to apply the change and save your screen document.

An example of how custom properties can be used on screens is:

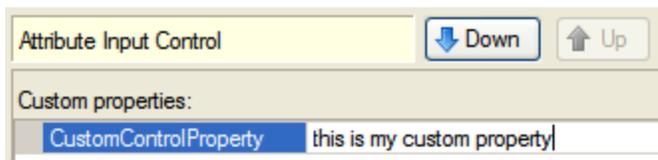
- to make a whole screen display differently. For example, to have a different watermark or size of font from other screens.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

## Assign a custom property to a control

To assign a custom property to a control (screen input):

1. Open the screen edit window for the screen on which your control is defined.
2. Switch to the **Custom Properties** tab.
3. Select the property for which you wish to provide a value and enter a value in the right-hand text box.



4. Click **OK** to apply the change and save your screen document.

Some examples of how custom properties can be used on controls are:

- to control the appearance of inferred attributes on screens
- to make radio buttons appear down the page (rather than the default position of across the page)
- to change commentary based on answers to earlier questions. For example, to substitute the claimant's name in the commentary for a particular control.
- to enable a question based on the answer to another question on the same screen. For example, to only enable the question 'What is the dog's name?' if the user has already answered on that screen that they have a dog.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

## Assign a custom property to an interview document

To assign a custom property to an interview document:

1. Open your screens file and right-click the name of the interview document for which you wish to add a custom property value.
2. Select **Custom Properties** from the pop-up menu.
3. In the **Custom Properties** dialog box, select the appropriate custom property and enter a value into the right-hand text box.



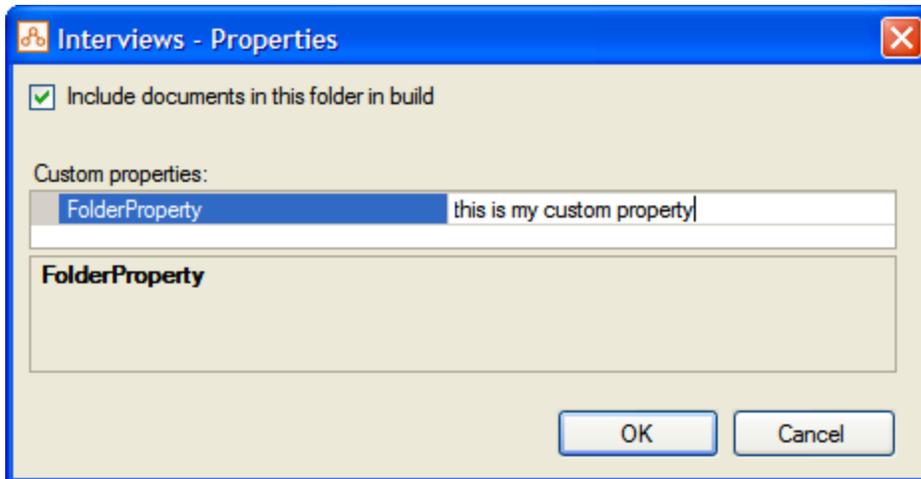
4. Click **OK** to apply the change and save your screen document.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

### Assign a custom property to a folder

To assign a custom property to a folder:

1. Open the Project Explorer in Oracle Policy Modeling.
2. Select the folder you want to assign a custom property to. Right-click and select **Properties...**
3. Select the property for which you wish to provide a value and enter a value in the right-hand text box.



4. Click **OK** to apply the change.

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

### Assign a custom property to the project

To assign a custom property to a project:

1. Select **File | Project Properties | Common Properties | Custom Project Properties.**
2. Your defined project custom properties will be displayed in the **Custom properties** list.
3. Select the property for which you wish to provide a value and enter a value in the right-hand text box.

**Project - Custom Properties**

This page allows you to set values for properties on the project that you have defined in 'Custom Property Definitions'.

Custom properties:

Build Number	20
--------------	----

**Build Number**

OK Cancel Apply

4. Click **OK** to apply the change and save your project.

Some examples of how custom properties can be used on the project are:

- to specify the product version. For example, 'build number 121'.
- to specify the project's release status

NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

### Implement a custom property using application support

The typical process for the implementation of custom properties is:

1. A requirement is identified that is not met by the standard properties and methods provided by the API (eg information on how to format an attribute value)
2. Rule developers and application developers agree on a design to meet that requirement that includes one or more custom properties.
3. Rule developers and application developers agree on the property names and value ranges.

4. Rule developers define the custom properties and set values for the properties, generally in Oracle Policy Modeling. (The exception is rule custom properties which are set in Microsoft Word).
5. Application developers write code to query the custom properties at runtime. They need to know the exact names of the properties to query for and the values to expect.
6. The code does something based on the property value (eg format an attribute value to be title case based on an attribute custom property 'format' having the value 'title\_case').

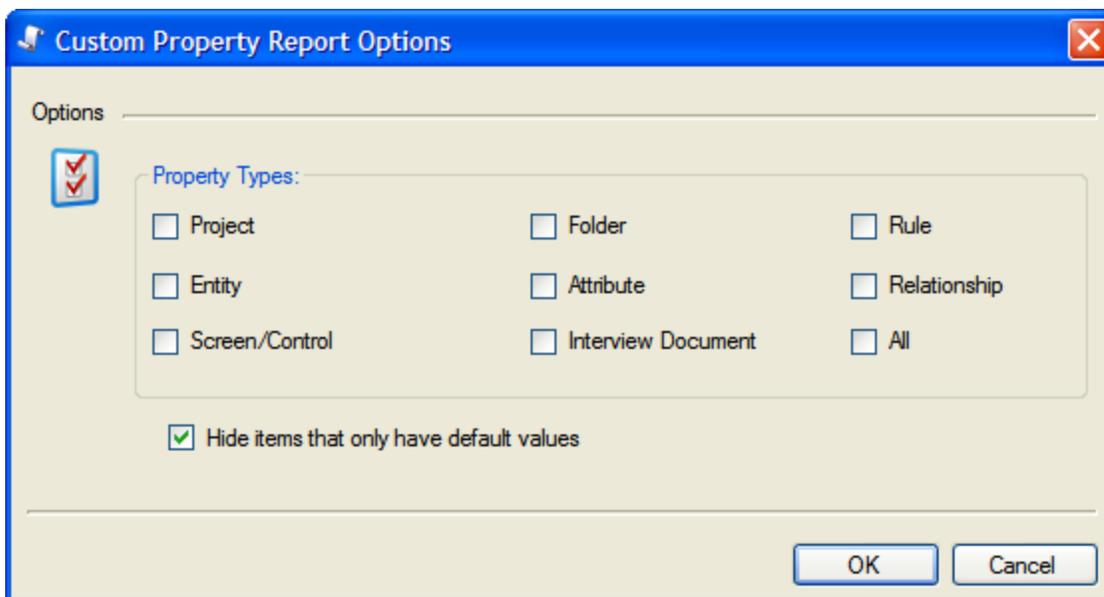
TIP: It does not make sense to use custom properties when rules alone can do the work or when they would require replication of logic in the rulebase. Also, it is not advisable to use custom properties when a change is required globally and there is no likelihood in the future of it being changed again.

### Generate a report of custom properties in a project

There is a Custom Properties report in Oracle Policy Modeling that you can use to generate a report of some or all of the custom properties used in the project.

To run a Custom Properties report:

1. In Oracle Policy Modeling, select **Reports | Custom Properties**. A **Custom Property Report Options** dialog will be shown.



2. Select the property types to display. You also have the option to hide items that only have default values from the report.
3. Click **OK** to generate the report.

### Build the rulebase from the command line

The Oracle Policy Modeling Command Line Compiler provides a means of building a rulebase from an Oracle Policy Modeling project using the command line. This allows the rulebase build process to be automated by including the command in a script.

The tool operates off an Oracle Policy Modeling project file. The project file settings and the documents included in the project are used to build the rulebase. The tool loads the project file, compiles the documents included in the project and builds the rulebase and other output files. The build process performed is the same as using the **Build | Build** menu item in Oracle Policy Modeling.

The build tool may also be used to compile and deploy a rulebase to the Determinations Server. The build and deploy process performed is the same as using the **Oracle Determinations Server** option under the **Build | Build and Run...** menu item in Oracle Policy Modeling.

By default, the tool performs validation on the rulebase model for rule loops and multiply-proven attributes. If the options detailed below are specified, additional validation can be performed. The build will fail if any validation errors are detected.

Projects created in old versions of Oracle Policy Modeling can be upgraded using the tool. Note that the project files will be copied to a backup location to ensure that you have the original version of the project to refer to if necessary. Release folders are not included in the upgrade process. The treatment of entities and their [containment relationships](#) in particular must be brought up to date from older project versions. See [Principles for the upgrading of entities and their containment relationships](#) for more information.

## Syntax

The Oracle Policy Modeling Command Line Compiler is executed from the command line using the following format:

```
buildtoolpath projectpath [build options] [validation options] [report options] [upgrade options] [help options]
```

## Parameters

Parameter	Description
buildtoolpath	The relative or absolute path of the Oracle.Policy.Modeling.CommandLineCompiler.exe file
projectpath	The relative or absolute path of the Oracle Policy Modeling project file to be built
<b>Build Options</b>	
-sb	Recompiles source documents before building the rulebase
-m	Builds the project as a module
-n <build number>	Sets the version number of the built rulebase/module
<b>Validation Options</b>	
-vd	Validates the rulebase model against the data model specified in the Oracle Policy Modeling project
-vds	Validates the rulebase for compatibility with Oracle Determinations Server, notably that all relevant attributes have public names
<b>Report Options</b>	
-cd	Analyzes a *.coverage file and produces a document-oriented report (.xml)
-cg	Analyzes a *.coverage file and produces a goal-oriented report( .xml)
<b>Upgrade Options</b>	
-upgrade	Checks if the project is compatible with the current version.

Parameter	Description
	<p>If it is compatible, it proceeds to compilation.</p> <p>If it needs upgrade, the project is upgraded before being compiled.</p> <p>If it is not compatible (ie the project was created before v9.0), an error is displayed then it exits.</p>
-remReadOnly	<p>Removes write-protection for read-only files. This flag is only valid in the presence of the <i>-upgrade</i> flag.</p> <p>When set, write-protection will be removed for read-only project files.</p> <p>When not set, read-only project files will still be copied to the upgraded project directory but won't be processed.</p>
<b>Help Options</b>	
-h	Prints the help message

### Example

For example, a command to build a project called Eligibility, recompile the source documents and then validate the rulebase model against the data model might look like this:

```
C:\Oracle.Policy.Modeling.CommandLineCompiler.exe C:\Eligibility\Eligibility.xprj -sb -vd
```

### Write rules to use in Siebel

The Oracle Policy Automation Connector for Siebel enables integration between Siebel Applications and the Oracle Policy Automation Determination Server. Once a rulebase has been authored and tested, it can then easily be deployed to Siebel. Using Siebel Administration Screens, the mapping of data between Siebel business components and fields, and rulebase entities and attributes, can then be defined.

For more information on the Oracle Policy Automation Connector for Siebel, see the [Oracle Policy Automation Developer's Guide](#).

### Import a Data Mapping from Siebel

Before you can import the data mapping from Siebel to Oracle Policy Modeling, it must first be exported from Siebel to an XML file so that it is compatible with Oracle Policy Modeling. Once you have the XML file, the data mapping can be imported directly into an Oracle Policy Modeling project.

The following describe the steps you must follow to first Export the data mapping from Siebel to an XML file and then to Import that file into an Oracle Policy Modeling project.

### Export a Data Mapping from Siebel to an XML File

1. Launch **Siebel** and click on the **Administration - Policy Automation** tab. This is where the mappings can be found.
2. Highlight the mapping that you want to export and click on the **Export** button.

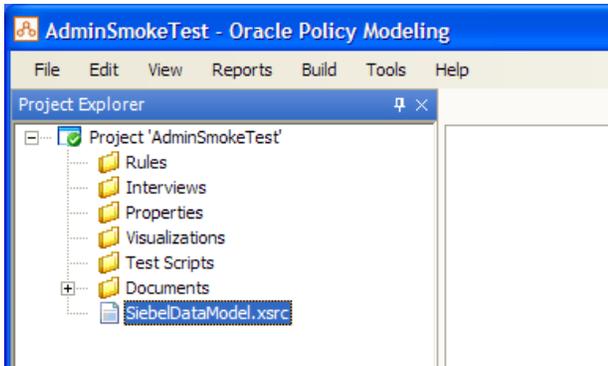
The screenshot shows the Siebel Policy Automation Configuration tool interface. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Navigate', 'Query', 'Tools', and 'Help'. Below the menu bar is a toolbar with various icons. The main area is divided into several sections:

- Entities:** A navigation bar with tabs for 'Home', 'Calendar', 'Cases', 'Contacts', 'Incidents', 'Service', 'Evidence', 'Leads', 'Applications', 'Administration - Policy Automation', and 'Policy Automation'.
- Mappings:** A section with a 'Menu' dropdown and buttons for 'New', 'Edit', 'Delete', 'Query', 'Validate', 'Import', and 'Export'. A version number 'Version 10.1.0.1' is displayed. Below this is a table with columns: 'Mapping Name', 'Business Object', 'Default Value', 'Active Business Ob', and 'Outbound Port'. The table contains four rows: 'AdminSmokeTest' (Employee, Unknown, AdminSmokeTest), 'BPlans' (HLS Case, Unknown, BPlans), 'SSScreening' (HLS Case, Unknown, SSScreening), and 'FrankHLSCase' (HLS Case, Unknown, FrankHLSCase).
- Attributes:** A section with a 'Menu' dropdown and buttons for 'New', 'Edit', 'Delete', and 'Query'. Below this is a table with columns: 'Entity Name', 'Business Component', and 'Query String'. The table contains one row: 'global' (Employee).
- Attributes (bottom):** A section with a 'Menu' dropdown and buttons for 'New', 'Edit', 'Delete', and 'Query'. Below this is a table with columns: 'Attribute Name', 'Data Type', 'Field Name', 'Use Change History', and 'Static Value'. The table contains two rows: 'created' (Auto, Created, Default) and 'firstname' (Auto, First Name, Default).
- Outcomes:** A section with a 'Menu' dropdown and buttons for 'New', 'Edit', 'Delete', and 'Query'. Below this is a table with columns: 'Name' and 'Decision Report Style'. The table contains one row: 'validity\_text' (None).

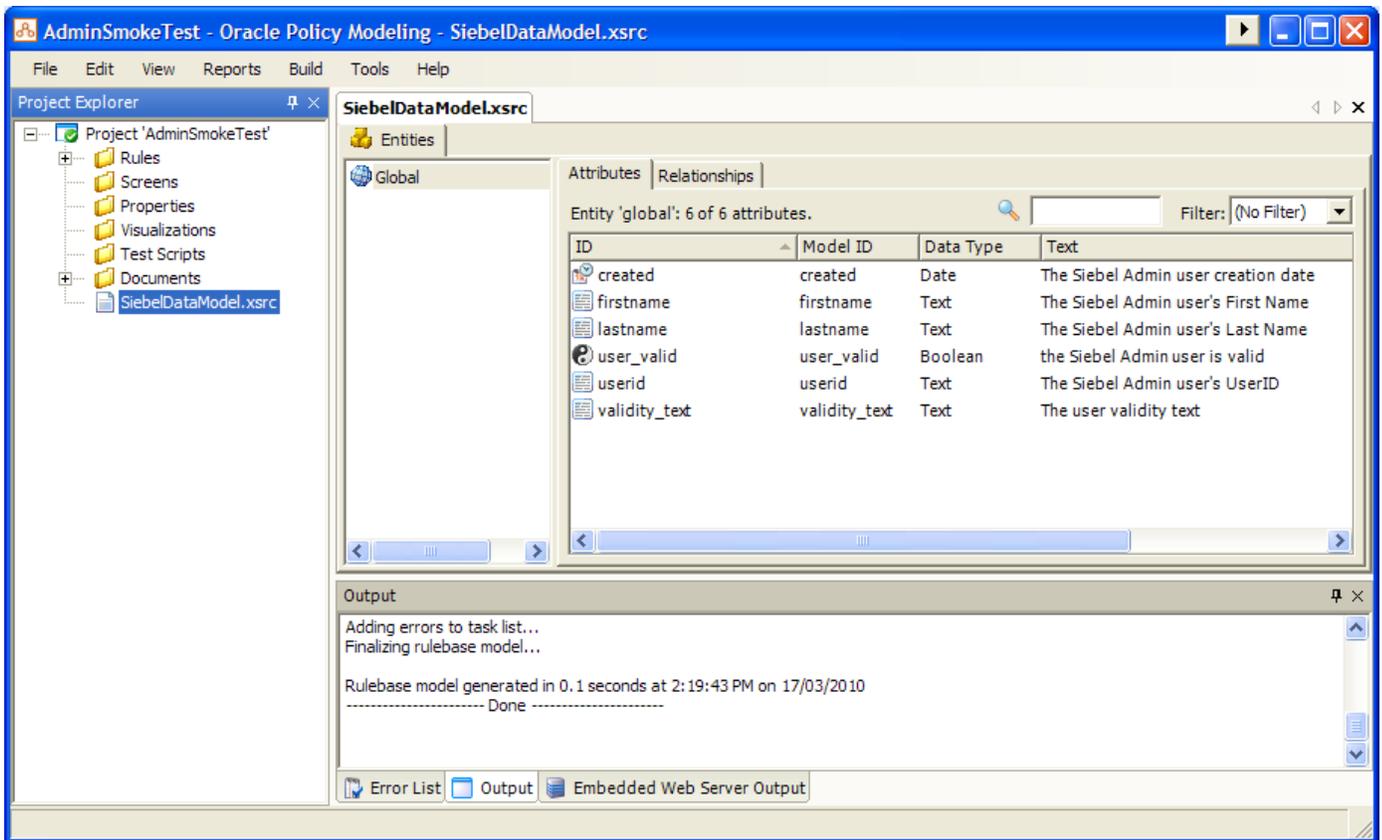
3. In the **Policy Automation Configuration Export confirmation** dialog, click on the **Export** button.
4. In the **File Download** dialog click on the **Save** button to complete the export of the data mapping file from Siebel (do not click on the **Open** button).
5. In the **Save As** dialog, select the location to which you wish to save the file, give the file an appropriate name (for example, name of the Siebel mapping with an *\_Mapping* appended) and then click on the **Save** button .

### Import the Data Mapping into an Oracle Policy Modeling Project:

1. Launch the **Oracle Policy Modeling** application and select **File | New Project...** .
2. In the **New Project** dialog, give the project a name and click on the **Create** button. It is suggested that you use the same name as the data mapping you are importing.
3. From the main menu, select **Tools | Siebel | Import Data Model**.
4. On the **Import Data Model** dialog, locate the *<mapping name>\_Mapping.XML* file and click on the **Open** button to import the data mapping to your project.  
You will notice that a new **SiebelDataModel.xsrc** properties file has been placed in your project; by default, the properties file will always be given that name.



5. Double click on the SiebelDataModel.xsrc properties file to view its contents:



# Accessibility

Topics in "Accessibility"

- [Keyboard shortcuts for Oracle Policy Modeling](#)
- [Modify the appearance or layout of Oracle Policy Modeling](#)
- [Accessibility features in Oracle Web Determinations](#)

## Keyboard shortcuts for Oracle Policy Modeling

Shortcut keys are keys or key combinations that are provided as a quick and alternative way to access frequently performed actions. The following shortcut keys can be used in Oracle Policy Modeling to insert styles or perform functions:

- [Shortcut keys for Oracle Policy Modeling](#)
- [Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Word](#)
- [Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Excel](#)
- [Shortcut keys for the Screen Flow Editor in Oracle Policy Modeling](#)

## Shortcut keys for Oracle Policy Modeling

Shortcut Key	Function / Navigation
Ctrl+N	New Project
Ctrl+O	Open Project
Ctrl+S	Save Selected Item
Ctrl+Shift+S	Save All
Ctrl+F	Find Model Attribute
Ctrl+Shift+F	Find Document Attribute
Ctrl+Shift+B	Build
F5	Build and Debug
Ctrl+F5	Build and Run
Ctrl+Alt+B	Build Module
Ctrl+F4	In the top right hand pane, closes the open tab
Ctrl+>	In the top right hand pane, cycles forwards between the open tabs
Ctrl+<	In the top right hand pane, cycles backwards between the open tabs

Shortcut Key	Function / Navigation
Ctrl+Tab	In the Attribute Editor, toggles between Common, Custom Properties and Decision Reports tabs. In the Summary Screen Editor and Question Screen Editor, toggles between Common and Custom Properties tabs.
Ctrl+F2	In the Project Explorer, toggles between the Project Explorer tab and the Attribute Usage tab
Ctrl+F3	In the Project Explorer, toggles between displaying the active tab (Project Explorer or Attribute Usage) and hiding the tab

### Access menu items in Oracle Policy Modeling

Access keys are provided for all menu items in Oracle Policy Modeling. Access keys are alphanumeric keys that are used with the Alt key to activate the menu controls. The access key is shown by the underlined character in the text label of the menu item. If the access keys are hidden by default, pressing the Alt key will activate them.

### Access shortcut menus in Oracle Policy Modeling

The application key is used to display the shortcut menu for the selected object in Oracle Policy Modeling. The application key is located between the Windows key and the Ctrl key on a standard keyboard. (If your keyboard does not have an application key, you can use Shift+F10 instead.)

Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Word

Shortcut key	Style/Function
Alt+R	Compiles the Oracle Policy Modeling document
Alt+1	Heading style
Alt+2	Heading 2 style
Alt+3	Heading 3 style
Alt+B	Blank Line style
Alt+C	Conclusion style
Alt+F	Configuration style
Alt+L	Legend style
Alt+N	Rule Name style
Alt+F1	Level 1 style
F2	Level 2 style
F3	Level 3 style

Shortcut key	Style/Function
F4	Level 4 style
F5	Level 5 style
F9	Ignore style
F10	Commentary style
F7	Inserts a shortcut rule
F11	Decreases indent
F12	Increases indent
Alt+D	Opens the Data Model Browser
Alt+G	Adds a variable attribute definition to the rulebase
Alt+I	Inserts an invisible operator
Alt+J	Opens the Attribute Editor
Alt+K	Strips hidden text
Alt+P	Opens the Rule Properties editor
Alt+S	Inserts a silent operator
Alt+Y	Show Oracle Policy Modeling styles in style area (Word 2003 and later)
Alt+Z	Inserts a rule table
Alt+F12	Toggles comment

Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Excel

Shortcut key	Style/Function
Ctrl+Shift+C	Compiles the Oracle Policy Modeling document
Ctrl+Shift+W	Attribute Type Heading style
Ctrl+Shift+E	Attribute Text Heading style
Ctrl+Shift+T	Legend Key Heading style
Ctrl+Shift+S	Attribute Type style
Ctrl+Shift+D	Attribute Text style
Ctrl+Shift+G	Legend Key style

Shortcut key	Style/Function
Ctrl+Shift+I	Conclusion Heading style
Ctrl+Shift+K	Conclusion style
Ctrl+Shift+Y	Condition Heading style
Ctrl+Shift+H	Condition style
Ctrl+Shift+L	Else style
Ctrl+Shift+M	Commentary style
Ctrl+Shift+V	Opens the Attribute Editor

### Shortcut keys for the Screen Flow Editor in Oracle Policy Modeling

Shortcut key	Style/Function
Arrow keys	Moves the cursor, if there are no selected shapes; Moves selected shapes
Shift+Arrow keys	Jumps the cursor towards the next shape in that direction
Space	Selects the shape/connection under the cursor; Clears the selection of shapes; In the Screens/Decisions/Flows tab, adds the selected screen/- decision/flow to the screen flow
Ctrl+Arrow keys	Moves the cursor without moving any selected shapes
Alt+Arrow keys	Resizes the selected shape
Ctrl-Space	Toggles the selection of the shape/connection under the cursor
C	Starts or finishes drawing a connector from/to the shape under the cursor
Enter	Finishes drawing a connector to the shape under the cursor; In the Screens/Decisions/Flows tab, adds the selected screen/- decision/flow to the screen flow
/	Cycles the selection through the outgoing connectors of the shape under the cursor
F2	Edits the condition text of the selected connector
Alt+R	Errors list

## Modify the appearance or layout of Oracle Policy Modeling

You can customize the appearance and layout of Oracle Policy Modeling to suit your own preferences.

### What do you want to do?

[Dock/undock the panes](#)

[Pin/unpin the panes](#)

[Resize the panes](#)

[Move the tabs around](#)

[Change the color scheme](#)

[Change the number of items in the recent projects list](#)

### Dock/undock the panes

Any pane in Oracle Policy Modeling can be undocked and then docked in a new location. To undock and redock a pane:

1. Click anywhere on the top of the pane.
2. Drag the pane towards the center of the screen. You will see docking icons appear (at the top, bottom, left and right of the screen).



3. Select the docking icon where you want to dock the pane. The pane will now appear in that location.

NOTE: To be able to undock the panes, the **Lock Windows** option under **Tools | Options | Environment | General** must be unchecked.

### Pin/unpin the panes

Many of the panes can be pinned to the side or bottom of the interface to make more room for your workspace. To pin and unpin a pane:

1. Click on the  button in the top right corner of the pane. The pane will collapse to the side or bottom of the window. To view the pane you can hover over the tab on the side/bottom of the window.
2. To unpin the pane, click on the tab for the pane and then click on the  button. The pane will return to its previous location in the interface.

### Resize the panes

To resize a pane:

1. Move your cursor over the join between panes until you see a double headed arrow.
2. Use your mouse to drag the join until the pane is the desired size.

### Move the tabs around

In any of the panes in Oracle Policy Modeling, you can move the tabs around. To do this:

1. Click on the tab you want to move.
2. Drag it to the new location on the same pane.

## Change the color scheme

To change the color scheme you can change the visual style of Oracle Policy Modeling. To do this:

1. In Oracle Policy Modeling, select **Tools | Options | Environment | General**.
2. Select a different option from the **Visual style** drop-down list.
3. Click **Apply** to see what the new visual style looks like.
4. Click **OK**.

## Change the number of items in the recent projects list

To change the number of items in the recent projects list in the **File** menu:

1. In Oracle Policy Modeling, select **Tools | Options | Environment | General**.
2. Change the number in the **Display** field to the number of items you want displayed.

## Accessibility features in Oracle Web Determinations

The default Oracle Web Determinations (OWD) user interface contains a number of accessibility features. (For more information on OWD, see [Deploy an interview to Web Determinations](#)).

### Keyboard-only navigation

One of the key aspects of accessibility in OWD is keyboard-only navigation for interviews, without the need to use a mouse. The primary method of navigation is the **Tab** key.

Note: if you have made your own modifications to the style sheets used for an OWD interview (see [Customize Oracle Web Determinations](#)), some of the keyboard-only navigation techniques described in this topic may not work. You will need to perform your own checks to ensure that your style sheet modifications do not compromise the accessibility of your application.

### General principles

When you first launch an OWD interview, the focus will be on the URL pane of your web browser. Hit the **Tab** key to navigate one at a time through the controls and links on the screen. The focus of the cursor will follow a top-down, left to right order. To navigate in reverse order, use **Shift + Tab**.

When the cursor focus is on a control you wish to activate or a link you wish to follow, hit the **Enter** key.

### Interview screens with input controls

On interview screens with input controls, after hitting the **Tab** key once the cursor focus will be on the top-most input control.

- For text boxes, simply type text into the box once the cursor focus is on it.
- For input controls with pre-defined sets of valid inputs, such as radio buttons and drop-down lists, you can use the right and left arrows to navigate between the different pre-defined options.

When you are ready to move to the next input control, hit the **Tab** key.

When you have completed entering data on an interview screen, move the cursor focus to the **Submit** button, then hit the **Enter** key.

### Decision Report screens

On a Decision Report screen (reached by clicking the word '**Why**' on the summary screen at the conclusion of an interview):

- use the **Tab** key to navigate to tree nodes within the decision report;
- expand a node using the right arrow key; and
- collapse a node using the left arrow key.

#### Data Review screens

On the Data Review screen (reached by clicking the **Data Review** link from any OWD screen):

- use the **Tab** key to navigate to different screens on the list;
- expand a screen (or a group of screens if they are grouped into folders) using the right arrow key;
- collapse screens and screen groups using the left arrow key; and
- to revisit a particular screen, move the cursor focus (using the **Tab** key) to the screen's name, then hit the **Enter** key.

## Reference

### Topics in "Reference"

- [Rule syntax reference](#)
- [Rule function examples](#)
- [File extensions](#)
- [Truth tables](#)
- [Basic English grammar](#)
- [Rule principles for Oracle Policy Modeling](#)
- [Text substitution principles](#)
- [Value conditions for screen flow connections](#)
- [BI Publisher code for Oracle Policy Modeling](#)
- [Troubleshooting guide for using BI Publisher with Oracle Policy Modeling](#)
- [Seeded data in imported projects](#)
- [Keyboard shortcuts for Oracle Policy Modeling](#)
- [Formatting of attribute values](#)
- [Command line tools](#)

### Rule syntax reference

#### Topics in "Rule syntax reference"

- [Function reference \(US English\)](#)
- [Function reference \(all languages\)](#)
- [Structural configuration settings](#)

## Logical connectors

Syntax	Description
<b>if</b>	Optional term that can appear at the end of a conclusion line that has a following proof
<b>and</b>	Logical conjunction between two attributes
<b>or</b>	Logical disjunction between two attributes
<b>either one of any at least one of the following is true</b>	Grouping element used with disjunctions where two or more attributes need to be grouped

Syntax	Description
<b>any of the following are satisfied</b>	
<b>both</b> <b>all</b> <b>all of the following are true</b> <b>all of the following are satisfied</b>	Grouping element used with conjunctions where two or more attributes need to be grouped
<b>otherwise</b>	Term that appears at the end of a table rule to indicate the otherwise clause
<b>is</b>	Term that is used in a legend entry between the abbreviated phrase and the full attribute text

## Logical functions

Syntax	Description
<b>it is not true that</b> <i>&lt;expr&gt;</i>	Operator used to return true if attribute has a value which is false
<i>&lt;var&gt;</i> <b>is certain</b> <b>it is certain whether</b> [ <b>or not</b> ] <i>&lt;expr&gt;</i>	Operator used to return true if attribute has a value which is not uncertain
<i>&lt;var&gt;</i> <b>is uncertain</b> <i>&lt;var&gt;</i> <b>is not certain</b> <b>it is uncertain that</b> <i>&lt;expr&gt;</i> <b>it is uncertain whether</b> [ <b>or not</b> ] <i>&lt;expr&gt;</i> <b>it is not certain that</b> <i>&lt;expr&gt;</i>	Operator used to return true if attribute value is uncertain
<i>&lt;var&gt;</i> <b>is known</b> <i>&lt;var&gt;</i> <b>is currently known</b> <b>it is known whether</b> [ <b>or not</b> ] <i>&lt;expr&gt;</i> <b>it is currently known whether</b> [ <b>or not</b> ] <i>&lt;expr&gt;</i>	Operator used to return true if attribute has any value
<i>&lt;var&gt;</i> <b>is</b> [ <b>currently</b> ] <b>unknown</b> <b>it is</b> [ <b>currently</b> ] <b>unknown whether</b> [ <b>or not</b> ] <i>&lt;expr&gt;</i>	Operator used to return true if attribute has no value

## Logical constants

Syntax	Description
<b>true</b>	Constant true value used for table rules.
<b>false</b>	Constant false value used for table rules.
<b>uncertain</b>	Constant uncertain value used for table rules.

## Comparison operators

Syntax	Description
<x><y> <x> <b>is earlier than</b> <y>	Less than <b>Note:</b> there is no natural language form when this operator is used with numerical and currency values.
<x> > <y> <x> <b>is later than</b> <y>	Greater than <b>Note:</b> there is no natural language form when this operator is used with numerical and currency values.
<x><= <y> <x> <b>is less than or equal to</b> <y> <x> <b>is on or earlier than</b> <y> <x> <b>is at or earlier than</b> <y>	Less than or equal to
<x> >= <y> <x> <b>is greater than or equal to</b> <y> <x> <b>is on or later than</b> <y> <x> <b>is at or later than</b> <y>	Greater than or equal to
<x>= <y> <x> <b>is equal to</b> <y> <x> <b>equals</b> <y>	Equals
<x> <b>is not equal to</b> <y> <x> <> <y>	Not equal

## Numerical functions

Syntax	Description
<b>Number</b> (<numText>)	Convert the specified string into a <b>number</b> value
<x> + <y>	Mathematical addition
<x> - <y>	Mathematical subtraction
<x> * <y>	Mathematical multiplication
<x> / <y>	Mathematical division
<x> \ <y>	Integer division
<x> <b>modulo</b> <y>	Remainder after integer division
<b>Maximum</b> (<x>, <y>) <b>Maximum</b> (<date/time/datetime1>, <date/time/datetime2>) <b>the greater of &lt;x&gt; and &lt;y&gt;</b> <b>the latest of &lt;x&gt; and &lt;y&gt;</b>	Returns the greater of two values

Syntax	Description
<b>Minimum</b> (<x>, <y>) <b>Minimum</b> (<date/time/datetime1>, <date/time/datetime2>) <b>the lesser of &lt;x&gt; and &lt;y&gt;</b> <b>the earliest of &lt;x&gt; and &lt;y&gt;</b>	Returns the lesser of two values
<b>Xy</b> (<x>, <y>) <b>&lt;x&gt; raised to the power of &lt;y&gt;</b>	x to the power of y
<b>Ex</b> (<x>) <b>e to the power of &lt;x&gt;</b>	Constant e to the power of x
<b>Abs</b> (<x>) <b>the absolute value of &lt;x&gt;</b>  <x>	Absolute value of x
<b>Ln</b> (<x>) <b>the natural logarithm of &lt;x&gt;</b>	Natural logarithm of x
<b>Log</b> (<x>) <b>the logarithm base 10 of &lt;x&gt;</b>	Logarithm base 10 of x
<b>Sqrt</b> (<x>) <b>the square root of &lt;x&gt;</b>	Square root of x
<b>Round</b> (<x>, <n>) <x> rounded to <n> decimal place <x> rounded to <n> decimal places	Rounds x to n decimal places
<b>Trunc</b> (<x>, <n>) <x> truncated to <n> decimal place <x> truncated to <n> decimal places	x truncated to n decimal places
<b>Sin</b> (<x>)	Sine of x
<b>Cos</b> (<x>)	Cosine of x
<b>Tan</b> (<x>)	Tangent of x
<b>Asin</b> (<x>)	Arcsine of x
<b>Acos</b> (<x>)	Arccosine of x
<b>Atan</b> (<x>)	Arctangent of x

## Date functions

Syntax	Description
<b>CurrentDate</b> ()	Returns the current date at the start of the session.

Syntax	Description
<b>the current date</b>	
<b>Date</b> ( <i>&lt;text&gt;</i> )	Converts the specified string into a date value
<b>MakeDate</b> ( <i>&lt;year&gt;</i> , <i>&lt;month&gt;</i> , <i>&lt;day&gt;</i> )	Returns a date formed from the specified year, month, and day.
<b>ExtractDay</b> ( <i>&lt;date/datetime&gt;</i> )	Returns the day component of a date/datetime attribute.
<b>ExtractMonth</b> ( <i>&lt;date/datetime&gt;</i> )	Returns the month component of a date/datetime attribute.
<b>ExtractYear</b> ( <i>&lt;date/datetime&gt;</i> )	Returns the year component of a date/datetime attribute.
<b>NextDayOfTheWeek</b> ( <i>&lt;date/datetime&gt;</i> , <i>&lt;day&gt;</i> ) <b>the next Monday on or after</b> <i>&lt;from-date&gt;</i> <b>the Monday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Tuesday on or after</b> <i>&lt;from-date&gt;</i> <b>the Tuesday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Wednesday on or after</b> <i>&lt;from-date&gt;</i> <b>the Wednesday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Thursday on or after</b> <i>&lt;from-date&gt;</i> <b>the Thursday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Friday on or after</b> <i>&lt;from-date&gt;</i> <b>the Friday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Saturday on or after</b> <i>&lt;from-date&gt;</i> <b>the Saturday on or before</b> <i>&lt;from-date&gt;</i> <b>the next Sunday on or after</b> <i>&lt;from-date&gt;</i> <b>the Sunday on or before</b> <i>&lt;from-date&gt;</i>	Returns the date of the next weekday on or before/after a date (depending on the syntax used).
<b>NextDate</b> ( <i>&lt;date&gt;</i> , <i>&lt;day&gt;</i> ,	Returns the next instance of the given day and month after a date.

Syntax	Description
<p>&lt;month&gt;)  <b>the previous UK tax year start date on or before</b> &lt;from-date&gt;  <b>the next UK tax year end date on or after</b> &lt;from-date&gt;</p>	<p>Returns the start date for the previous UK tax year (6 April), relative to date.  Returns the end date for the next UK tax year (5 April), relative to date.</p>
<p><b>AddDays</b>( &lt;date/datetime&gt;, &lt;num_days&gt;)  <b>the date</b> &lt;num_days&gt; <b>days after</b> &lt;datetime&gt;  <b>the date</b> &lt;num_days&gt; <b>days before</b> &lt;datetime&gt;  <b>the date</b> &lt;num_days&gt; <b>day after</b> &lt;datetime&gt;  <b>the date</b> &lt;num_days&gt; <b>day before</b> &lt;datetime&gt;  <b>the time</b> &lt;num_days&gt; <b>days after</b> &lt;datetime&gt;  <b>the time</b> &lt;num_days&gt; <b>days before</b> &lt;datetime&gt;  <b>the time</b> &lt;num_days&gt; <b>day after</b> &lt;datetime&gt;  <b>the time</b> &lt;num_days&gt; <b>day before</b> &lt;datetime&gt;</p>	<p>Adds/subtracts a number of days to a date. When using the terse syntactic form, the number must be a positive integer in order to add days to the input date, or a negative number in order to subtract days from the input date.</p>
<p><b>AddWeeks</b>( &lt;date/datetime&gt;, &lt;num_weeks&gt;)  <b>the date</b> &lt;num_weeks&gt; <b>weeks after</b> &lt;datetime&gt;  <b>the date</b> &lt;num_weeks&gt; <b>weeks before</b> &lt;datetime&gt;  <b>the date</b> &lt;num_weeks&gt; <b>week after</b> &lt;datetime&gt;  <b>the date</b> &lt;num_weeks&gt; <b>week before</b> &lt;datetime&gt;  <b>the time</b> &lt;num_weeks&gt; <b>weeks after</b> &lt;datetime&gt;  <b>the time</b> &lt;num_weeks&gt; <b>weeks before</b> &lt;datetime&gt;  <b>the time</b> &lt;num_weeks&gt; <b>week after</b> &lt;datetime&gt;  <b>the time</b> &lt;num_weeks&gt; <b>week before</b> &lt;datetime&gt;</p>	<p>Adds a number of weeks to a date. When using the terse syntactic form, the number must be a positive integer in order to add weeks to the input date.</p>

Syntax	Description
<b>AddMonths</b> ( <date/datetime>, <num_months>) <b>the date</b> <num_months> <b>months after</b> <datetime> <b>the date</b> <num_months> <b>months before</b> <datetime> <b>the date</b> <num_months> <b>month after</b> <datetime> <b>the date</b> <num_months> <b>month before</b> <datetime> <b>the time</b> <num_months> <b>months after</b> <datetime> <b>the time</b> <num_months> <b>months before</b> <datetime> <b>the time</b> <num_months> <b>month after</b> <datetime> <b>the time</b> <num_months> <b>month before</b> <datetime>	<p>Adds a number of months to a date. When using the terse syntactic form, the number must be a positive integer in order to add months to the input date.</p>
<b>AddYears</b> ( <date/datetime>, <num_years>) <b>the date</b> <num_years> <b>years</b> <b>after</b> <datetime> <b>the date</b> <num_years> <b>years</b> <b>before</b> <datetime> <b>the date</b> <num_years> <b>year</b> <b>after</b> <datetime> <b>the date</b> <num_years> <b>year</b> <b>before</b> <datetime> <b>the time</b> <num_years> <b>years</b> <b>after</b> <datetime> <b>the time</b> <num_years> <b>years</b> <b>before</b> <datetime> <b>the time</b> <num_years> <b>year</b> <b>after</b> <datetime> <b>the time</b> <num_years> <b>year</b> <b>before</b> <datetime>	<p>Adds a number of years to a date. When using the terse syntactic form, the number must be a positive integer in order to add years to the input date.</p>
<b>WeekdayCount</b> ( <date1>, <date2>) <b>the number of weekdays</b> <b>(inclusive) between</b> <date1> <b>and</b> <date2>	<p>Counts the number of weekdays between date1 and date2. That is, the number of days falling between Monday and Friday.            Note: The earlier date is inclusive and the later date is exclusive.</p>
<b>YearStart</b> ( <date/datetime>)	<p>Returns the first date in the year in which a date falls.</p>

Syntax	Description
<b>the first day of the year in which &lt;from-date&gt; falls</b>	
<b>YearEnd</b> ( <date/datetime>) <b>the last day of the year in which &lt;from-date&gt; falls</b>	Returns the last date in the year in which a date falls.
<b>DayDifference</b> ( <date/d-atetime1>, <date/datetime2>) <b>the number of days from &lt;date/datetime1&gt; to &lt;date/d-atetime2&gt;</b>	Returns the number of whole days between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
<b>DayDifferenceInclusive</b> ( <date/datetime1>, <date/d-atetime2>) <b>the number of days (inclusive) from &lt;date/datetime1&gt; to &lt;date/datetime2&gt;</b>	Returns the number of whole days (inclusive) between date/datetime1 and date/d-atetime2. This calculation includes both endpoints. Where the dates are the same, the result is 1. The order of the two dates does not affect the result.
<b>DayDifferenceExclusive</b> ( <date/datetime1>, <date/d-atetime2>) <b>the number of days (exclusive) from &lt;date/datetime1&gt; to &lt;date/datetime2&gt;</b>	Returns the number of whole days (exclusive) between date/datetime1 and date/d-atetime2. This calculation excludes both endpoints. Where the dates are the same, the result is 0. The order of the two dates does not affect the result.
<b>WeekDifference</b> ( <date/d-atetime1>, <date/datetime2>) <b>the number of weeks from &lt;date/datetime1&gt; to &lt;date/d-atetime2&gt;</b>	Returns the number of whole elapsed weeks between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
<b>WeekDifferenceInclusive</b> ( <date/datetime1>, <date/d-atetime2>) <b>the number of weeks (inclusive) from &lt;date/datetime1&gt; to &lt;date/datetime2&gt;</b>	Returns the inclusive number of whole elapsed weeks between date/datetime1 and date/d-atetime2. The order of the two dates does not affect the result.
<b>WeekDifferenceExclusive</b> ( <date/datetime1>, <date/d-atetime2>) <b>the number of weeks (exclusive) from &lt;date/datetime1&gt; to &lt;date/datetime2&gt;</b>	Returns the exclusive number of whole elapsed weeks between date/datetime1 and date/d-atetime2. The order of the two dates does not affect the result.

Syntax	Description
<b>MonthDifference</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of months from</b> <i>&lt;date/datetime1&gt;</i> <b>to</b> <i>&lt;date/datetime2&gt;</i>	Returns the number of whole elapsed months between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.
<b>MonthDifferenceInclusive</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of months (inclusive) from</b> <i>&lt;date/datetime1&gt;</i> <b>to</b> <i>&lt;date/datetime2&gt;</i>	Returns the number of whole inclusive elapsed months between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.
<b>MonthDifferenceExclusive</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of months (exclusive) from</b> <i>&lt;date/datetime1&gt;</i> <b>to</b> <i>&lt;date/datetime2&gt;</i>	Returns the number of whole exclusive elapsed months between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.
<b>YearDifference</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of whole years which</b> <i>&lt;date/datetime2&gt;</i> <b>is after</b> <i>&lt;date/datetime1&gt;</i> <b>the number of years between</b> <i>&lt;date/datetime1&gt;</i> <b>and</b> <i>&lt;date/datetime2&gt;</i>	Returns the number of years between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.
<b>YearDifferenceInclusive</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of years (inclusive) between</b> <i>&lt;date/datetime1&gt;</i> <b>and</b> <i>&lt;date/datetime2&gt;</i>	Returns the inclusive number of years between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.
<b>YearDifferenceExclusive</b> ( <i>&lt;date/datetime1&gt;</i> , <i>&lt;date/datetime2&gt;</i> ) <b>the number of years (exclusive) between</b> <i>&lt;date/datetime1&gt;</i> <b>and</b> <i>&lt;date/datetime2&gt;</i>	Returns the exclusive number of years between <i>date/datetime1</i> and <i>date/datetime2</i> . The order of the two dates does not affect the result.

## Time of day functions

Syntax	Description
<b>TimeOfDay</b> ( <i>&lt;text&gt;</i> )	Converts the given string into a time of day
<b>ExtractSecond</b> ( <i>&lt;time/datetime&gt;</i> )	Returns the second component of a timeofday/datetime attribute.
<b>ExtractMinute</b> ( <i>&lt;time/datetime&gt;</i> )	Returns the minute component of a timeofday/datetime attribute.
<b>ExtractHour</b> ( <i>&lt;time/datetime&gt;</i> )	Returns the hour component of a timeofday/datetime attribute.

## Date and time functions

Syntax	Description
<b>CurrentDateTime</b> <b>the current date time</b>	Returns the current date and time at the start of the session.
<b>DateTime</b> ( <i>&lt;text&gt;</i> )	Converts the specified string into a datetime value
<b>ConcatenateDateTime</b> ( <i>&lt;date&gt;</i> , <i>&lt;time&gt;</i> ) <i>&lt;date&gt;</i> <b>at</b> <i>&lt;time-of-day&gt;</i> <i>&lt;time-of-day&gt;</i> <b>on</b> <i>&lt;date&gt;</i>	Sets the date time by joining the date and time of day together.
<b>SecondDifference</b> ( <i>&lt;datetime1&gt;</i> , <i>&lt;datetime2&gt;</i> ) <b>SecondDifference</b> ( <i>&lt;timeOfDay1&gt;</i> , <i>&lt;timeOfDay2&gt;</i> ) <b>the number of seconds from</b> <i>&lt;datetime1&gt;</i> <b>to</b> <i>&lt;datetime2&gt;</i>	Returns the number of seconds between datetime1 and datetime2.
<b>SecondDifferenceInclusive</b> ( <i>&lt;datetime1&gt;</i> , <i>&lt;datetime2&gt;</i> ) <b>SecondDifferenceInclusive</b> ( <i>&lt;timeOfDay1&gt;</i> , <i>&lt;timeOfDay2&gt;</i> ) <b>the number of seconds (inclusive)</b> from <i>&lt;datetime1&gt;</i> <b>to</b> <i>&lt;datetime2&gt;</i>	Returns the inclusive number of seconds between datetime1 and datetime2.
<b>SecondDifferenceExclusive</b> ( <i>&lt;datetime1&gt;</i> , <i>&lt;datetime2&gt;</i> ) <b>SecondDifferenceExclusive</b> ( <i>&lt;timeOfDay1&gt;</i> , <i>&lt;timeOfDay2&gt;</i> ) <b>the number of seconds (exclusive)</b> from <i>&lt;datetime1&gt;</i> <b>to</b> <i>&lt;datetime2&gt;</i>	Returns the exclusive number of seconds between datetime1 and datetime2.
<b>MinuteDifference</b> ( <i>&lt;datetime1&gt;</i> , <i>&lt;datetime2&gt;</i> )	Returns the number of minutes between datetime1 and datetime2.

Syntax	Description
<b>MinuteDifference</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of minutes from</b> <datetime1> <b>to</b> <datetime2>	
<b>MinuteDifferenceInclusive</b> (<datetime1>, <datetime2>) <b>MinuteDifferenceInclusive</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of minutes (inclusive) from</b> <datetime1> <b>to</b> <datetime2>	Returns the inclusive number of minutes between datetime1 and datetime2.
<b>MinuteDifferenceExclusive</b> (<datetime1>, <datetime2>) <b>MinuteDifferenceExclusive</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of minutes (exclusive) from</b> <datetime1> <b>to</b> <datetime2>	Returns the exclusive number of minutes between datetime1 and datetime2.
<b>HourDifference</b> (<datetime1>, <datetime2>) <b>HourDifference</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of hours from</b> <datetime1> <b>to</b> <datetime2>	Returns the number of hours between datetime1 and datetime2.
<b>HourDifferenceInclusive</b> (<datetime1>, <datetime2>) <b>HourDifferenceInclusive</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of hours (inclusive) from</b> <datetime1> <b>to</b> <datetime2>	Returns the inclusive number of hours between datetime1 and datetime2.
<b>HourDifferenceExclusive</b> (<datetime1>, <datetime2>) <b>HourDifferenceExclusive</b> (<timeOfDay1>, <timeOfDay2>) <b>the number of hours (exclusive) from</b> <datetime1> <b>to</b> <datetime2>	Returns the exclusive number of hours between datetime1 and datetime2.
<b>ExtractDate</b> (<datetime>)	Extracts the date from a datetime attribute.
<b>ExtractTimeOfDay</b> (<datetime>)	Extracts the time of day from a datetime attribute. Can be used to set the value of a

Syntax	Description
	timeofday attribute to the time the rule is executed by extracting the time from the current date and time.
<b>AddHours</b> ( <i>&lt;datetime&gt;</i> , <i>&lt;num_hours&gt;</i> ) <b>AddHours</b> ( <i>&lt;timeOfDay&gt;</i> , <i>&lt;num_hours&gt;</i> ) <b>the time</b> <i>&lt;num_hours&gt;</i> <b>hours</b> <b>after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_hours&gt;</i> <b>hours</b> <b>before</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_hours&gt;</i> <b>hour</b> <b>after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_hours&gt;</i> <b>hour</b> <b>before</b> <i>&lt;datetime&gt;</i>	Adds a number of hours to a date time.
<b>AddMinutes</b> ( <i>&lt;datetime&gt;</i> , <i>&lt;num_minutes&gt;</i> ) <b>AddMinutes</b> ( <i>&lt;timeOfDay&gt;</i> , <i>&lt;num_minutes&gt;</i> ) <b>the time</b> <i>&lt;num_minutes&gt;</i> <b>minutes after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_minutes&gt;</i> <b>minutes before</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_minutes&gt;</i> <b>minute after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_minutes&gt;</i> <b>minute before</b> <i>&lt;datetime&gt;</i>	Adds a number of minutes to a date time.
<b>AddSeconds</b> ( <i>&lt;datetime&gt;</i> , <i>&lt;num_seconds&gt;</i> ) <b>AddSeconds</b> ( <i>&lt;timeOfDay&gt;</i> , <i>&lt;num_seconds&gt;</i> ) <b>the time</b> <i>&lt;num_seconds&gt;</i> <b>seconds after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_seconds&gt;</i> <b>seconds before</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_seconds&gt;</i> <b>second after</b> <i>&lt;datetime&gt;</i> <b>the time</b> <i>&lt;num_seconds&gt;</i> <b>second before</b> <i>&lt;datetime&gt;</i>	Adds a number of seconds to a date time.

## Text functions

Syntax	Description
<code>&lt;text1&gt; &amp; &lt;text2&gt;</code>	Combines text1 with text2 and so on to form a single text value. <b>Note:</b> that you can use variables of any type. Values are formatted using the formatter that is installed in the rule session.
<b>the concatenation of</b> <code>&lt;text1&gt; &amp; &lt;text2&gt;</code>	Combines text1 with text2 and so on to form a single text value. <b>Note:</b> that you can use variables of any type. Values are formatted using the formatter that is installed in the rule session.
<b>Contains</b> ( <code>&lt;text&gt;</code> , <code>&lt;sub-string&gt;</code> ) <code>&lt;text&gt; contains &lt;sub-string&gt;</code>	Returns a boolean value indicating whether the given text value contains the given text sub-string.
<b>EndsWith</b> ( <code>&lt;text&gt;</code> , <code>&lt;sub-string&gt;</code> ) <code>&lt;text&gt; ends with &lt;sub-string&gt;</code>	Returns a boolean value indicating whether the given text value ends with the given text sub-string.
<b>IsNumber</b> ( <code>&lt;text&gt;</code> ) <code>&lt;text&gt; is a number</code>	Returns a boolean value indicating whether the given text value represents a valid number.
<b>Length</b> ( <code>&lt;text&gt;</code> ) <b>the length of</b> <code>&lt;text&gt;</code>	Returns the character length of the given text value.
<b>StartsWith</b> ( <code>&lt;text&gt;</code> , <code>&lt;substring&gt;</code> ) <code>&lt;text&gt; starts with &lt;substring&gt;</code>	Returns a boolean value indicating whether the given text value starts with the given text sub-string.
<b>Substring</b> ( <code>&lt;text&gt;</code> , <code>&lt;offset&gt;</code> , <code>&lt;length&gt;</code> )	Returns the substring of text that starts at the given offset, that is the specified length in characters. Fewer characters are returned if the end of the string is reached.
<b>Text</b> ( <code>&lt;number&gt;</code> ) <b>Text</b> ( <code>&lt;date&gt;</code> ) <b>Text</b> ( <code>&lt;datetime&gt;</code> ) <b>Text</b> ( <code>&lt;timeOfDay&gt;</code> )	Convert the specified number or date attribute into a text value.

## Entity and relationship functions

Syntax	Description
<b>For</b> ( <code>&lt;relationship&gt;</code> , <code>&lt;Exp&gt;</code> ) <b>in the case of</b> <code>&lt;relationship&gt;</code> , <code>&lt;attr&gt;</code> <code>&lt;val&gt;</code> , <b>in the case of</b> <code>&lt;relationship&gt;</code>	Used to refer from one entity to another entity in a "One To One", "Many To One" or "Many To Many" relationship where there is only one condition.
<b>ForScope</b> ( <code>&lt;relationship&gt;</code> , <code>&lt;alias&gt;</code> ) <b>ForScope</b> ( <code>&lt;relationship&gt;</code> )	Used to refer from one entity to another entity in a "One To One", "Many To One" or "Many To Many" relationship where there are one or more conditions.

Syntax	Description
<b>in the case of</b> <relationship> <b>in the case of</b> <relationship> (<alias>)	
<b>ForAll</b> (<relationship>, <Exp>) <b>each of</b> <relationship-attr> <b>for each of</b> <relationship>, <attr> <b>for all of</b> <relationship>, <attr>	<p>Used to refer from one entity to another entity in a "One To Many" or "Many To Many" relationship, when you need to determine whether all members of the target entity group need to satisfy the rule.</p> <p>This form is used when there is only one condition in the rule.</p>
<b>ForAllScope</b> (<relationship>) <b>ForAllScope</b> (<relationship>, <alias>) <b>for all of</b> <relationship> <b>each of</b> <relationship> <b>for each of</b> <relationship> <b>for all of</b> <relationship> (<alias>) <b>each of</b> <relationship> (<alias>) <b>for each of</b> <relationship> (<alias>)	<p>Used to refer from one entity to another entity in a "One To Many" or "Many To Many" relationship, when you need to determine whether all members of the target entity group need to satisfy the rule.</p> <p>This form is used when there are one or more conditions in the rule.</p>
<b>Exists</b> (<relationship>, <Exp>) <b>at least one of</b> <relationship-attr> <b>for at least one of</b> <relationship>, <attr>	<p>Used to refer from one entity to another entity in a "One To Many" or "Many To Many" relationship, when you need to determine whether any members of the target entity group need to satisfy the rule.</p> <p>This form is used when there is only one condition in the rule.</p>
<b>ExistsScope</b> (<relationship>) <b>ExistsScope</b> (<relationship>, <alias>) <b>at least one of</b> <relationship> <b>for at least one of</b> <relationship> <b>at least one of</b> <relationship> (<alias>) <b>for at least one of</b> <relationship> (<alias>)	<p>Used to refer from one entity to another entity in a "One To Many" or "Many To Many" relationship, when you need to determine whether any members of the target entity group need to satisfy the rule.</p> <p>This form is used when there are one or more conditions in the rule.</p>
<b>IsMemberOf</b> (<target>, <relationship>) <b>IsMemberOf</b> (<target>, <alias>, <relationship>) <ent-target> <b>is a member of</b> <relationship> <ent-target> (<alias>) <b>is a member of</b> <relationship>	<p>Used as a conclusion to infer that an entity instance is a member of a relationship.</p> <p>Used as a condition to test that an entity instance is a target of a relationship for which a second entity instance is the source.</p>
<b>IsNotMemberOf</b> (<target>, <relationship>) <ent-target> <b>is not a member of</b> <relationship>	<p>Used as a condition to test that an entity instance is not a target of a relationship for which a second entity instance is the source.</p>
<b>InstanceCount</b> (<relationship>) <b>the number of</b> <relationship>	<p>Counts the number of instances that exist for an entity.</p>
<b>InstanceCountIf</b> (<relationship>,	<p>Counts the number of instances there are of an entity for which a particular entity-</p>

Syntax	Description
<p>&lt;Exp&gt;  <b>the number of</b> &lt;relationship&gt; <b>for which it is the case that</b> &lt;condition&gt;</p>	<p>level attribute has a particular value.</p>
<p><b>InstanceMaximum</b>( &lt;relationship&gt;, &lt;number-attr&gt; )  <b>InstanceMaximum</b>( &lt;relationship&gt;, &lt;date-attr&gt; )  <b>InstanceMaximum</b>( &lt;relationship&gt;, &lt;datetime-attr&gt; )  <b>InstanceMaximum</b>( &lt;relationship&gt;, &lt;time-attr&gt; )  &lt;date-attr&gt; <b>which is the latest for all</b> [of] &lt;relationship&gt;  &lt;max-attr&gt; <b>which is the greatest for all</b> [of] &lt;relationship&gt;  <b>the latest of all</b> &lt;relationship-attr&gt;  <b>the latest of all</b> &lt;attr&gt; <b>for</b> &lt;relationship&gt;  <b>the greatest of</b> [all] &lt;relationship-attr&gt;  <b>the greatest of</b> [all] &lt;attr&gt; <b>for</b> [all][of] &lt;relationship&gt;</p>	<p>Obtains the highest/most recent value of an entity-level variable for all instances of the entity.</p>
<p><b>InstanceMaximumIf</b>( &lt;relationship&gt;, &lt;number-attr&gt;, &lt;condition&gt; )  <b>InstanceMaximumIf</b>( &lt;relationship&gt;, &lt;date-attr&gt;, &lt;condition&gt; )  <b>InstanceMaximumIf</b>( &lt;relationship&gt;, &lt;datetime-attr&gt;, &lt;condition&gt; )  <b>InstanceMaximumIf</b>( &lt;relationship&gt;, &lt;time-attr&gt;, &lt;condition&gt; )  &lt;date-attr&gt; <b>which is the latest for all</b> [of] &lt;relationship&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  &lt;max-attr&gt; <b>which is the greatest for all</b> [of] &lt;relationship&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  <b>the latest of all</b> &lt;relationship-attr&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  <b>the greatest of all</b> &lt;relationship-attr&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  <b>the greatest of</b> &lt;attr&gt; <b>for all</b> [of] &lt;relationship&gt; <b>for which it is the case that</b> &lt;ent-test&gt;</p>	<p>Obtains the highest/most recent value of an entity-level variable for all instances of the entity for which a particular entity-level attribute has a particular value.</p>

Syntax	Description
<p><b>InstanceMinimum</b>(<i>&lt;relationship&gt;</i>, <i>&lt;number-attr&gt;</i>)</p> <p><b>InstanceMinimum</b>(<i>&lt;relationship&gt;</i>, <i>&lt;date-attr&gt;</i>)</p> <p><b>InstanceMinimum</b>(<i>&lt;relationship&gt;</i>, <i>&lt;datetime-attr&gt;</i>)</p> <p><b>InstanceMinimum</b>(<i>&lt;relationship&gt;</i>, <i>&lt;time-attr&gt;</i>)</p> <p><i>&lt;date-attr&gt;</i> <b>which is the earliest for all [of]</b> <i>&lt;relationship&gt;</i></p> <p><i>&lt;attr&gt;</i> <b>which is the least for all [of]</b> <i>&lt;relationship&gt;</i></p> <p><b>the earliest of all</b> <i>&lt;relationship-attr&gt;</i></p> <p><b>the earliest of all</b> <i>&lt;attr&gt;</i> <b>for</b> <i>&lt;relationship&gt;</i></p> <p><b>the least of [all]</b> <i>&lt;relationship-attr&gt;</i></p> <p><b>the least of [all]</b> <i>&lt;attr&gt;</i> <b>for [all][of]</b> <i>&lt;relationship&gt;</i></p>	<p>Obtains the lowest/least recent value of an entity-level variable for all instances of the entity.</p>
<p><b>InstanceMinimumIf</b>(<i>&lt;relationship&gt;</i>, <i>&lt;number-attr&gt;</i>, <i>&lt;condition&gt;</i>)</p> <p><b>InstanceMinimumIf</b>(<i>&lt;relationship&gt;</i>, <i>&lt;date-attr&gt;</i>, <i>&lt;condition&gt;</i>)</p> <p><b>InstanceMinimumIf</b>(<i>&lt;relationship&gt;</i>, <i>&lt;datetime-attr&gt;</i>, <i>&lt;condition&gt;</i>)</p> <p><b>InstanceMinimumIf</b>(<i>&lt;relationship&gt;</i>, <i>&lt;time-attr&gt;</i>, <i>&lt;condition&gt;</i>)</p> <p><i>&lt;date-attr&gt;</i> <b>which is the earliest for all [of]</b> <i>&lt;relationship&gt;</i> <b>for which it is the case that</b> <i>&lt;ent-test&gt;</i></p> <p><i>&lt;num-attr&gt;</i> <b>which is the least for all [of]</b> <i>&lt;relationship&gt;</i> <b>for which it is the case that</b> <i>&lt;ent-test&gt;</i></p> <p><b>the least of all</b> <i>&lt;relationship-attr&gt;</i> <b>for which it is the case that</b> <i>&lt;ent-test&gt;</i></p> <p><b>the least of all</b> <i>&lt;attr&gt;</i> <b>for</b> <i>&lt;relationship&gt;</i> <b>for which it is the case that</b> <i>&lt;ent-test&gt;</i></p> <p><b>the earliest of all</b> <i>&lt;attr&gt;</i> <b>for</b> <i>&lt;relationship&gt;</i> <b>for which it is the case that</b> <i>&lt;ent-test&gt;</i></p>	<p>Obtains the lowest/least recent value of an entity-level variable for all instances of the entity for which a particular entity-level attribute has a particular value.</p>
<p><b>InstanceSum</b>(<i>&lt;relationship&gt;</i>, <i>&lt;number-attr&gt;</i>)</p> <p><i>&lt;num-attr&gt;</i> <b>(totalled   totalled) for all</b></p>	<p>Obtains the sum of all instances of an entity-level variable.</p>

Syntax	Description
<p>[<b>of</b>] &lt;relationship&gt;  <b>the total amount of [all]</b> &lt;relationship-attr&gt;  <b>the total for all</b> &lt;relationship-attr&gt;  <b>total for all</b> &lt;relationship&gt;, &lt;attr&gt;</p>	
<p><b>InstanceSumIf</b>( &lt;relationship&gt;, &lt;number-attr&gt;, &lt;condition&gt; )  &lt;num-attr&gt; <b>totalled for all [of]</b> &lt;relationship&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  &lt;num-attr&gt; <b>totalled for all [of]</b> &lt;relationship&gt; <b>for which it is the case that</b> &lt;ent-test&gt;  <b>the total amount of all</b> &lt;relationship-attr&gt; <b>only where</b> &lt;condition&gt;  <b>the total amount of [all]</b> &lt;relationship-attr&gt; <b>for which it is the case that</b> &lt;condition&gt;  <b>total for all</b> &lt;relationship&gt;, &lt;attr&gt; <b>only where</b> &lt;condition&gt;</p>	<p>Obtains the sum of all instances of an entity-level variable for which it is true of the entity that a specific entity-level Boolean attribute is true.</p>
<p><b>InstanceValueIf</b>( &lt;relationship&gt;, &lt;number-attr&gt;, &lt;condition&gt; )  <b>InstanceValueIf</b>( &lt;relationship&gt;, &lt;text-attr&gt;, &lt;condition&gt; )  <b>InstanceValueIf</b>( &lt;relationship&gt;, &lt;date-attr&gt;, &lt;condition&gt; )  <b>InstanceValueIf</b>( &lt;relationship&gt;, &lt;datetime-attr&gt;, &lt;condition&gt; )  <b>InstanceValueIf</b>( &lt;relationship&gt;, &lt;time-attr&gt;, &lt;condition&gt; )</p>	<p>Obtains a value from a unique entity instance, identified from the target entity instances of a relationship by a condition.</p> <ul style="list-style-type: none"> <li>• If the condition identifies a single target entity instance, then the value is the value calculated against that entity instance.</li> <li>• If more than one target instance meets the condition, then Uncertain is returned.</li> <li>• If no target instances meet the condition and the relationship is known the value is Uncertain.</li> </ul>
<p><b>InstanceEquals</b>( &lt;instance1&gt;, &lt;instance2&gt; )  &lt;ent-target&gt; <b>is</b> &lt;ent-target&gt;</p>	<p>Determines if two instances of an entity are the same instance.</p>
<p><b>InstanceNotEquals</b>( &lt;instance1&gt;, &lt;instance2&gt; )  &lt;ent-target&gt; <b>is not</b> &lt;ent-target&gt;</p>	<p>Determines if two instances of an entity are not the same instance.</p>
<p><b>InferInstance</b>( &lt;relationship&gt;, &lt;identity&gt; )  &lt;rel&gt;( &lt;identity&gt; ) <b>exists</b></p>	<p>Used as a conclusion to infer that an entity instance exists and is a member of a relationship.</p>

## Temporal reasoning functions

Syntax	Description
<b>IntervalCountDistinct</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;variable&gt;</i> ) <b>IntervalCountDistinct</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;condition&gt;</i> )	Counts the number of known distinct values for the variable, in the interval from the start date (inclusive) to the end date (exclusive).
<b>IntervalCountDistinctIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;variable&gt;</i> , <i>&lt;condition&gt;</i> )	Counts the number of known distinct values for the variable, in the interval from the start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true.
<b>IntervalDailySum</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;number-attr&gt;</i> )	Calculates the sum of a currency or number variable, in the interval from the start date (inclusive) to end date (exclusive). The attribute is assumed to be a daily quantity.
<b>IntervalDailySumIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;number-attr&gt;</i> , <i>&lt;condition&gt;</i> )	Calculates the sum of all the daily values for a currency or number variable, in the interval from a start date (inclusive) to an end date (exclusive), only including times when a condition is true.
<b>IntervalMaximum</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;number-attr&gt;</i> ) <b>IntervalMaximum</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;date-attr&gt;</i> ) <b>IntervalMaximum</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;datetime-attr&gt;</i> ) <b>IntervalMaximum</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;time-attr&gt;</i> )	Selects the maximum value of a variable in the interval from a start date (inclusive) to an end date (exclusive).
<b>IntervalMaximumIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;number-attr&gt;</i> , <i>&lt;condition&gt;</i> ) <b>IntervalMaximumIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;date-attr&gt;</i> , <i>&lt;condition&gt;</i> ) <b>IntervalMaximumIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;datetime-attr&gt;</i> , <i>&lt;condition&gt;</i> ) <b>IntervalMaximumIf</b> ( <i>&lt;start-date&gt;</i> , <i>&lt;end-date&gt;</i> , <i>&lt;time-attr&gt;</i> , <i>&lt;condition&gt;</i> )	Selects the maximum value of a variable in the interval from a start date (inclusive) to an end date (exclusive), only including times when a condition is true.
<b>IntervalMinimum</b> ( <i>&lt;start-</i>	Selects the minimum value of a variable in the interval from a start date (inclusive) to an end

Syntax	Description
<p><i>date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>number-attr</i>&gt;)</p> <p><b>IntervalMinimum</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>date-attr</i>&gt;)</p> <p><b>IntervalMinimum</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>datetime-attr</i>&gt;)</p> <p><b>IntervalMinimum</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>time-attr</i>&gt;)</p>	<p>date (exclusive).</p>
<p><b>IntervalMinimumIf</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>number-attr</i>&gt;, &lt;<i>condition</i>&gt;)</p> <p><b>IntervalMinimumIf</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>date-attr</i>&gt;, &lt;<i>condition</i>&gt;)</p> <p><b>IntervalMinimumIf</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>datetime-attr</i>&gt;, &lt;<i>condition</i>&gt;)</p> <p><b>IntervalMinimumIf</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>time-attr</i>&gt;, &lt;<i>condition</i>&gt;)</p>	<p>Selects the minimum value of a variable in the interval from a start date (inclusive) to an end date (exclusive), only including times when a condition is true.</p>
<p><b>IntervalWeightedAverage</b> (&lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>number-attribute</i>&gt;)</p>	<p>Calculates the average value of a currency or number variable in the interval from a start date (inclusive) to an end date (exclusive) weighted by the time span to which each value applies.</p>
<p><b>IntervalWeightedAverageIf</b> (&lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>number-attribute</i>&gt;, &lt;<i>condition</i>&gt;)</p>	<p>Calculates the average value of a currency or number variable in the interval from a start date (inclusive) to an end date (exclusive), only including times when a boolean condition is true (weighted by the time span to which each value applies and where the filter is true).</p>
<p><b>IntervalAlways</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>condition</i>&gt;)</p>	<p>Returns true if and only if a boolean condition is true at all times in the interval from the start date (inclusive) to the end date (exclusive).</p>
<p><b>IntervalAtLeastDays</b>( &lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>NumDays</i>&gt;, &lt;<i>condition</i>&gt;)</p>	<p>Returns true if and only if a boolean condition is true for at least the specified number of days (not necessarily consecutive) in the interval from the start date (inclusive) to the end date (exclusive).</p>
<p><b>IntervalConsecutiveDays</b> (&lt;<i>start-date</i>&gt;, &lt;<i>end-date</i>&gt;, &lt;<i>NumDays</i>&gt;, &lt;<i>condition</i>&gt;)</p>	<p>Returns true if and only if a boolean condition is true for at least a given number of consecutive days in the interval from the start date (inclusive) to the end date (exclusive).</p>
<p><b>IntervalSometimes</b>( &lt;<i>start-</i></p>	<p>Returns true if and only if a boolean condition is ever true in the interval from the start date</p>

Syntax	Description
<i>date</i> >, < <i>end-date</i> >, < <i>condition</i> >)	(inclusive) to the end date (exclusive).
<b>ValueAt</b> (< <i>date</i> >, < <i>value</i> >)	Returns the value of the given attribute at the specified date.
<b>WhenLast</b> (< <i>date</i> >, < <i>condition</i> >)	Returns the date on which a boolean condition was last true, looking backwards from (and including) a specified date.
<b>WhenNext</b> (< <i>date</i> >, < <i>condition</i> >)	Returns the date on which a boolean condition will next be true, looking forwards from (and including) a specified date.
<b>Latest</b> ()	Returns a date value equivalent to the latest possible date - namely a date guaranteed to be later than any other date that a date attribute may take or an expression may evaluate to.
<b>Earliest</b> ()	Returns a date value equivalent to the earliest possible date - namely a date guaranteed to be earlier than any other date that a date attribute may take or an expression may evaluate to.
<b>TemporalDaysSince</b> (< <i>date</i> >, < <i>end-date</i> >)	Returns a number variable that varies every day and is the number of full days since the date.
<b>TemporalWeeksSince</b> (< <i>date</i> >, < <i>end-date</i> >)	Returns a number variable that varies every week and is the number of full weeks since the date.
<b>TemporalMonthsSince</b> (< <i>date</i> >, < <i>end-date</i> >)	Returns a number variable that varies every month and is the number of full months since the date. Note: Where the supplied date is after the 28th day of the month, and a subsequent month has fewer days than the supplied month, the change point for the anniversary month will be created on the last day of that month. For example, if the supplied date is 28, 29, 30 or 31 January 2007, the first change point will be 28 February 2007.
<b>TemporalYearsSince</b> (< <i>date</i> >, < <i>end-date</i> >)	Returns a number variable that varies every year and is the number of full years since the date.
<b>TemporalAlwaysDays</b> (< <i>days</i> >, < <i>condition</i> >)	Returns a boolean attribute that varies over time and is true if and only if a boolean condition is true for all of a given number of preceding days, not including the current day.
<b>TemporalConsecutiveDays</b> (< <i>minDays</i> >, < <i>days</i> >, < <i>condition</i> >)	Returns a boolean attribute that varies over time and is true if and only if a boolean condition is true for at least a minimum number of consecutive days at any time within the preceding set number of days, not including the current day.
<b>TemporalSometimesDays</b> (< <i>days</i> >, < <i>condition</i> >)	Returns a boolean attribute that varies over time and is true if and only if a boolean condition is ever true within a specified number of preceding days, not including the current day.
<b>TemporalAfter</b> (< <i>date</i> >)	Returns a boolean attribute that varies over time and is true after a date and false on and before.
<b>TemporalBefore</b> (< <i>date</i> >)	Returns a boolean attribute that varies over time and is true before a date and false on and afterwards.
<b>TemporalOn</b> (< <i>date</i> >)	Returns a boolean attribute that varies over time and is true on a date and false before and afterwards.

Syntax	Description
<b>TemporalOnOrAfter</b> ( <date> )	Returns a boolean attribute that varies over time and is true on or after a date and false before.
<b>TemporalOnOrBefore</b> ( <date> )	Returns a boolean attribute that varies over time and is true on and before a date and false afterwards.
<b>TemporalFromStartDate</b> ( <relationship>, <date>, <value> )	Returns a single temporal attribute (at the source entity level) from a relationship and a value attribute on the entities, with values that take effect from a start date attribute.
<b>TemporalFromEndDate</b> ( <relationship>, <date>, <value> )	Returns a single temporal attribute (at the source entity level) from a relationship and a value attribute on the entities, with values that take effect up until an end date attribute.
<b>TemporalFromRange</b> ( <relationship>, <start-date>, <end-date>, <Value> )	Returns a single temporal attribute (at the source entity level) from a relationship and a value attribute on the entities, with values that takes effect from a start date attribute (inclusive) until and end date attribute (exclusive). The value is uncertain if it expires before the next start date.
<b>TemporalIsWeekday</b> ( <startdate>, <enddate> )	Returns true on dates that are weekdays and false on dates that are weekends from the specified start date (inclusive) to the end date (exclusive). Returns uncertain outside of the date range.
<b>TemporalOncePerMonth</b> ( <startdate>, <enddate>, <dayofmonth> )	Returns true if the day is equal to the day-of-month parameter and false on all other days of the month from the specified start date (inclusive) to the end date (exclusive). Returns uncertain outside of the date range. When the day-of-month exceeds the number of days in the current month, the value is true on the last day of that month, so that the function returns a value that is true exactly one day per month.

## Validation event functions

Syntax	Description
<b>Error</b> ( <text> )	An error event is used to pass a message to the user, and prevent them from continuing an investigation until the condition which triggered that error no longer applies.
<b>Warning</b> ( <text> )	A warning event is used to pass a message to the user, but permits them to continue despite the condition which triggered that warning.

## Deprecated functions

Syntax	Description
<b>CallCustomFunction</b> ( <A>, <B> )	Returns the result of an external call to a code library. The code library must be provided to the determinations engine for the custom function call to succeed.

## Localized function references (all languages)

The Oracle Policy Modeling Function Reference has been localized for the languages listed below. Click on the appropriate link to proceed to a copy of the Function Reference for that language:

Language	Locale Code	Parser Type
Arabic (Saudi Arabia)	ar-SA	Syntactic
Brazilian	pt-BR	Syntactic
Chinese (Simplified)	zh-CN	Syntactic
Chinese (Traditional)	zh-HK	Syntactic
Czech	cs-CZ	Non-syntactic
Danish	da-DK	Syntactic
Dutch	nl-NL	Syntactic
English (Great Britain)	en-GB	Syntactic
English (United States)	en-US	Syntactic
Finnish	fi-FI	Syntactic
French (France)	fr-FR	Syntactic
German (Germany)	de-DE	Syntactic
Hebrew	he-IL	Syntactic
Italian	it-IT	Syntactic
Japanese	ja-JP	Syntactic
Korean	ko-KR	Syntactic
Norwegian (Bokmål)	nb-NO	Non-syntactic
Polish	pl-PL	Non-syntactic
Portuguese (Portugal)	pt-PT	Syntactic
Russian	ru-RU	Syntactic
Spanish (Modern)	es-ES	Syntactic
Swedish	sv-SE	Syntactic
Thai	th-TH	Non-syntactic
Turkish	tr-TR	Syntactic

## Structural configuration settings

The following settings are used for modeling the structure of legislation. These elements must be formatted using the **Con-figuration** style on the Oracle Policy Modeling toolbar.

Be sure to use the exact syntax for these functions including spacing and brackets as specified below.

Element	Syntax	Description
Default_structural_element	Default_structural_element[<replacement structural text>]	Used to bypass the default text ("section ") generated for structural elements.

Element	Syntax	Description
		<p>You may specify multiple Default_structural_element entries in a single rule document to apply to all rules following each entry.</p> <p>Note that this is space-sensitive. If you want to have a space between the element and the element number, you must include a space in the Configuration entry.</p>
Default_structural_globalproof	Default_structural_globalproof[<replacement structural text including structural element ^x>]	<p>Used to bypass the default text ("^x is satisfied") generated for structural elements.</p> <p>You may specify multiple Default_structural_globalproof entries in a single rule document to apply to all rules following each entry.</p> <p>Note that this works in conjunction with Default_structural_element, which is used to define the ^x form.</p>
Default_structural_entityproof	Default_structural_entityproof[<replacement structural text including structural elements ^x and ^entity>]	<p>Used to bypass the default text ("^entity satisfies ^x") generated for entity-level structural elements.</p> <p>You may specify multiple Default_structural_entityproof entries in a single rule document to apply to all rules following each entry.</p> <p>Note that this works in conjunction with Default_structural_element and the entity defined in the properties file, which are used to define both ^x and</p>

Element	Syntax	Description
		^entity forms.
Ignore	Ignore[<text to be ignored>]	Defines a string to be ignored by Oracle Policy Modeling when generating boolean attributes from a rule document, allowing more meaningful generation of structural elements.
Replace	Replace[<text to be replaced>, <replacement text including structural element ^x>]	Replaces generic text with predefined text for automatic structural element generation. This is used in conjunction with the substitution token "^x".
Replace_entity	Replace_entity[<text to be replaced>, <replacement structural text including structural elements ^x and ^entity>]	Replaces generic entity-level text with predefined text for automatic structural element generation. This is used in conjunction with the substitution tokens "^x" and "^entity".

See also:

- [Use keywords to customize automatic structural attributes](#)

## Rule function examples

Topics in "Rule function examples"

- [Comparison operator rule examples](#)
- [Date function rule examples](#)
- [Time of day function rule examples](#)
- [Date and time function rule examples](#)
- [Numerical function rule examples](#)
- [Text function rule examples](#)
- [Entity and relationship function rule examples](#)
- [Temporal reasoning function rule examples](#)
- [Certain and known operator rule examples](#)

## Comparison operator rule examples

When using variables in rules you must state the value, or range of acceptable values, that are sufficient to satisfy the rule. To do this, you must use one of the logical operators.

Operator	Example
Less than	<b>the pre-2007 rules apply if</b> the date of claim < 2007-01-01
Greater than	<b>the new rates apply if</b> the date of investigation > 2007-06-30
Less than or equal to	<b>the person can apply for Immunization Allowance for the child if</b> the date of claim <= the child's second birthday
Greater than or equal to	<b>the individual qualifies for age pension if</b> the individual's age >= 65
Equals	<b>the person is 18 if</b> the person's age = 18
Not equal to	<b>the person's salary has been adjusted if</b> the new salary <> the old salary

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Use variables in rules](#)

## Date function rule examples

Date functions are used to perform a number of common calculations which frequently appear in rules.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below. Date and number inputs may be either constant values or variables.

Note that in date calculations attempting to exceed the allowable date range with a date too far in the past or future will lead to the [Earliest](#) or [Latest](#) value as appropriate.

## Rule examples

Function	Example rule	Inputs	Outputs	Further information
CurrentDate	<p><b>today = the current date</b></p> <p><b>the date of the investigation = CurrentDate()</b></p>	<p>today's date: 2005-04-15</p> <p>today's date: 2009-08-31</p>	<p>today = 2005-04-15</p> <p>the date of the invest- igation = 2009-08-31</p>	<p>Get today's date</p>
Date	<p><b>the date of effect =Date ("2012-01-01")</b></p>	<p>2012-01-01</p>	<p>the date of effect = 2012-01-01</p>	<p>Convert a text string into a number or date</p>
MakeDate	<p><b>the calculation date =MakeDate(2006, 10, 19)</b></p>	<p>yyyy: 2006; mm: 10; dd: 19</p>	<p>the cal- culation date = 2006-10- 19</p>	<p>Get a date formed from a specified year, month and day</p>
ExtractDay	<p><b>the day of expiry = ExtractDay (the use-by date on the packet)</b></p>	<p>the use-by date on the packet: 2008-06-12</p>	<p>the day of expiry = 12</p>	<p>Get the day component of an input date</p>
ExtractMonth	<p><b>the month of expiry = ExtractMonth(the use-by date on the packet)</b></p>	<p>the use-by date on the packet: 2007-04-16</p>	<p>the month of expiry = 04</p>	<p>Get the month component of an input date</p>
ExtractYear	<p><b>the year of expiry = ExtractYear(the use-by date on the packet)</b></p>	<p>the use-by date on the packet: 2009-02-21</p>	<p>the year of expiry = 2009</p>	<p>Get the year component of an input date</p>
NextDayOfTheWeek	<p><b>next Monday = Nex- tDayOfTheWeek(the current date,"Monday")</b></p> <p><b>next Tuesday = the next Tues- day on or after the current date</b></p> <p><b>last Thursday = the Thursday on or before the current date</b></p>	<p>the current date: 2009-08-09</p>	<p>next Monday = 10 August 2009</p> <p>next Tues- day = 11 August 2009</p> <p>last Thursday = 6 August 2009</p>	<p>Get the date of the next or previous spe- cified day</p>
AddDays	<p><b>the date that the library book</b></p>	<p>the date of loan:</p>	<p>the date that</p>	<p>Add or sub-</p>

Function	Example rule	Inputs	Outputs	Further information
	<p><b>must be returned by = AddDays(the date of loan,21)</b></p> <p><b>the date that the library book must be returned by = the date 21 days after the date of loan</b></p> <p><b>the closing date for the entry = AddDays(the date of the show,-10)</b></p>	<p>2006-01-04</p> <p>the date of the show: 2007-05-15</p>	<p>the library book must be returned by = 25</p> <p>the closing date for the entry = 2007-05-05</p>	<p>tract a specified number of days to an input date</p>
AddWeeks	<p><b>the date that the event finishes = AddWeeks(the date that the event begins,the number of weeks in the event)</b></p> <p><b>the date that the event finishes = the date the number of weeks in the event weeks after the date that the event begins</b></p> <p><b>the start date of the 5 week promotion = AddWeeks(the end date of the promotion,-5)</b></p> <p><b>the start date of the 5 week promotion = the date 5 weeks before the end date of the promotion</b></p>	<p>the date that the event begins: 2001-08-13</p> <p>the number of weeks in the event: 12</p> <p>the end date of the promotion: 2008-12-24</p>	<p>the date that the event finishes = 2001-11-05</p> <p>the start date of the 5 week promotion = 2008-11-19</p>	<p>Add or subtract a specified number of weeks to an input date</p>
AddMonths	<p><b>the date that the player can return from suspension = AddMonths(the date of the suspension,3)</b></p> <p><b>the date that the player can return from suspension = the date 3 months after the date of the suspension</b></p> <p><b>the start date of the player's 12 month contract = AddWeeks(the end date of the player's contract,-12)</b></p> <p><b>the start date of the player's 12 month contract = the date 12 months before the end date of the player's contract</b></p>	<p>the date of the suspension: 2005-12-12</p> <p>the end date of the player's contract: 2006-06-30</p>	<p>the date that the player can return from suspension = 2006-03-12</p> <p>the start date of the player's 12 month contract = 2005-06-30</p>	<p>Add or subtract a specified number of months to an input date</p>

Function	Example rule	Inputs	Outputs	Further information
AddYears	<p><b>the date of the trial = AddYears(the date of the crime,3)</b></p> <p><b>the date of the trial = the date 3 years after the date of the crime</b></p> <p><b>the date that the prison sentence starts = AddYears(the date that the prison sentence ends,-20)</b></p> <p><b>the date that the prison sentence starts = the date 20 years before the date that the prison sentence ends</b></p>	<p>the date of the crime: 2002-01-01</p> <p>the date that the prison sentence ends: 1980-03-16</p>	<p>the date of the trial = 2005-01-01</p> <p>the date that the prison sentence starts = 1960-03-16</p>	<p>Add or subtract a specified number of years to an input date</p>
YearStart	<p><b>the start of the first relevant year = YearStart(2009-09-09)</b></p> <p><b>the start of the second relevant year = the first day of the year in which the date of the grand occasion falls</b></p>	<p>the date of the grand occasion: 2007-09-09</p>	<p>the start of the first relevant year = 2009-01-01</p> <p>the start of the second relevant year = 2007-01-01</p>	<p>Find the first date in the year</p>
YearEnd	<p><b>the end of the relevant year = YearEnd(the relevant date)</b></p> <p><b>the end of the relevant year = the last day of the year in which the relevant date falls</b></p>	<p>the relevant date: 2005-10-15</p>	<p>the end of the relevant year = 2005-12-31</p>	<p>Find the last date in the year</p>
NextDate	<p><b>the end of the Australian tax year = NextDate(the test date, 30, 6)</b></p>	<p>the test date: 2005-07-02</p>	<p>the end of the Australian tax year = 2006-06-30</p>	<p>Find the next instance of the given day/month</p>
UKTaxYearDates	<p><b>the date of effect = the next UK tax year end date on or after the test date</b></p> <p><b>the assessment date = the previous UK tax year start date on or before the test date</b></p>	<p>the test date: 2003-09-21</p>	<p>the date of effect = 2004-04-05</p> <p>the assess-</p>	<p>Find the start or the end date for the previous or next UK tax year</p>

Function	Example rule	Inputs	Outputs	Further information
			ment date = 2003-04-06	
WeekdayCount	<p><b>the number of working days until my holiday = WeekdayCount(2007-12-03, 2007-12-13)</b></p> <p><b>the number of business days in the specified period = the number of weekdays (inclusive) between 2007-10-15 and 2007-10-31</b></p>	<p>date1: 2007-12-03; date2: 2007-12-13</p> <p>date1: 2007-10-15; date2: 2007-10-31</p>	<p>the number of working days until my holiday = 8</p> <p>the number of business days in the specified period = 12</p>	Count the number of weekdays between two dates
DayDifference	<b>the number of days in the assessment period = DayDifference(2006-10-01,2006-10-14)</b>	<p>date1: 2006-10-01</p> <p>date2: 2006-10-14</p>	the number of days in the assessment period = 13	Count the number of whole days between two dates
DayDifferenceInclusive	<b>the number of days in the assessment period = DayDifferenceInclusive(2006-10-01,2006-10-14)</b>	<p>date1: 2006-10-01</p> <p>date2: 2006-10-14</p>	the number of days in the assessment period = 14	
DayDifferenceExclusive	<b>the number of days in the assessment period = DayDifferenceExclusive(2006-10-01,2006-10-14)</b>	<p>date1: 2006-10-01</p> <p>date2: 2006-10-14</p>	the number of days in the assessment period = 12	
WeekDifference	<b>the number of weeks until Christmas = WeekDifference (the current date,2011-12-25)</b>	the current date: 2011-11-25	the number of weeks remaining = 4	Count the number of whole weeks between two dates
WeekDifferenceInclusive	<b>the number of weeks until Christmas = WeekDifferenceInclusive(the current date,2011-12-25)</b>	the current date: 2011-11-25	the number of weeks remaining = 5	
WeekDifferenceExclusive	<b>the number of weeks until Christmas = WeekDifferenceExclusive(the current date,2011-12-25)</b>	the current date: 2011-11-25	the number of weeks remaining = 3	

Function	Example rule	Inputs	Outputs	Further information
MonthDifference	the number of months remaining in the phone contract = <b>MonthDifference</b> (the current date, the expiry date of the contract)	the current date: 2011-11-28 the expiry date of the contract: 2013-03-24	the number of months remaining = 15	Count the number of whole months between two dates
MonthDifferenceInclusive	the number of months remaining in the phone contract = <b>MonthDifferenceInclusive</b> (the current date, the expiry date of the contract)	the current date: 2011-11-28 the expiry date of the contract: 2013-03-24	the number of months remaining = 16	
MonthDifferenceExclusive	the number of months remaining in the phone contract = <b>MonthDifferenceExclusive</b> (the current date, the expiry date of the contract)	the current date: 2011-11-28 the expiry date of the contract: 2013-03-24	the number of months remaining = 14	
YearDifference	the age of the tree in years = <b>YearDifference</b> (the date the tree was planted, the date the tree was assessed)  the age of the tree in years = the number of years between the date the tree was planted and the date the tree was assessed	the date the tree was planted: 2000-03-12 the date the tree was assessed: 2003-12-12	the age of the tree in years = 3	Count the number of whole years between two dates
YearDifferenceInclusive	the age of the tree in years = <b>YearDifferenceInclusive</b> (the date the tree was planted, the date the tree was assessed)  the age of the tree in years = the number of years (inclusive) between the date the tree was planted and the date the tree was assessed	the date the tree was planted: 2000-03-12 the date the tree was assessed: 2003-12-12	the age of the tree in years = 4	
YearDifferenceExclusive	the age of the tree in years = <b>YearDifferenceExclusive</b> (the date the tree was planted, the date the tree was assessed)	the date the tree was planted: 2000-03-12 the date the tree	the age of the tree in years = 2	

Function	Example rule	Inputs	Outputs	Further information
	<b>the age of the tree in years = the number of years (exclusive) between the date the tree was planted and the date the tree was assessed</b>	was assessed: 2003-12-12		

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Get the latest or earliest date or time](#)
- [Find the day from a date](#)

## Time of day function rule examples

Time of day functions are used with time of day variables to set the time of day and to extract the second/minute/hour from a time of day.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below.

### Rule examples

Function	Example rule	Inputs	Outputs	Further information
TimeOfDay	<b>the latest submission time = TimeOfDay("12:30:00")</b>	12:30:00	the latest submission time = 12:30:00	<a href="#">Get the time of day from a text string</a>
ExtractSecond	<b>the second component of the submission time = ExtractSecond(the submission time)</b>	the submission time: 14:42:32	the second component of the submission time = 32	<a href="#">Get the second component of an input time</a>
ExtractMinute	<b>the minute component of the submission time = ExtractSecond(the submission time)</b>	the submission time: 14:42:32	the minute component of the submission time = 42	<a href="#">Get the minute component of an input time</a>
ExtractHour	<b>the hour component of the submission time =</b>	the submission time: 14:42:32	the hour component	<a href="#">Get the hour component of</a>

Function	Example rule	Inputs	Outputs	Further information
	<b>ExtractHour(the submission time)</b>		of the submission time = 14	<a href="#">an input time</a>

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Get the latest or earliest date or time](#)

## Date and time function rule examples

Date and time functions are used with date and time variables to express the current date and time (at the start of the session), to set the date and time, to calculate the difference in units between two dates, to extract a unit from a date and time and to extract a time of day.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below.

### Rule examples

Function	Example rule	Inputs	Outputs	Further information
CurrentDateTime	<b>the date and time of the investigation = CurrentDateTime()</b>	the current date time: 2009-09-15 03:24:12	the date and time of the investigation = 2009-09-15 03:24:12	<a href="#">Get the current date and time</a>
DateTime	<b>the latest submission date and time= DateTime(the submission date and time specified on the application form)</b>	the submission date and time specified on the application form: 2012-12-31 18:00:00	the latest submission date and time= 2012-12-31 18:00:00	<a href="#">Get a date and time from a text string</a>
ConcatenateDateTime	<b>the latest submission time = ConcatenateDateTime(the submission date, the submission closing time)</b>	the submission date: 2010-01-15 the submission closing time: 17:00:00	the latest submission time = 2010-01-15 17:00:00	<a href="#">Get a date and time by joining together a separate date and time</a>

Function	Example rule	Inputs	Outputs	Further information
SecondDifference	<b>the number of seconds between first place and second place = SecondDifference(the first place time, the second place time)</b>	the first place time: 2008-06-30 09:31:05 the second place time: 2008-06-30 09:31:10	the number of seconds between first place and second place = 5	Count the number of seconds between two times
SecondDifferenceInclusive	<b>the number of seconds between first place and second place = SecondDifferenceInclusive(the first place time, the second place time)</b>	the first place time: 2008-06-30 09:31:05 the second place time: 2008-06-30 09:31:10	the number of seconds between first place and second place = 6	
SecondDifferenceExclusive	<b>the number of seconds between first place and second place = SecondDifferenceExclusive(the first place time, the second place time)</b>	the first place time: 2008-06-30 09:31:05 the second place time: 2008-06-30 09:31:10	the number of seconds between first place and second place = 4	
MinuteDifference	<b>the number of minutes late the plumber is = MinuteDifference(the time the plumber was meant to arrive, the time that the plumber actually arrived)</b>	the time the plumber was meant to arrive: 2009-10-18 08:30:00 the time that the plumber actually arrived: 2009-10-18 09:00:40	the number of minutes late the plumber is = 30	Count the number of whole minutes between two times
MinuteDifferenceInclusive	<b>the number of minutes late the plumber is = MinuteDifferenceInclusive(the time the plumber was meant to arrive, the time that the plumber actually arrived)</b>	the time the plumber was meant to arrive: 2009-10-18 08:30:00 the time that the plumber actually arrived: 2009-10-18 09:00:40	the number of minutes late the plumber is = 31	
MinuteDifferenceExclusive	<b>the number of minutes late the plumber is = MinuteDifferenceExclusive(the time</b>	the time the plumber was meant to arrive:	the number of minutes late the	

Function	Example rule	Inputs	Outputs	Further information
	<b>the plumber was meant to arrive, the time that the plumber actually arrived)</b>	2009-10-18 08:30:00 the time that the plumber actually arrived: 2009-10-18 09:00:40	plumber is = 29	
HourDifference	<b>the number of hours the plane was delayed by = HourDifference(the scheduled arrival time of the flight, the arrival time of the delayed flight)</b>	the scheduled arrival time of the flight: 2006-10-13 09:50:00 the arrival time of the delayed flight: 2006-10-13 11:00:00	the number of hours the plane was delayed by = 1	
HourDifferenceInclusive	<b>the number of hours the plane was delayed by = HourDifferenceInclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)</b>	the scheduled arrival time of the flight: 2006-10-13 09:50:00 the arrival time of the delayed flight: 2006-10-13 11:00:00	the number of hours the plane was delayed by = 2	Count the number of whole hours between two times
HourDifferenceExclusive	<b>the number of hours the plane was delayed by = HourDifferenceExclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)</b>	the scheduled arrival time of the flight: 2006-10-13 09:50:00 the arrival time of the delayed flight: 2006-10-13 11:00:00	the number of hours the plane was delayed by = 0	
ExtractDate	<b>the password expiry date = ExtractDate(the password expiry date time)</b>	the password expiry date time: 2009-09-11 00:00:00	the password expiry date = 2009-09-11	Get the date from a date and time
ExtractTimeOfDay	<b>the time of the assessment = ExtractTimeOfDay(the current date time)</b>	the current date time: 2009-09-04 10:46:12	the time of the assessment = 10:46:12	Get the time of day from a date and time

Function	Example rule	Inputs	Outputs	Further information
AddHours	<b>the date time that the offer expires = AddHours(the date time that the offer starts, 48)</b>	the date time that the offer starts: 2010-10-08 12:00:00	the date time that the offer expires = 2010-10-10 12:00:00	Get a date and time by adding or subtracting a specified number of hours to another date and time
AddMinutes	<b>the date time that the train is due = the time 25 minutes after the date time that the train departed</b>	the date time that the train departed: 2005-01-16 22:50:00	the date time that the train is due = 2005-01-16 23:15:00	Get a date and time by adding or subtracting a specified number of minutes to another date and time
AddSeconds	<b>the date time at the start of the recording = the time 40 seconds before the date time at the end of the recording</b>	the date time at the end of the recording: 2008-01-01 14:27:52	the date time at the start of the recording = 2008-01-01 14:27:12	Get a date and time by adding or subtracting a specified number of seconds to another date and time

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Get the latest or earliest date or time](#)

## Numerical function rule examples

Numerical functions are used with number and currency variables to perform basic and complex arithmetic calculations, trigonometric calculations and maximum/minimum calculations.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below.

## Rule examples

Function	Example rule	Inputs	Outputs
Number	<b>the number =Number(the number text)</b>	the number text: 15	the number = 15
Addition	<b>the total = the first amount + the second amount</b>	the first amount: 2; the second amount: 3	the total = 5
Subtraction	<b>the total = the first amount - the second amount</b>	the first amount: 100; the second amount: 5	the total = 95
Multiplication	<b>the total = the first amount * the second amount</b>	the first amount: 7; the second amount: 2	the total = 14
Division	<b>the total = the first amount / the second amount</b>	the first amount: 10; the second amount: 5	the total = 2
Integer Division	<b>the result = the value \ 5</b>	the value: 54.25	the result = 10
Remainder after Integer Division	<b>the total = the first amount modulo the second amount</b>	the first amount: 9; the second amount: 3	the result = 0
Maximum	<b>the highest number of fish caught = Maximum (the number of fish caught by Bob, the number of fish caught by Mary)</b>	the number of fish caught by Bob: 8; the number of fish caught by Mary: 7	the highest number of fish caught =8
Minimum	<b>the score for the better round of golf = the lesser of the score of the round of golf for James and the score of the round of golf for Simon</b>	the score of the round of golf for James: 75; the score of the round of golf for Simon: 80	the score for the better round of golf = 75
Exponentiation (xy)	<b>the result = Xy(the value,3)</b>	the value: 5	the result = 125
Mathematical Constant (ex)	<b>the result = Ex(the value)</b>	the value: 0.3527	the result = 1.42290420813407
Absolute value	<b>the result = Abs(the value)</b>	the value: -80	the result = 80

Function	Example rule	Inputs	Outputs
Natural Logarithm	<b>the result = Ln(the value)</b>	the value: 0.3527	the result = -1.04213744174013
Logarithm Base	<b>the result = Log(the value)</b>	the value: 0.3527	the result = -0.45259454033251
Square Root	<b>the result = Sqrt(the value)</b>	the value: 64	the result = 8
Round	<b>the result = Round(the value,3)</b>	the value: 2.45678	the result = 2.457
Truncation	<b>the result = Trunc(the value,1)</b>	the value: 64.4657	the result = 64.4
Sine	<b>the result = Sin(the value)</b>	the value: 0.3527	the result = 0.345432860836779
Cosine	<b>the result = Cos(the value)</b>	the value: 0.3527	the result = 0.938443465880667
Tangent	<b>the result = Tan(the value)</b>	the value: 0.3527	the result = 0.368091284553421
Inverse Sine	<b>the result = Asin(the value)</b>	the value: 0.3527	the result = 0.360454968099581
Inverse Cosine	<b>the result = Acos(the value)</b>	the value: 0.3527	the result = 1.21034135869532
Inverse Tangent	<b>the result = Atan(the value)</b>	the value: 0.3527	the result = 0.339078136684554

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Use a variable in a mathematical calculation in a rule conclusion](#)
- [Convert a text string into a number or date](#)

## Text function rule examples

Text functions are used with text variables to combine text strings and to extract parts of text strings.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below. Note that text functions are case-insensitive.

## Rule examples

Function	Example rule	Inputs	Outputs	Further information
Contains	<p><b>the account may be a benefit account if</b></p> <p>Contains(the account name, "benefit")</p>	the account name: "Special Benefits"	the account may be a benefit account = true	Check if a text string contains a particular substring
StartsWith	<p><b>the person should be represented if</b></p> <p>StartsWith(the person's name, "Sir")</p>	the person's name: "Sir Lancelot"	the person should be represented = true	Check if a text string contains a particular substring at the start of the string
EndsWith	<p><b>the product uses the ascending sort code if</b></p> <p>EndsWith(the product code, "-ASC")</p>	the product code: "B421-A3N-ASC"	the product uses the ascending sort code = true	Check if a text string contains a particular substring at the end of the string
IsNumber	<p><b>the postcode is a valid Australian postcode if</b></p> <p>IsNumber(the postcode) and Length(the postcode) = 4</p>	the postcode: "2612"	the postcode is a valid Australian postcode = true	Check if a text string is a number
Length	<p><b>the product code is valid if</b></p> <p>Length(the product code) &gt; 8</p>	the product code: "123456789"	the product code is valid = true	Find the length of a text string
Concatenation	<p><b>the screen heading variable for the person = the concatenation of the person's first name &amp; ", " &amp; the person's age &amp; ", " &amp; the person's occupation</b></p>	<p>the person's first name: William</p> <p>the person's age: 20</p> <p>the person's occupation: Student</p>	the screen heading variable for the person = "William, 20, Student"	Combine multiple text strings into a single text variable
Substring	<p><b>customer reference = Substring(customer name, 4, 4)</b></p>	customer name: "maryjane"	customer reference = "jane"	Extract part of a text string

Function	Example rule	Inputs	Outputs	Further information
Text	<b>the customer's age text = Text(the customer's age)</b>	the customer's age: 25	the customer's age text = "25"	Convert a number or date into a text string

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

## Entity and relationship function rule examples

Entity functions are used to perform operations on entity-specific data to produce global results, such as counting the number of instances of an entity, obtaining the highest/most recent or lowest/least recent value of an entity-level variable, and adding up numerical values gathered from each instance of the entity.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below.

### Rule examples

Function	Declarations	Example rule	Inputs	Outputs	Further information
For	Source entity: the child Target entity: the school Relationship type: Many-to-one Relationship text: the child's school	<p><b>the child may apply for a scholarship if</b></p> <p>For(the child's school, the school has a scholarship program)</p> <p><b>the child does not have to go to school if</b></p> <p>in the case of the child's school, the school is closed for a pupil free day</p> <p><b>the child's school name = the school name, in the case of the child's school</b></p>	<p>the child's school: St Mary's; the school has a scholarship program: false</p> <p>the child's school: St Joseph's; the school is closed for a pupil free day: true</p>	<p>the child may apply for a scholarship = false</p> <p>the child does not have to go to school = true</p> <p>the child's school name = St Clare's Public School</p>	<p><a href="#">Refer to entities connected by a to-one relationship</a></p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
			the child's school: St Clare's Public School; the school name: St Clare's Public School		
ForScope	<p>Source entity: the person</p> <p>Target entity: the car</p> <p>Relationship type: One-to-one</p> <p>Relationship text: the person's car</p>	<p><b>the person has a reliable car if</b></p> <p>in the case of the person's car</p> <p>t- h- e</p> <p>n- u- m- b- e- r</p> <p>o- f</p> <p>t- i- m- e- s</p> <p>t- h- e</p> <p>c- a- r</p> <p>h- a- s</p> <p>b- r- o-</p>	<p>the car: NSW001;</p> <p>the number of times the car has broken down: 4</p>	<p>the person has a reliable car = false</p>	<p>Extend the For, For All and Exists functions</p>



Function	Declarations	Example rule	Inputs	Outputs	Further information
ForAll	<p>Source entity: Global</p> <p>Target entity: the child</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the children</p>	<p><b>the playground is empty if</b></p> <p>ForAll(the children, the child is at home)</p> <p><b>the playground is empty if</b></p> <p>ForAll(the children, the child's location = "home")</p> <p><b>the playground is empty if</b></p> <p>each of the children is at home</p>	<p>the child: Sally; the child is at home: true; the child's location: home</p> <p>the child: Molly; the child is at home: true; the child's location: home</p> <p>the child: Elizabeth; the child is at home: false; the child's location: playground</p>	<p>the playground is empty = false</p>	<p>Check that a condition returns true for every instance of an entity</p>
ForAllScope	<p>Source entity: the person</p> <p>Target entity: the cat</p> <p>Relationship</p>	<p><b>the person is happy if</b></p> <p>for all of the person's cats</p>	<p>the cat: Tiger; the cat is happy: false</p> <p>the cat: Kit; the cat is</p>	<p>the person is happy = false</p>	<p>Extend the For, For All and Exists functions</p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
	type: One-to-many Relationship text: the person's cats	a- t i- s  h- a- p- p- y	happy: true the cat: Patch; the cat is happy: true		
ForAllScope (Alias)	Source entity: the person Target entity: the person Relationship type: One-to-many Relationship text: the person's dependents	<p><b>the person has one large party if</b></p> <p>for all of the person's dependents (the dependent)</p> t- h- e  p- e- r- s- o- n- '- s  b- i- rthday  =  t- h- e  d- e- pendent's  b- i- rthday	the person: Tobias; the person's birthday: 3 May the person: Alexandra; the person's birthday: 3 May the person: Victoria; the person's birthday: 5 May	the person has one large party = false	Remove ambiguity when reasoning about more than one instance of the same entity
Exists	Source entity: Global Target	<p><b>the playground has good equipment if</b></p> <p>Exists(the children, the child is happy)</p>	the child: Isabelle ; the child is happy: false	the playground has good equipment = true	Check that a condition returns true for at

Function	Declarations	Example rule	Inputs	Outputs	Further information
	entity: the child Relationship type: One-to-many Relationship text: the children	<p><b>the playground has good equipment if</b></p> <p>at least one of the children is happy</p>	the child: Xavier; the child is happy: true the child: Phoebe; the child is happy: false the child: Rachel; the child is happy: false		least one instance of an entity
ExistsScope	Source entity: the plan Target entity: the product Relationship type: Many-to-many Relationship text: the plan's products	<p><b>the plan has incompatible products if</b></p> <p>ExistsScope(the plan's products)</p>	the plan: Plan 1; the plan's network: Vodafone the plan: Plan 2; the plan's network: Telstra the product: Product 1; the product's network: Optus the product: Product 2; the product's network: Vodafone	the plan has incompatible products = true	Extend the For, For All and Exists functions

Function	Declarations	Example rule	Inputs	Outputs	Further information
ExistsScope (Alias)	<p>Source entity: Global</p> <p>Target entity: the child</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the children</p>	<p><b>the child is a twin if</b></p> <p>ExistsScope(the children, the other child)</p>	<p>the child: Kenneth;</p> <p>the child's date of birth: 2007-10-15;</p> <p>the child's mother: Samantha Jane Smith</p> <p>the child: Benny;</p> <p>the child's date of birth: 2007-10-15;</p> <p>the child's mother: Samantha Jane Smith</p> <p>the child: Jenny;</p> <p>the child's</p>	<p>the child is a twin = true (Kenneth)</p> <p>the child is a twin = true (Benny)</p> <p>the child is a twin = false (Jenny)</p>	<p>Remove ambiguity when reasoning about more than one instance of the same entity</p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
		<p>h- e</p> <p>o- t- h- e- r</p> <p>c- h- i- l- d- '- s</p> <p>d- a- t- e</p> <p>o- f b- i- r- t- h</p> <p>a- n- d</p> <p>t- h- e</p> <p>c- h- i- l- d- '- s</p> <p>m- o- t- h- e- r</p>	<p>date of birth: 2006-01- 02; the child's mother: Samantha Jane Smith</p>		



Function	Declarations	Example rule	Inputs	Outputs	Further information
	<p>person</p> <p>Target entity: the bird</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the person's hated birds</p> <p>Reverse relationship text: the bird's owner</p>	<p>in the case of the bird's owner</p> <pre> I- s- NotMemberOf (- t- h- e c- a- t- , t- h- e p- e- r- s- o- n- ' s h- a- t- e- d b- i- r- d- s- ) </pre>	<p>Chirpy; the bird is a member of the person's hated birds: true</p>	<p>happy = false</p>	<p>membership as a rule input</p>
InferInstance	<p>Source entity: the employee</p> <p>Target entity: the location</p> <p>Relationship type:</p>	<p><b>the location in which the employee works (the employee's local office) exists</b></p> <p><b>InferInstance(the location in which the employee works, the employee's local office)</b></p>	<p>the employee: Gordon; the employee's local office: "London" the</p>	<p>new inferred location: London; the location = "London", the</p>	<p>Infer existence of entities to satisfy the relationship</p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
	<p>Many-to-one Relationship text: the location in which the employee works</p> <p>Reverse relationship text: the employees at the location</p>		<p>employee: Britney; the employee's local office: "London"</p> <p>the employee: Dominique; the employee's local office: "Paris"</p>	<p>employee (Gordon) is a member of the employees at the location (London) = true, the employee (Britney) is a member of the employees at the location (London) = true, the employee (Dominique) is a member of the employees at the location (London) = false</p> <p>new inferred location: Paris; the location = "Paris", the employee (Gordon) is a member of the</p>	

Function	Declarations	Example rule	Inputs	Outputs	Further information
				employees at the location (Paris) = false, the employee (Britney) is a member of the employees at the location (Paris) = false, the employee (Dominique) is a member of the employees at the location (Paris) = true	
InstanceCount	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	<b>the number of children that the claimant has = InstanceCount(the claimant's children)</b>	the child: Anthony the child: Peter the child: Rebecca the child: Fiona	the number of children that the claimant has = 4	Count the number of instances of an entity
InstanceCount-	Source	<b>the number of school students that</b>	the child:	the num-	Count the

Function	Declarations	Example rule	Inputs	Outputs	Further information
InstanceCountIf	<p>entity: the claimant</p> <p>Target entity: the child</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the claimant's children</p>	<p><b>the claimant has = InstanceCountIf (the claimant's children, the child is a school student)</b></p>	<p>Anthony; the child is a school student: false</p> <p>the child: Peter; the child is a school student: false</p> <p>the child: Rebecca; the child is a school student: true</p>	<p>number of school students that the claimant has = 1</p>	<p>number of instances of an entity for which a particular attribute is true</p>
InstanceMaximum	<p>Source entity: the claimant</p> <p>Target entity: the child</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the claimant's children</p>	<p><b>the highest bank balance for a child of the claimant = InstanceMaximum(the claimant's children, the child's bank balance)</b></p>	<p>the child: Max; the child's bank balance: \$50</p> <p>the child: Sophie; the child's bank balance: \$175</p> <p>the child: Katie; the child's bank balance: \$120</p>	<p>the highest bank balance for a child of the claimant = \$175</p>	<p>Get the highest/-most recent value of an entity-level variable</p>
InstanceMaximumIf	<p>Source entity: the company</p> <p>Target entity: the employee</p>	<p><b>the most recent date of employment of a permanent employee by the company = InstanceMaximumIf(the company's employees, the employee's date of employment, the employee is a permanent employee)</b></p>	<p>the employee: David; the employee's date of employment: 2010-01-01</p>	<p>the most recent date of employment of a permanent employee: 2010-01-01</p>	<p>Get the highest/-most recent value of an entity-level variable</p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
	<p>Relationship type: One-to-many</p> <p>Relationship text: the company's employees</p>		<p>s date of employment: 01/01/2006; the employee is a permanent employee: true</p> <p>the employee: Shaun; the employee's date of employment: 24/08/2006; the employee is a permanent employee: false</p> <p>the employee: Anita; the employee's date of employment: 15/05/2006; the employee is a permanent employee: true</p>	<p>permanent employee by the company = 2006-05-15</p>	<p>level variable for which a particular attribute is true</p>
InstanceMin-	Source	<b>the lightest weight for a child of the</b>	the child:	the light-	Get the

Function	Declarations	Example rule	Inputs	Outputs	Further information
InstanceMinimum	entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	<b>claimant = InstanceMinimum(the claimant's children, the child's weight in kilograms)</b>	Harry; the child's weight in kilograms: 15 the child: Sharon; the child's weight in kilograms: 30 the child: Fran; the child's weight in kilograms: 45	the lowest weight for a child of the claimant = 15	lowest-/least recent value of an entity-level variable
InstanceMinimumIf	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	<b>the youngest of the claimant's female children = InstanceMinimumIf(the claimant's children, the child's age, the child is female)</b>	the child: Sam; the child's age: 3; the child is female: false the child: Alex; the child's age: 4; the child is female: true the child: Sharon; the child's age: 6; the child is female: false the child: Paris; the	the youngest of the claimant's female children = 4	Get the lowest-/least recent value of an entity-level variable for which a particular attribute is true

Function	Declarations	Example rule	Inputs	Outputs	Further information
			child's age: 8; the child is female: true		
InstanceSum	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	<b>the total Child Care Benefit payable to the claimant = InstanceSum(the claimant's children, the Child Care Benefit amount for the child)</b>	the child: Mary; the Child Care Benefit amount for the child: \$500 the child: Sam; the Child Care Benefit amount for the child: \$250 the child: Lizzie; the Child Care Benefit amount for the child: \$150	the total Child Care Benefit payable to the claimant = \$900	Add up numerical values gathered from each instance of an entity
InstanceSumIf	Source entity: the claimant Target entity: the child Relationship type: One-to-many Rela-	<b>the total cost of boarding school fees for the claimant = InstanceSumIf(the claimant's children, the annual school fees for the child, the child attends a boarding school)</b>	the child: Sally; the annual school fees for the child: \$18000; the child attends a boarding school:	the total cost of boarding school fees for the claimant = \$33000	Add up numerical values gathered from each instance of an entity for which a particular attribute is true

Function	Declarations	Example rule	Inputs	Outputs	Further information
	<p>relationship text: the claimant's children</p>		<p>true</p> <p>the child: James; the annual school fees for the child: \$15000; the child attends a boarding school: true</p> <p>the child: Bob; the annual school fees for the child: \$10000; the child attends a boarding school: false</p>		
InstanceValueIf	<p>Source entity: Global</p> <p>Target entity: the child</p> <p>Relationship type: One-to-many</p> <p>Relationship text: the children</p>	<p><b>the name of the oldest child = InstanceValueIf(the children, the child's name, the child's age = the age of the oldest child)</b></p>	<p>the age of the oldest child = 8</p> <p>the child: Sam; the child's age: 3</p> <p>the child: Alex; the child's age: 4</p> <p>the child: Sharon; the child's age: 6</p>	<p>the name of the oldest child = Paris</p>	<p>Get a value from a unique entity instance</p>

Function	Declarations	Example rule	Inputs	Outputs	Further information
			the child: Paris; the child's age: 8		
InstanceEquals	<p>Source entity: the product</p> <p>Target entity: the product</p> <p>Relationship type: Many-to-many</p> <p>Relationship text: the products</p>	<p><b>the product is a duplicate if</b></p> <p>ExistsScope(the products, the other product)</p> <p>the product's code = the other product's code and</p> <p>InstanceEquals(the product, the other product)</p>	<p>the product: Product A; the product's code: TD2010</p> <p>the product: Product B; the product's code: SM2031</p> <p>the product: Product A; the product's code: TD2010</p>	<p>the product (Product A) is a duplicate = true</p> <p>the product (Product B) is a duplicate = false</p>	Compare instances of the same entity
InstanceNotEquals	<p>Source entity: the employee</p> <p>Target entity: the employee</p> <p>Relationship type: Many-to-many</p> <p>Relationship text: the employees</p>	<p><b>the employee has a conflicting ID if</b></p> <p>ExistsScope(the employees, the other employee)</p> <p>the employee's ID = the other employee's ID and</p> <p>InstanceNotEquals(the employee, the other employee)</p>	<p>the employee: Harry; the employee's ID: RN6710</p> <p>the employee: Will; the employee's ID: RN5812</p> <p>the</p>	<p>the employee (Harry) has a conflicting ID = false</p> <p>the employee (Will) has a conflicting ID = true</p> <p>the employee (Kate) has</p>	Compare instances of the same entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
			employee: Kate; the employee's ID: RN5812	a conflicting ID = true	

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

### Temporal reasoning function rule examples

Temporal reasoning functions are used in rules to compute results for, and express relationships that involve, attributes over multiple periods.

Be sure to use the exact syntax for these functions including spacing and parentheses as specified below.

#### Rule examples

Function	Example	Further information
IntervalCountDistinct	<b>the client's distinct address count = IntervalCountDistinct (2005-07-01,2006-07-01,the client's address)</b>	Calculate the number of distinct values for a variable in a time period
IntervalCountDistinctIf	<b>the client's distinct address count = IntervalCountDistinctIf (2000-01-01,2007-01-01,the client's address,the client is aged over 18)</b>	Calculate the number of distinct values for a variable in a time period only when a condition is true
IntervalDailySum	<b>the amount of benefit payable for the assessment period = IntervalDailySum(2006-07-05,2006-08-01,the daily rate of benefit)</b>	Calculate the sum of a variable in a time period

Function	Example	Further information
IntervalDailySumIf	<p><b>the total amount spent on weekends in December = IntervalDailySumIf(2006-12-01,2007-01-01,the daily amount spent,the day is a weekend)</b></p>	<p>Calculate the sum of a variable in a time period only when a condition is true</p>
IntervalMaximum	<p><b>the maximum rate of benefit during the assessment period = IntervalMaximum(2006-07-05,2006-08-01,the daily rate of benefit)</b></p>	<p>Find the maximum amount in a period</p>
IntervalMaximumIf	<p><b>the maximum rate of benefit payable during the assessment period = IntervalMaximumIf(2006-07-05,2006-08-01,the maximum daily rate of benefit,the client is eligible for the benefit)</b></p>	<p>Find the maximum amount in a period when a boolean attribute is true</p>
IntervalMinimum	<p><b>the minimum rate of benefit during the assessment period = IntervalMinimum(2006-07-05,2006-08-01,the daily rate of benefit)</b></p>	<p>Find the minimum amount in a period</p>
IntervalMinimumIf	<p><b>the minimum rate of benefit payable during the assessment period = IntervalMinimumIf(2006-07-05,2006-08-01,the minimum daily rate of benefit,the client is eligible for the benefit)</b></p>	<p>Find the minimum amount in a period when a boolean attribute is true</p>
IntervalWeightedAverage	<p><b>the average number of children in care = IntervalWeightedAverage(2007-01-22,2007-01-29,the number of children in care)</b></p>	<p>Calculate the average value of a variable in a time period</p>
IntervalWeightedAverageIf	<p><b>the average number of children in care for the weekdays in the assessment period = IntervalWeightedAverageIf(2007-01-22,2007-01-29,the number of children in care,the day is a weekday)</b></p>	<p>Calculate the average value of a variable in a time period when a condition is true</p>
IntervalAlways	<p><b>the client was in jail at all times during the assessment period if</b></p>	<p>Check if a condition is true at all times in</p>

Function	Example	Further information
	IntervalAlways(2006-07-10,2006-07-21,the client was in jail)	the time period
IntervalAtLeastDays	<p><b>the employee has been at work for at least 5 days during the assessment period if</b></p> <p>IntervalAtLeastDays(2007-07-01,2007-07-08,5,the employee was working)</p>	Check if a condition is true for at least the specified number of days in the time period
IntervalConsecutiveDays	<p><b>the employee has been at work for at least 5 consecutive days during the assessment period if</b></p> <p>IntervalConsecutiveDays(2007-07-01,2007-07-08,5,the employee was working)</p>	Check if a condition is true for at least the specified number of consecutive days in the time period
IntervalSometimes	<p><b>the client has been in Australia if</b></p> <p>IntervalSometimes(2007-01-08,2007-01-23,the client was in Australia)</p>	Check if a condition is ever true in the time period
ValueAt	<p><b>the rate of benefit payable on the date of claim = ValueAt (the date of claim,the rate of benefit)</b></p>	Determine a rule attribute on a given date
WhenLast	<p><b>the date the customer's bank balance was last over \$100 = WhenLast(the current date,the customer's bank balance &gt; 100)</b></p>	Find the closest date when an attribute was true
WhenNext	<p><b>the date the customer's bank balance was over \$100 for the first time in 2007 = WhenNext(2007-01-01,the customer's bank balance &gt; 100)</b></p>	Find the closest date when an attribute was true
Latest	<p><b>the amount of benefit paid to the applicant since 1/7/2007 = IntervalDailySum(2007-07-01,Latest()),the amount of the monthly payment)</b></p>	Get a date value equivalent to the latest possible

Function	Example	Further information
		date
Earliest	<b>the amount of benefit paid to the applicant up until 1/7/2007 = IntervalDailySum(Earliest(),2007-07-01,the amount of the monthly payment)</b>	Get a date value equivalent to the earliest possible date
TemporalIsWeekday	<b>the applicant receives money if</b>  TemporalIsWeekday(2006-07-01,2006-07-15)	Calculate the weekdays in a given time period
TemporalOncePerMonth	<b>the applicant receives an allowance if</b>  TemporalOncePerMonth(2006-07-01,2006-08-31,15)	Calculate a specific day in a month for a given time period
TemporalDaysSince	<b>the number of days since it has rained = TemporalDaysSince (the date of the most recent rainfall,the current date)</b>	Calculate the number of days since a given date
TemporalWeeksSince	<b>the number of weeks in the assessment period = TemporalWeeksSince(2007-03-12,2007-04-11)</b>	Calculate the number of weeks since a given date
TemporalMonthsSince	<b>the number of months the mobile phone contract has been in effect = TemporalMonthsSince(the start date of the mobile phone contract,the current date)</b>	Calculate the number of months since a given date
TemporalYearsSince	<b>the child's age = TemporalYearsSince(the child's date of birth,the child's fifth birthday)</b>	Calculate the number of years since a given date
TemporalAlwaysDays	<b>the employee has been at work for the last 4 days if</b>  TemporalAlwaysDays(4,the employee was working)	Check if a condition is true for all of a specified number of preceding days

Function	Example	Further information
TemporalConsecutiveDays	<p><b>the customer's bank balance has exceeded \$50 for at least 2 consecutive days in the last 5 days if</b></p> <p>TemporalConsecutiveDays(2,5,the customer's bank balance exceeds \$50)</p>	Check if a condition is true for at least the specified number of consecutive preceding days
TemporalSometimesDays	<p><b>the customer's bank balance has exceeded \$100 in the last 4 days if</b></p> <p>TemporalSometimesDays(4,the customer's bank balance exceeds \$100)</p>	Check if a condition is ever true within a specified number of preceding days
TemporalAfter	<p><b>the July 2005 rate changes apply if</b></p> <p>TemporalAfter(2005-06-30)</p>	Check if a condition is true after a given date and false on and before
TemporalBefore	<p><b>the pre-2007 Ministerial Determination is in force if</b></p> <p>TemporalBefore(2007-01-01)</p>	Check if a condition is true before a given date and false on and afterwards
TemporalOn	<p><b>the New Millennium Promotion is available to customers if</b></p> <p>TemporalOn(2000-01-01)</p>	Check if a condition is true on a given date and false before and afterwards
TemporalOnOrAfter	<p><b>the 2007 Ministerial Determination is in force if</b></p> <p>TemporalOnOrAfter(2007-01-01)</p>	Check if a condition is true on or after a given date and false before
TemporalOnOrBefore	<p><b>the pre-Christmas price list applies if</b></p>	Check if a con-

Function	Example	Further information
	TemporalOnOrBefore(2007-12-24)	dition is true on and before a given date and false afterwards
TemporalFromStartDate	<b>the person's most recent employer = TemporalFromStartDate(the person's jobs, the job's start date, the job's employer)</b>	Get a temporal attribute from entity instances with values from the start date
TemporalFromEndDate	<b>the person's effective first aid certificate ID = TemporalFromEndDate(the person's first aid certificates, the first aid certificate's expiry date, the first aid certificate ID)</b>	Get a temporal attribute from entity instances with values up until the end date
TemporalFromRange	<b>the person's effective security clearance = TemporalFromRange(the person's security clearances, the security clearance's start date, the security clearance's expiry date, the security clearance)</b>	Get a temporal attribute from entity instances with values from the start date until the end date

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

## Certain and known operator rule examples

The known and certain operators are used on rule conditions and cause the condition to evaluate a predictable way when the underlying attribute in the condition has a particular value.

Operator	Example
certain	<p><b>the applicant is eligible for the benefit if</b></p> <p>it is certain whether or not the applicant is entitled to a payment or</p> <p>the applicant's eligibility status is certain</p>
uncertain	<p><b>the outcome is unclear if</b></p> <p>it is uncertain whether or not the means have been achieved or</p> <p>the status of the investigation is uncertain</p>
known	<p><b>the interview has been completed if</b></p> <p>it is known whether or not the applicant is eligible for a payment or</p> <p>the applicant's rate of benefit is known</p>
unknown	<p><b>the generic heading should be shown if</b></p> <p>it is unknown whether or not the applicant is eligible or</p> <p>the applicant's rate of entitlement is unknown</p>
currently known	<p><b>the income details are available if</b></p> <p>the applicant's income is currently known</p>

The **known** operator is used to ascertain whether an issue has been addressed by the user. The known operator creates a condition that evaluates to true when the attribute used by the condition has a value, no matter what that value is. It is commonly used in procedural rules that drive an investigation. For example, forcing attributes to be known in a particular order before determining a goal (eg forcing a particular screen flow rather than letting the rulebase dictate the screen display order).

The **currently known** operator is used to test whether an attribute is known, without causing it to be brought up in the question search and asked of the user, ie it will test the *current* state of the attribute. In the example above, if the applicant's income is unknown, the conclusion will be inferred to false.

The **unknown** operator is most commonly used for defaulting values in the rulebase where the user has the option of providing an overriding value (either directly or through an inferred attribute). For example:

**the team's game point total = the team's points from round 1 + the team's points from round 2 + the team's points from round 3**

<b>the team's points from round 1</b>	
<b>0</b>	the team's points from round 1 (as recorded by the team) is unknown
<b>the team's points from round 1 (as recorded by the team)</b>	<b>otherwise</b>

<b>the team's points from round 2</b>	
<b>0</b>	the team's points from round 2 (as recorded by the team) is unknown
<b>the team's points from round 2 (as recorded by the team)</b>	<b>otherwise</b>

<b>the team's points from round 3</b>	
<b>0</b>	the team's points from round 3 (as recorded by the team) is unknown
<b>the team's points from round 3 (as recorded by the team)</b>	<b>otherwise</b>

These operators can be used to control the visibility of attributes and text on screens and in generated documentation.

TIP: The localized syntax for these functions may be viewed:

- by clicking here for [US English](#) and [other languages](#); or
- at **Help | Function Reference** in Oracle Policy Modeling, in the rule language set for the rulebase project.

See also:

- [Truth tables](#)
- [Decide whether to allow uncertainty in user answers](#)

## File extensions

Oracle Policy Modeling projects contain the following types of files:

File type	File extension	Description	Location in Development folder
Project	.xprj	The master project file records the file and folder structure of the project.	\
Microsoft Word	.doc	Microsoft Word files contain rules.	\Rules
Microsoft Excel	.xls	Microsoft Excel files contain rules. XLS files are also used for translation mappings.	\Rules or \Translations
Generated rule format	.xgen	An XGEN file contains the generated rule format for a rule document. These XGEN files are used to build rulebase files for use with the Oracle Determinations Engine. Each time a Word or Excel rule file is compiled, the XGEN file corresponding to that file is overwritten with a new one containing the updated set	\Rules or \Translations

File type	File extension	Description	Location in Development folder
		of rules. Each translation document also has an associated XGEN file. XGEN files are in XML format.	
Screens	.xint	Screens files contain screen definitions.	\Interviews
Properties	.xsrc	Properties files contain attribute, entity and relationship properties.	\Properties
External data model	.xsrc	Source files contain data models compiled from an external application such as Siebel.	\
Visualization	.dml	Visualization files contain visualizations of the rules in the form of tree diagrams.	\Visualizations
Test script	.tsc	Test script files are XML files which contain test cases and the set of outcome attributes that will be used by the test cases.	\Test Scripts
PDF	.pdf	PDF files, such as policy documents, can be included as necessary in the project.	\Documents
Screens	.exs	The EXS file is an XML file which contains information relating to the screen definition in the rulebase (ie the data about question screens, summary screens, screen orders and screen flows). The EXS file only provides deployment information for Oracle Web Determinations investigations.	\output
Language	.stxt	The STXT file is the XML language file which contains information on the presentation form to be used for all attributes. (For boolean attributes, this is the positive, negative, question and uncertain forms.)	\output
Flows	.flows	The flows file is an XML file which describes any flows defined for the project.	\output
Metadata	.metadata	The metadata file contains any metadata which has been defined about the project or attributes. This is an XML file. (Metadata information about screens or controls is contained in the EXS file.)	\output
Rulebase	.xml	The rulebase file is the compiled rulebase file created in Oracle Policy Modeling. It is an xml file describing the rules and is required by the Oracle Determinations Engine. This file contains a definition for all attributes and rules (only basic attribute identifiers are available) and as such is the primary file which acts as an anchor for any other required files.	\output
Interface	.xrbd	The XRBD file is the automatically generated interface file that is loaded when you link a module to a project. It is included in the .RMOD file when you build it.	\output
Module	.rmod	The RMOD file is a ZIP file of the project that contains the external data model (ie all entities and relationships, and attributes that have public names). This module file allows the rulebase to be shared with other rulebases.	\output
Compressed	.zip	Building a project in Oracle Policy Modeling will automatically build a <pro-	\output

File type	File extension	Description	Location in Development folder
(zipped) folder		ject>.zip file in the output folder. This package of all of the individual output components of a rulebase is the preferred method of deploying rulebases rather than as individual files. Additionally, any files in the include folder will be added to the rulebase zip. See <a href="#">Include extra files in the build</a> .	

## Truth tables

### AND truth table

The conjunction **and** produces the following set of possible outcomes:

P	Q	P AND Q
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

As you can see from the truth table, it is only if both conditions are true that the conjunction will equate to true. If one or other or both of the conditions in the conjunction are false, then the conjunction equates to false.

Also notice that when conditions are connected by AND that a single condition being false is sufficient for the conclusion to be false, but a single condition being true is not sufficient for the conclusion to be true (you would need to know the value of the other conditions).

In this way, the conjunction itself has its own truth value which is distinct from each of the conditions contained within (ie one of the conditions may be true, but the value of the conjunction is false).

### OR truth table

The disjunction **or** produces the following set of possible outcomes:

P	Q	P OR Q
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE

P	Q	P OR Q
FALSE	FALSE	FALSE

As you can see, if one or other of the conditions is true, the disjunction will equate to true. It is only if both conditions are false that the entire collection equates to false.

Also notice that when conditions are connected by OR that a single condition being true is sufficient for the conclusion to be true, but a single condition being false is not sufficient for the conclusion to be false (you would need to know the value of the other conditions).

A disjunction also has its own truth value distinct from its conditions (ie one of the conditions may be false, but the value of the disjunction is true).

## Uncertain truth tables

The following truth tables show how uncertainty works:

P	Q	P AND Q
TRUE	UNCERTAIN	UNCERTAIN
UNCERTAIN	TRUE	UNCERTAIN
FALSE	UNCERTAIN	FALSE
UNCERTAIN	FALSE	FALSE
UNCERTAIN	UNCERTAIN	UNCERTAIN

P	Q	P OR Q
TRUE	UNCERTAIN	TRUE
UNCERTAIN	TRUE	TRUE
FALSE	UNCERTAIN	UNCERTAIN
UNCERTAIN	FALSE	UNCERTAIN
UNCERTAIN	UNCERTAIN	UNCERTAIN

The uncertain operator causes the condition to return true only if its value is uncertain. A condition using the uncertain operator returns false if the underlying value is not uncertain.

## Basic English grammar

### Parts of speech

The parts of speech that are relevant to the writing of attribute text for use in Oracle Policy Modeling rules are listed below.

## Noun

A noun is a naming word. It names a person, place, animal, thing or abstract idea. Examples: *doctor, Sydney, cat, ticket, accident.*

## Verb

A verb is a word which expresses an action (doing something) or a state (being something). Examples: *run, sit, tell, decide, be.*

## Adverb

An adverb is a word that describes the verb. It can indicate the manner, time or place of the action or state. Examples: *quickly, softly, previously, yesterday, often.*

## Pronoun

A pronoun is a word that takes the place of a noun. It is usually used to avoid repetition of the noun. Examples: *it, I, he, she, you, my, him, us, theirs.*

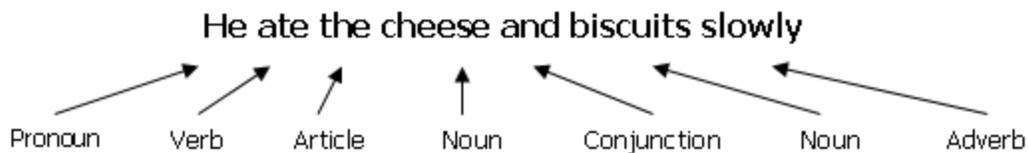
## Conjunction

A conjunction is a word that links words or phrases together. Examples: *and, or, but, so, because.*

## Article

An article is a word that introduces a noun. Examples: *the, a, an.*

For example:



## Apostrophes

Many people are confused about when to use the apostrophe. It is important that you understand how to use apostrophes correctly when writing Oracle Policy Modeling attribute text.

There are two main uses of the apostrophe:

1. to form the possessives of nouns
2. to show omission of letters

Apostrophes are not used:

- for possessive pronouns
- for noun plurals

### **Forming the possessives of nouns**

Apostrophes are used when indicating the possession or ownership of nouns. Follow the rules below when making the possessive of a noun:

- add 's to singular forms of the noun  
*the girl's coat*  
*James's house*
- add 's to plural forms of the noun that do not end in –s  
*the children's toys*  
*the geese's feathers*
- add ' to plural forms of the noun that end in –s  
*five days' work*  
*houses' fences*
- add 's to the last noun to show joint possession of an object  
*Sally and John's dog*  
*the individual and the company's agreement*

### **Showing omission of letters**

Contractions are formed by combining one or more words into a new word, usually by omitting one or more letters. The apostrophe is used to indicate this omission.

Examples of contractions are:

- it's = it is
- you'll = you will
- who's = who is
- shouldn't = should not
- let's = let us

Note that contractions are used in more informal styles of writing and speech and should not be used in Oracle Policy Modeling attributes.

## **Rule principles for Oracle Policy Modeling**

There are several principles that must be followed when writing rules using Oracle Policy Modeling. There are also five axioms that apply to the operation of rules in the Oracle Determinations Engine.

### **Principles for rule authoring in Oracle Policy Modeling**

In order to ensure the quality of rules created, Oracle Policy Modeling enforces a number of principles in rule authoring. The most important of these are:

1. Each conclusion must only be stated once. This is to avoid conflicting logic, for example if Rule 1 stated A is true if B is true, and Rule 2 stated A is false if C is true, if both B and C were true it would be impossible to determine the outcome of

A.

2. Each rule must have a comprehensive statement of conditions (including any reliance on other rules).
3. Each component of the rule must be clearly identifiable. The conclusion, conditions and any logical operators (and/or etc) must be separated for clarity.
4. Each condition must itself be logically complete in order to determine the value of the condition. This also means that each boolean attribute must be worded as a complete sentence.
5. Every rule must be knowable. It should not be possible for the rulebase not to know the outcome if all data has been provided. The rulebase should be a complete statement of the rules.
6. The order in which information is presented should not change the outcome of the rules. The rulebase will always give the same outcome in the same situation regardless of the order in which the information is collected.
7. Addition of new data should not change the outcome. If data is relevant to making a decision or could alter the outcome of a decision, the Oracle Determinations Engine will require that data to be provided before making the decision.

### Axioms for the operation of rules in Oracle Determinations Engine

There are five axioms that need to be observed for correct rules system operation. While it may certainly be possible to construct a functioning rulebase by violating one or more of these axioms, the resulting rulebase will be hard to maintain, hard to test, fragile and unreliable. These axioms have guided Oracle Policy Modeling product development and can be recognized in the types of rules that can be created using Oracle Policy Modeling and the error messages that appear if one of these axioms is violated.

The five axioms are:

#### **Axiom 1: Order independence**

Any conclusions reached by the business rules management system must be independent of the order in which information is provided.

This is reasonably self-evident, but many business rules management systems fail this basic test. By providing order independence, a safe contract is provided between the rulebase and the user interface designer (the screen designer can order attributes on screens however they desire), as well as the persistence layer (the order of attributes do not have to be preserved).

#### **Axiom 2: No memory**

If the input attributes are changed, no influence caused by the previous values may persist.

To put it another way, changing a value of an attribute within a session should be equivalent to inputting the entire set of attributes into a new session. This axiom provides both for Oracle Determinations Server (which relies on this axiom), as well as load-balancing web servers that support session failover. Event rules in the current system can violate this axiom (worst offenders are warning events), and should be used (and handled) with care.

#### **Axiom 3: Reverse entropy**

Adding new information can only lead to conclusions being reached, not forgotten or changed.

If the rulebase has concluded something, then the introduction of new information cannot result in a new conclusion – only by changing existing information can a conclusion be changed or forgotten.

One construct that this axiom forbids is the "not known" operator in a condition of a rule – evaluating to true if the attribute has not been collected yet, and false if it has. This is a dangerous construct, as it involves reasoning on the basis of the order that information is presented (a clear violation of Axiom 1), as well as violating this principle that underlies every other operator.

#### **Axiom 4: Every conclusion must be knowable**

Every rule must be able to conclude a result given sufficient information.

Axiom 3 introduces the concept that a business rules management system proceeds from conclusions being unknown to known. This axiom requires that they can actually get to the known state. A business rules engine that cannot conclude a value is incomplete – its inability to conclude a value results in rulebase looping or "goal exhaustion", the equivalent of a software crash. If a rule set has insufficient specification to cover every possibility, then it must at least return "uncertain" indicating the engine's uncertainty as to what the result should be.

#### **Axiom 5: No multiply proven attributes**

An attribute cannot have multiple proofs. An attribute with multiple proofs causes a number of violations of previous axioms, as well as introducing some new problems.

Firstly, unless the two proofs are completely distinct (and open-ended), there will be a race condition that violates Axiom 1 (ie they will race to see which one fires first, and the outcome can change depending on which rule fires first). Trying to band-aid this is impossible – for example if you allow one rule to have priority over the other, such that even if the other rule has already fired first, the primary rule can override it, then you fail Axiom 3.

If the rules are completely disjunctive, but do not cover every possibility, then you violate Axiom 4. If the rules are completely disjunctive, and cover every possibility, you still run into trouble because it is non-determinative which rule will be asked first by the `GetNextQuestion` runtime mechanism. Furthermore, this could change between different compiles of the rulebase, different versions of Oracle Policy Modeling, and so forth.

The other problem is that completely disjunctive rules are hard to maintain – if you make modifications to any of the rules, you have to update the other rules to maintain the strict disjunction, otherwise the rulebase will violate one of the other axioms.

All in all, multiply proven attributes will end up creating a broken rulebase unless you use rule fragments to specify the order in which the rules should apply. For more information see [Prove an attribute using multiple rules](#).

This axiom also applies to [shortcut rules](#). Using a shortcut rule to prove an intermediate attribute is a violation for the same reasons as presented above. A regular shortcut rule – ie one that proves a base attribute – is an interesting side issue to explore. Firstly, multiple shortcut rules proving a base level attribute can cause Axiom 1 violations. Secondly, a user-provided value for a base level attribute should have higher priority than a shortcut rule concluding the same attribute – otherwise there would be a violation of Axiom 3. In other words, introducing a new attribute could fire the shortcut rule, overriding an existing user answer, possibly causing an existing conclusion to change. Logically, the result from a shortcut rule should be treated as though it were a base level attribute – just merely having provided a shortcut for setting it. The user is then free to override this value when appropriate.

### Text substitution principles

The substitution of text in attributes and on screens follows the principles below.

#### 1. Text substitutions for attributes use the largest possible match

For example, if we had two substitution variables:

- the child = Bart , and
- the child's pet = Santa's little helper

then the attribute "the child's pet's is a dog" will be substituted as " Santa's little helper is a dog".

#### 2. Text substitution for attributes are by whole word only

For example, if we have the substitution variable:

- the person = Bob

then:

- "the personality disorder" remains "the personality disorder"
- "in respect of the person, the eligibility criteria has been met" becomes "in respect of Bob, the eligibility criteria has been met"
- "the person's car" becomes "Bob's car"

### 3. Text substitutions are case sensitive

If you have a substitution variable:

- the person = Sam

then:

- "the person's dog" becomes "Sam's dog", but
- "The person's dog" remains "The person's dog"

### 4. Text substitution is conditional on the substitution variable's value being known

For example, if we had the substitution variable

- the person = unknown

then "the person's dog" remains "the person's dog"

Note, however, that variables in captions, labels, screen titles etc will always be substituted even if the value is unknown.

### 5. Substitution variables must be in the same entity as the attribute being substituted

For example, if we had an entity 'the child' and a substitution variable 'the condition' in the global entity, then the following attribute won't substitute:

- "the condition the child is suffering from"

You can work around this by inferring the value of the condition's name down to the child entity. For example,

- the child's condition = the condition (ie "the child's condition the child is suffering from")

This restriction also applies to text substitutions on screens – they all must be in the same entity as the screen itself.

## Value conditions for screen flow connections

The allowable values for screen flow connections are specified below. These conditions are validated at compile for correctness.

For booleans:

[true|false|yes|no|y|n|unknown|uncertain]

For example, true

For dates:

[(>|>=|=|<=|<|<>|!)yyyy-MM-dd]

For example, >= 2005-06-12

For date-times:

[(>|>=|=|<=|<|<>|!)yyyy-MM-dd hh:mm:ss]

For example, 2010-03-26 22:04:12

For time of days:

[(>|>=|=|<=|<|<>|!)hh:mm:ss]

For example, 19:00:00

For numbers:

[(>|>=|=|<=|<|<>|!)any number]

For example, = 50000

For text comparisons:

[(=|!|<>|not)"any text"]

For example, the text value is case specific.

NOTES:

- a. ! means not equal to (the equivalent of <>)
- b. You can join comparisons together using 'and'. This allows you to test ranges, for example:  
>1000 and <=2000  
>2006-06-30 AND <=2007-07-30
- c. You can also join comparisons together using 'or'. For example,  
<1000 or uncertain  
"unemployed" or "student"  
This is necessary because you can't have two connections from a decision shape to the same shape.
- d. When using both 'and' and 'or' there are no parenthesis, so conditions are evaluated using an order of operations similar to addition/multiplication in maths. In this case, 'AND' has a higher precedence than 'OR', for example:  
"A and B or C and D" is evaluated as "(A and B) or (B and C)"  
"A or B and C" is evaluated as "A or (B and C)"  
The priority of OR versus AND means that you can always replace two separate connections with a single connection using the word OR.

## BI Publisher code for Oracle Policy Modeling

This topic shows the format that is required when using BI Publisher with Oracle Policy Modeling in order to display attributes (global and entity-level), conditional text and decision reports in an interview document.

For more information on using BI Publisher with Oracle Policy Modeling see [Develop a template for an interview document](#).

### Values and properties of global attributes

The table below shows the BI Publisher format (defined as Code in the Advanced tab) needed for the fields to display various values and properties of global attributes. "attribute\_id" is the public name of the global attribute.

To display	BI Publisher format	Example	Output	Notes
Attribute value (formatted)	<?attribute_id_value?>	<?assessment_date_value?>	3/06/11	Formatted for the region specified in the OPM project (in this example, Australia). Formatted attribute value fields do not need the default BI Publisher code modified. These fields can simply be dragged and dropped into your template from the Field dialog box and require no further modification.
Attribute text	<?attribute_id_text?>	<?improvements_text?>	There are improvements that the customer could make to the children's diet.	Attribute text fields do not need the default BI Publisher code modified. These fields can simply be dragged and dropped into your template from the Field dialog box and require no further modification.
Attribute value (unformatted)	<?attribute_id/-value?>	<?assessment_date/value?>	2011-06-03	BI Publisher provides a range of functions for working with <a href="#">unformatted</a> data, including date settings. To use BI Publishers formatting features, select the Type and Format that you want for the value in the Properties tab for the field.
Attribute question text	<?attribute_id/@question?>	<?improvements/@question?>	Are there improvements that the customer could make to the children's diet?	
Attribute type	<?attribute_id/@-type?>	<?improvements/@type?>	boolean	This will return the attribute type (ie Boolean, text, currency, number, date, date and time, time of day).
Inferred status	<?attribute_id/@inferred?>	<?improvements/@inferred?>	true	This indicates whether the attribute is inferred (ie true) or not (ie false).

## Conditional text and formatting

The table below shows the BI Publisher format needed to display conditional text or formatting. Each element enclosed in brackets (<>) is a separate BI Publisher field. Each of these fields must have the specified code set in the Advanced tab for the field. "attribute\_id" is the public name of the global attribute. "Display text" is the text that you want to have shown when the specified condition is met.

To display	BI Publisher format	Example	Notes
Text when a boolean attribute has a particular value	<?if:attribute_id/-value='unformatted attribute value'?>Display text<?end if?>	<?if:improvements/value='true'?>Please make an appointment with one of our friendly dieticians to discuss how you could improve your family's health.<?end if?>	
Text when a number attribute has a particular value	<?if:number(attribute_id/value)>unformatted attribute value?>Display text<?end if?>	<?if:number(cars_owned/value)>2?>As you have more than two cars, a double garage may not be suitable.<?end if?>	This format/example uses the greater than operator (>) but any of the <a href="#">comparison operators</a> can be used here.
Text when a currency attribute has a particular value	<?if:number(attribute_id/value) <unformatted attribute value?>Display text<?end if?>	<?if:number(total_reimbursement/value) <9?> You have been reimbursed less than the full amount.<?end if?>	This format/example uses the less than operator (<) but any of the <a href="#">comparison operators</a> can be used here.
Text when a date attribute has a particular value	<?if:date(attribute_id/value)<date ('unformatted attribute value')?>Display text<?end if?>	<?if:date(date_of_birth/value)<date ('2000-01-01')?>You were born last century.<?end if?>	This format/example uses the less than operator (<) but any of the <a href="#">comparison operators</a> can be used here.
Certain text when a formatted attribute has a particular value, otherwise displaying alternate text	<?choose:?><?when:attribute_id_value='formatted attribute value'?>Display text when condition met<?end when?><?otherwise:?>Alternate display text<?end otherwise?><?end choose?>	<?choose:?><?when:overall_rating_value='Excellent'?>Your children's diet is very well-balanced. Keep up the good work!<?end when?><?otherwise:?>There are improvements that you can make to your children's diets.<?end otherwise?><?end choose?>	Displays the text "Your children's diet is very well-balanced. Keep up the good work!" when the family's overall health assessment is Excellent. Otherwise displays the text "There are improvements that you can make to your children's diets."  This format/example uses equals (=) in the condition element but you can use other comparison operators, in

To display	BI Publisher format	Example	Notes
			which case, use the formatting described above.
Attribute value formatted a certain way when a formatted attribute has a particular value	<pre>&lt;?choose:?&gt;&lt;?when:attribute_id_value&gt;'formatted attribute value?'&gt;&lt;?attribute_id_value?&gt;&lt;?end when?&gt;&lt;?otherwise:?&gt;&lt;?attribute_id_value?&gt;&lt;?end otherwise?&gt;&lt;?end choose?&gt;</pre>	<pre>&lt;?choose:?&gt;&lt;?when:total_sweets_value&gt;'4'?&gt;&lt;?total_sweets_value?&gt;&lt;?end when?&gt;&lt;?otherwise:?&gt;&lt;?total_sweets_value?&gt;&lt;?end otherwise?&gt;&lt;?end choose?&gt;</pre>	<p>Displays the total number of sweets consumed by the children in bold red format if that number is greater than 4. Otherwise displays the total number of sweets consumed by the children in black non-bold format.</p> <p>This can also be achieved by implementing conditional formatting. See the BI Publisher Users Guide for more information.</p>

### Values and properties of entity-level attributes

The table below shows the BI Publisher format needed to display entity-level attributes in various layouts. Each element enclosed in brackets (<>) is a separate BI Publisher field. Each of these fields must have the specified code set in the Advanced tab for the field. "entity\_id" is the public name of the entity. <entity\_level\_attribute\_element> is a field that takes the same format as those used to display global attributes (see above) but using entity-level attribute values and properties instead.

To display	BI Publisher format	Example	Output	Notes
Entity-level attributes grouped by entity	<pre>&lt;?for-each:entity_id?&gt; &lt;entity_level_attribute_element&gt; &lt;entity_level_attribute_element&gt; &lt;entity_level_attribute_element&gt; ... &lt;?end for-each?&gt;</pre>	<pre>&lt;?for-each:child?&gt; &lt;?child_name_text?&gt; &lt;?child_rating_overall_text?&gt; &lt;?child_rating_overall/@question?&gt; &lt;?child_rating_overall_value?&gt; &lt;?end for-each?&gt;</pre>	<p>The child is Hayden.</p> <p>Hayden's overall star rating is 4.</p> <p>What is Hayden's overall star rating? 4</p> <p>The child is Courtney.</p> <p>Courtney's overall star rating is 2.</p> <p>What is Courtney's overall star rating? 2</p>	<p>This format is also used when displaying entity level attributes in a native Microsoft Word table. That is, the first cell in the row needs to start with the &lt;?for-each:entity_id?&gt; field, and the last cell in the same row needs to end with the &lt;?end for-each?&gt; field (with the entity-level attribute fields in between).</p>
Entity-level	<pre>&lt;?for-each:entity_</pre>	<pre>&lt;?for-each:child?&gt;&lt;?child_</pre>	<p>The child is Hay-</p>	

To display	BI Publisher format	Example	Output	Notes
attributes grouped by attribute	<pre>id?&gt; &lt;entity_level_attribute_element&gt; &lt;?end for-each?&gt; &lt;?for-each:entity_id?&gt; &lt;entity_level_attribute_element&gt; &lt;?end for-each?&gt; &lt;?for-each:entity_id?&gt; &lt;entity_level_attribute_element&gt; &lt;?end for-each?&gt; ...</pre>	<pre>name_text?&gt; &lt;?end for-each?&gt; &lt;?for-each:child?&gt; &lt;?child_rating_overall_text?&gt; &lt;?end for-each?&gt; &lt;?for-each:child?&gt; &lt;?child_rating_overall/@question?&gt; &lt;?child_rating_overall_value?&gt; &lt;?end for-each?&gt;</pre>	<pre>den. The child is Courtney. Hayden's overall star rating is 4. Courtney's overall star rating is 2. What is Hayden's overall star rating? 4 What is Courtney's overall star rating? 2</pre>	
Entity-level attributes sorted alphabetically by entity name	<pre>&lt;?for-each:entity_id?&gt; &lt;?sort:entity_name_id_value;'ascending';data-type='text'?&gt; &lt;entity_level_attribute_element&gt; &lt;entity_level_attribute_element&gt; &lt;entity_level_attribute_element&gt; ... &lt;?end for-each?&gt;</pre>	<pre>&lt;?for-each:child?&gt; &lt;?sort:child_name_value;'ascending';data-type='text'?&gt; &lt;?child_name_value?&gt;: &lt;?child_servings_sweets_text?&gt; &lt;?child_servings_fruit_text?&gt; &lt;?child_servings_dairy_text?&gt; &lt;?end for-each?&gt;</pre>	<pre>Courtney: The number of servings of sweets Courtney eats per day is 5. The number of servings of fruit Courtney eats per day is 2. The number of servings of dairy food Courtney eats per day is 3. Hayden: The number of servings of sweets Hayden eats per day is 0. The number of servings of fruit Hayden eats per day is 3. The number of servings of dairy food Hayden eats per day is 2.</pre>	<pre>This format can be used in native Microsoft Word tables too. That is, the first cell in the row needs to start with the &lt;?for-each:entity_id?&gt; &lt;?sort:entity_name_id_value;'ascending';data-type='text'?&gt; fields, and the last cell in the same row needs to end with the &lt;?end for-each?&gt; field (with the entity-level attribute fields in between).</pre>

## Decision reports

To display a decision report you need to:

- have the attribute selected in the [Decision Reports available for the document](#), and
- have a field ("decision-report template") in your template which defines the structure and format of the decision report, and
- have a field ("call decision report template") in your template which specifies the attribute ("attribute\_id") to give the decision report on.

The table below specifies the BI Publisher code needed to define the two fields described above.

Field	BI Publisher code	Example	Notes
decision-report template	<pre>&lt;?template@inlines:decision-report?&gt; &lt;?if@inlines:"attribute-node"?&gt; &lt;fo:list-block start-indent="{count(ancestor::attribute-node) * 7}mm"&gt; &lt;fo:list-item&gt; &lt;fo:list-item-label&gt; &lt;fo:block&gt;*&lt;/fo:block&gt; &lt;/fo:list-item-label&gt; &lt;fo:list-item-body&gt; &lt;fo:block&gt;&lt;xsl:value-of select="@text"/&gt;&lt;/fo:block&gt; &lt;/fo:list-item-body&gt; &lt;/fo:list-item&gt; &lt;/fo:list-block&gt; &lt;?for-each@inlines:./attribute-node?&gt;&lt;?call-template:decision-report?&gt;&lt;?end for-each?&gt; &lt;?end if?&gt; &lt;?end template?&gt;</pre>	n/a	This field only needs to appear in a template document once.
call decision report template	<pre>&lt;?for-each:/global-instance/attribute_id/-decision-report/*?&gt;&lt;?call-template:decision-report?&gt;&lt;?end for-each?&gt;</pre>	<pre>&lt;?for-each:/global-instance/improvements/decision-report/*?&gt;&lt;?call-template:decision-report?&gt;&lt;?end for-each?&gt;</pre>	

## Troubleshooting guide for using BI Publisher with Oracle Policy Modeling

This topic explains some problems that might be encountered when using BI Publisher with Oracle Policy Modeling and what to do about them.

### Document will not generate

Check the template size. Large file sizes (eg due to images within the document) require more memory allocation. If in doubt, remove images and re-try.

Check all fields have the correct syntax. If in doubt, delete any area that you are concerned about then re-generate the document.

If deploying to an external version of Web Determinations (ie using [Build and Run](#)), check that the document generation server is correctly configured/started (refer to installation instructions for more information). If in doubt, try to [Build and Debug](#) within OPM to determine whether it is an issue with the document template or the external configuration.

### Normal text does not appear

If the text follows a [conditional](#) or [entity-level](#) region, check the previous region has the appropriate end fields.

### Font does not display correctly

Check the font is a predefined font for BI Publisher. If not, you will need to define a font mapping from a base font in the RTF or PDF template to a target font to be used in the published document. More information is available in the BI Publisher documentation/forums.

### Field values and/or conditional text do not appear

Check the syntax matches the recommended syntax (see [BI Publisher code for Oracle Policy Modeling](#)).

If using [conditional text](#), check that the output value in the [generated XML](#) matches the value in the condition, including any formatting if using the [formatted value](#).

If the text follows a conditional or entity-level region, check the previous region has the appropriate end fields.

If using an [entity-level attribute](#):

- ensure that the appropriate entity tags are in place before and after the field, and
- ensure that the entity public name is unique (ie check you do not have an entity and an attribute with the same public name).

Check the font is a standard font. Some fonts are only supported by some outputs (eg Wingdings will not appear in a pdf output). More information on supported fonts is available in the BI Publisher documentation/forums.

### Headings/images do not appear

If using an image, shape, text box or similar:

- check whether the object is grouped (grouped objects will not appear in some outputs), and
- try changing the Text Wrapping setting (depending on the context of how the image is used, some text layout options may impede the display of the image in the generated document).

Check the font is a standard font. Some fonts are only supported by some outputs (eg Wingdings will not appear in a pdf output). More information on supported fonts is available in the BI Publisher documentation/forums.

See also, "Field values and/or conditional text does not appear" troubleshooting section above.

### Text appears on a new line

See the BI Publisher documentation for how to include fields and conditional formatting in-line. Including the text within a table (even a 1x1 table) is a simple way to work around most line break issues (see [sample OPM projects](#) for examples).

## Error when clicking on document link

If you encounter an error when clicking on the document link on the summary screen, the BI Publisher template's Conditional Region settings may have not been correctly defined (for example, if an incorrect data type has been used). If this occurs, you should open the BI Publisher template and make the appropriate adjustments (see [insert conditional text](#) for more information).

See also:

- the Template Builder for Word Help file (available under \Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word) and/or
- the BI Publisher Users Guide (available under \Program Files\Oracle\BI Publisher\BI Publisher Desktop\Template Builder for Word\doc).

## Seeded data in imported projects

An Oracle Policy Modeling project created by importing an existing project will be seeded with the various project folders and documents based on the data in the project interchange file. These will include:

### **A project file**

An Oracle Policy Modeling project file (.xprj) is created when a project interchange file is imported. The project file name will be based on the project interchange file name. Project custom properties and templates for custom properties for other project items will be included in the project file. The project file will be automatically saved on completion of the import.

### **A master data model file**

A master data model file, datamodel.xsrc, is created and inserted in the project's root folder. For each entity, attribute and relationship in the interchange file's <model> section, a corresponding declaration will be added to the master data model XSRC file.

NOTE: While all data model elements defined in a module are exported, only those elements that were defined within the actual project itself are re-imported.

### **Project folders**

For each ruleset, a project folder is created and assigned the properties associated with that ruleset in the interchange file.

### **Rules documents**

A "starter" Microsoft Word or Excel rule document will be created for each unique rule file name (as specified by the /rules/\*/rule/-document element) in a ruleset and allocated to the ruleset's corresponding folder. Each rule in the interchange file will be allocated to a starter rule document (as specified by the rule's ruleset membership and its document element).

### **Rules**

For each rule, appropriate content will be added to the rule document for each non-empty attribute and sub-element specified in the interchange file. This includes:

- Name (this element is mandatory on import)
- Source
- Definition
- Effective date range

- Custom properties
- Rule text

If a rule has a non-empty rule-text element, the importer will use the contained XHTML to recreate the original rule text. The rule/rule-text/@format attribute, and the rule/document/@document-type attribute, governs whether the rule text is represented as a table or regular paragraphs rule, and whether it is included in a Word or an Excel rule document.

NOTE: Rules defined in a module are not exported and therefore are not re-imported.

## Definition of 'relevant' in decision reports

Decision reports show every value that is relevant to the result of a rule. This topic describes the definition of what constitutes a 'relevant' value.

### **Rule 1: A value is relevant if changing it could cause the conclusion of the rule to change**

Example 1:

```
A if
    B and
    C
```

If B is true and C is false, then A is false. In the decision report:

- B is not shown because no matter what you change it to, C's value of false keeps A false.
- C is shown because you could change it to true, and A would become true .

Example 2:

```
Result = InstanceSumIf(Relationship, Condition, Value)
```

With the following sets of conditions and values, the result is 50.

- Condition1 = true
- Value1 = 50
- Condition2 = false
- Value2 = 100

Condition1, Value1, and Condition2 are all relevant due to Rule 1. Value2 is not relevant because no matter what it is set to, the false of Condition2 stops it from having any effect.

### **Rule 2: Where a set of values are not relevant individually (via Rule 1) but could cause the conclusion to change if they change together, then all values in the set are considered relevant**

This is intended to cover situations where attributes are equally relevant to the conclusion, with neither one being enough to actually have an effect if it changes. Using Example 1 above, if B and C are both false, then A is false. Changing either B or C independently does not change the conclusion, so Rule 1 does not apply. However, you could change both of them to true, and it would change the conclusion, so because of Rule 2, they are both considered relevant.

**Rule 3: Where the result is unknown, all values that could be relevant if unknown values became known, are considered relevant**

Example 1:

$$A = B + C$$

If B is unknown and C is 5, then the result is unknown. In the decision report:

- B is relevant because if it changed (to become known), it would affect the outcome (Rule 1)
- C is relevant because if B became known, it would be relevant to the outcome (Rule 3)

No special consideration of uncertainty is required - handling for uncertainty falls naturally out of the above rules.

Example 2:

$$A = B + C$$

If B is uncertain and C is unknown, then the conclusion is uncertain. No matter what value C becomes, A will always be uncertain. In the decision report:

- B is relevant, because if B changed to be unknown, then A would become unknown (Rule 1).
- C is not relevant, because even if it becomes known, it cannot become relevant.

### Keyboard shortcuts for Oracle Policy Modeling

Shortcut keys are keys or key combinations that are provided as a quick and alternative way to access frequently performed actions. The following shortcut keys can be used in Oracle Policy Modeling to insert styles or perform functions:

- [Shortcut keys for Oracle Policy Modeling](#)
- [Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Word](#)
- [Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Excel](#)
- [Shortcut keys for the Screen Flow Editor in Oracle Policy Modeling](#)

### Shortcut keys for Oracle Policy Modeling

Shortcut Key	Function/Navigation
Ctrl+N	New Project
Ctrl+O	Open Project
Ctrl+S	Save Selected Item
Ctrl+Shift+S	Save All
Ctrl+F	Find Model Attribute
Ctrl+Shift+F	Find Document Attribute

Shortcut Key	Function / Navigation
Ctrl+Shift+B	Build
F5	Build and Debug
Ctrl+F5	Build and Run
Ctrl+Alt+B	Build Module
Ctrl+F4	In the top right hand pane, closes the open tab
Ctrl+>	In the top right hand pane, cycles forwards between the open tabs
Ctrl+<	In the top right hand pane, cycles backwards between the open tabs
Ctrl+Tab	In the Attribute Editor, toggles between Common, Custom Properties and Decision Reports tabs. In the Summary Screen Editor and Question Screen Editor, toggles between Common and Custom Properties tabs.
Ctrl+F2	In the Project Explorer, toggles between the Project Explorer tab and the Attribute Usage tab
Ctrl+F3	In the Project Explorer, toggles between displaying the active tab (Project Explorer or Attribute Usage) and hiding the tab

### Access menu items in Oracle Policy Modeling

Access keys are provided for all menu items in Oracle Policy Modeling. Access keys are alphanumeric keys that are used with the Alt key to activate the menu controls. The access key is shown by the underlined character in the text label of the menu item. If the access keys are hidden by default, pressing the Alt key will activate them.

### Access shortcut menus in Oracle Policy Modeling

The application key is used to display the shortcut menu for the selected object in Oracle Policy Modeling. The application key is located between the Windows key and the Ctrl key on a standard keyboard. (If your keyboard does not have an application key, you can use Shift+F10 instead.)

Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Word

Shortcut key	Style/Function
Alt+R	Compiles the Oracle Policy Modeling document
Alt+1	Heading style
Alt+2	Heading 2 style
Alt+3	Heading 3 style

Shortcut key	Style/Function
Alt+B	Blank Line style
Alt+C	Conclusion style
Alt+F	Configuration style
Alt+L	Legend style
Alt+N	Rule Name style
Alt+F1	Level 1 style
F2	Level 2 style
F3	Level 3 style
F4	Level 4 style
F5	Level 5 style
F9	Ignore style
F10	Commentary style
F7	Inserts a shortcut rule
F11	Decreases indent
F12	Increases indent
Alt+D	Opens the Data Model Browser
Alt+G	Adds a variable attribute definition to the rulebase
Alt+I	Inserts an invisible operator
Alt+J	Opens the Attribute Editor
Alt+K	Strips hidden text
Alt+P	Opens the Rule Properties editor
Alt+S	Inserts a silent operator
Alt+Y	Show Oracle Policy Modeling styles in style area (Word 2003 and later)
Alt+Z	Inserts a rule table
Alt+F12	Toggles comment

Shortcut keys for Oracle Policy Modeling styles and functions in Microsoft Excel

Shortcut key	Style/Function
Ctrl+Shift+C	Compiles the Oracle Policy Modeling document
Ctrl+Shift+W	Attribute Type Heading style
Ctrl+Shift+E	Attribute Text Heading style
Ctrl+Shift+T	Legend Key Heading style
Ctrl+Shift+S	Attribute Type style
Ctrl+Shift+D	Attribute Text style
Ctrl+Shift+G	Legend Key style
Ctrl+Shift+I	Conclusion Heading style
Ctrl+Shift+K	Conclusion style
Ctrl+Shift+Y	Condition Heading style
Ctrl+Shift+H	Condition style
Ctrl+Shift+L	Else style
Ctrl+Shift+M	Commentary style
Ctrl+Shift+V	Opens the Attribute Editor

### Shortcut keys for the Screen Flow Editor in Oracle Policy Modeling

Shortcut key	Style/Function
Arrow keys	Moves the cursor, if there are no selected shapes; Moves selected shapes
Shift+Arrow keys	Jumps the cursor towards the next shape in that direction
Space	Selects the shape/connection under the cursor; Clears the selection of shapes; In the Screens/Decisions/Flows tab, adds the selected screen/- decision/flow to the screen flow
Ctrl+Arrow keys	Moves the cursor without moving any selected shapes
Alt+Arrow keys	Resizes the selected shape
Ctrl-Space	Toggles the selection of the shape/connection under the cursor
C	Starts or finishes drawing a connector from/to the shape under the cursor

Shortcut key	Style/Function
Enter	Finishes drawing a connector to the shape under the cursor; In the Screens/Decisions/Flows tab, adds the selected screen/- decision/flow to the screen flow
/	Cycles the selection through the outgoing connectors of the shape under the cursor
F2	Edits the condition text of the selected connector
Alt+R	Errors list

## Formatting of attribute values

Attribute values throughout Oracle Policy Modeling and Oracle Policy Automation are either unformatted (ie using an internal data format), or formatted (ie using the format specified by the [rulebase Region settings](#)).

### Unformatted attribute values

Unformatted attribute values are used:

- In Oracle Policy Modeling itself, wherever data values are entered while creating the rulebase (eg default values on screen controls, maximum or minimum allowed values for variables, etc).
- For any date, date/time or time values used when writing rules in Word or Excel (eg constants, values used in comparisons).
- In the debugger and test case editor, where data values are entered directly (ie not using drop down lists or other pre-defined options for data entry) in the Data and Decision tabs.
- In document templates using BI Publisher, to control the display of images and text.
- For number variables flagged in the Attribute Editor as "Unformatted", where [formatted values](#) would normally be used when displaying those variable's values.

Unformatted values take the following forms:

Attribute type	Unformatted value form	Example
Boolean	true/false	true
Number	x.x (always has at least one decimal place)	3.0
Currency	x.x (always has at least one decimal place, no currency symbol, no comma)	5.0
Text	Any text (with the text surrounded by quotation marks where referenced in rules)	yellow
Date	yyyy-MM-dd	2007-10-25
Time of day	hh:mm:ss	07:47:31

Attribute type	Unformatted value form	Example
Date and time	yyyy-MM-dd hh:mm:ss	2009-08-12 17:30:00

Where the formatting of dates and times are referred to the following conventions are used:

yyyy	four-digit year
MM	two-digit month (01 through 12)
dd	two-digit day of month (0 through 31)
hh	two-digits of hour (00 through 23)
mm	two-digits of minute (00 through 59)
ss	two-digits of seconds (00 through 59)

### Formatted attribute values

Formatted attribute values based on [rulebase region](#) are used:

- In the debugger and the test case editor, when displaying data in the Decision tab and in the Text column of the Data tab.
- For any number or currency values used when writing rules in Word or Excel (eg constants, values used in comparisons).
- In Oracle Web Determinations, when entering and displaying data.
- When displaying information in generated documents.

For example, if your region was set to the United States you would see the following:

Attribute type	Formatted value form (English - United States)	Example
Boolean	Yes/No/Unknown	Yes
Number	x	15
Currency	\$x.xx (two decimal places)	\$123.00
Text	Any text	submarine
Date	MM-dd-yy	6/17/11
Time of day	hh:mm:ss (24 hour clock)	15:21:45
Date and time	MM-dd-yy hh:mm:ss AM/PM (12 hour clock)	6/17/11 3:21:45 PM

Other examples of how this formatting could apply include:

Variable type	Data format specified by rulebase region	Example
Number	Region set to France, which includes the comma as the decimal separator	a Word rule conclusion setting a number variable: the threshold interest rate for savings accounts = 7,75
Currency	Region set to United Kingdom (English), which includes £ currency symbol, comma as thousand separator and full stop as decimal separator	the value for a currency variable "the balance of the applicant's savings account" is set in Oracle Web Determinations to: £25,524.50
Text	Region set to Germany, which accepts any text formatting	the value of a text variable "the person's name" is set in Oracle Web Determinations to: Karla
Date	Region set to Australia, which includes the date format dd/MM/yy	the value of a date variable "the applicant's date of birth" is displayed in the debugger Decision tab as: 25/10/90
Time of day	Region set to Brazil, which includes the time format hh:mm:ss NOTE: The seconds component is optional when entering data in Oracle Web Determinations and the debugger, and will be set to ":00" if omitted.	the value of a time variable "the weekday closing time" is set in Oracle Web Determinations to: 16:45
Date and time	Region set to Japan, which includes the date format yy/MM/dd and time format hh:mm:ss NOTES: If the regional format settings specify an output format for datetimes, all datetimes will be displayed with that format regardless of whether the 'Display seconds' option was ticked for that variable in the Attribute Editor. Also, the seconds component <i>may</i> be optional when entering data in Oracle Web Determinations and the debugger, depending on the regional format settings. If so, it is set to ":00" if omitted.	the value of a date/time variable "the application lodgment time" is set in Oracle Web Determinations to: 09/07/29 10:15:30

Where the formatting of dates and times are referred to the following conventions are used:

yy	two-digit year
MM	two-digit month (01 through 12)
dd	two-digit day of month (0 through 31)
hh	two-digits of hour (00 through 23)
mm	two-digits of minute (00 through 59)
ss	two-digits of seconds (00 through 59)

Note that to find the exact formatting of data values for your rulebase region, Oracle Policy Modeling will check the relevant settings for that region within your system. You may override these individual system settings for a region if required - see the [Oracle Policy Automation Developer's Guide](#) for further details.

TIP: When using BI Publisher to develop a template for an interview document, you can see the formatted and unformatted attribute values in the XML sample data file:

```
- <amount_payable_value>
  <value state="known">$123.00</value>      Formatted value
</amount_payable_value>
<amount_payable_text>The total amount payable is $123.00.</amount_payable_text>
- <amount_payable inferred="true" question="What is the total amount payable?" type="currency">
  <value state="known">123.0</value>      Unformatted value
</amount_payable>
```

See also:

- [Use constant values in rules](#)
- [Oracle Policy Automation Developer's Guide](#) - for the format used in Oracle Determinations Server

## Command line tools

The following table lists the Oracle Policy Modeling command-line tools.

Command line tool	Description
<a href="#">Build</a>	Provides a means of building a rulebase from an Oracle Policy Modeling project using the command line
<a href="#">Regression tester</a>	Provides a means of executing a rulebase project's text scripts using the command line
<a href="#">Batch processor</a>	Provides a means of processing a large number of cases in batch using the command line