Oracle® Policy Modeling User's Guide

User Assistance and How-To Guide

V10.4.7

Oracle Support

Oracle® Policy Modeling User's Guide
Release 10.4 Update 7

E79061-01

September 2016

Copyright © 2009, 2016, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

What's new in Oracle Policy Modeling V10

Version 10.4

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 v11.1.1.3 to v11.1.1.9 (as of 10.4 Update 7)
- 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 Update 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 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

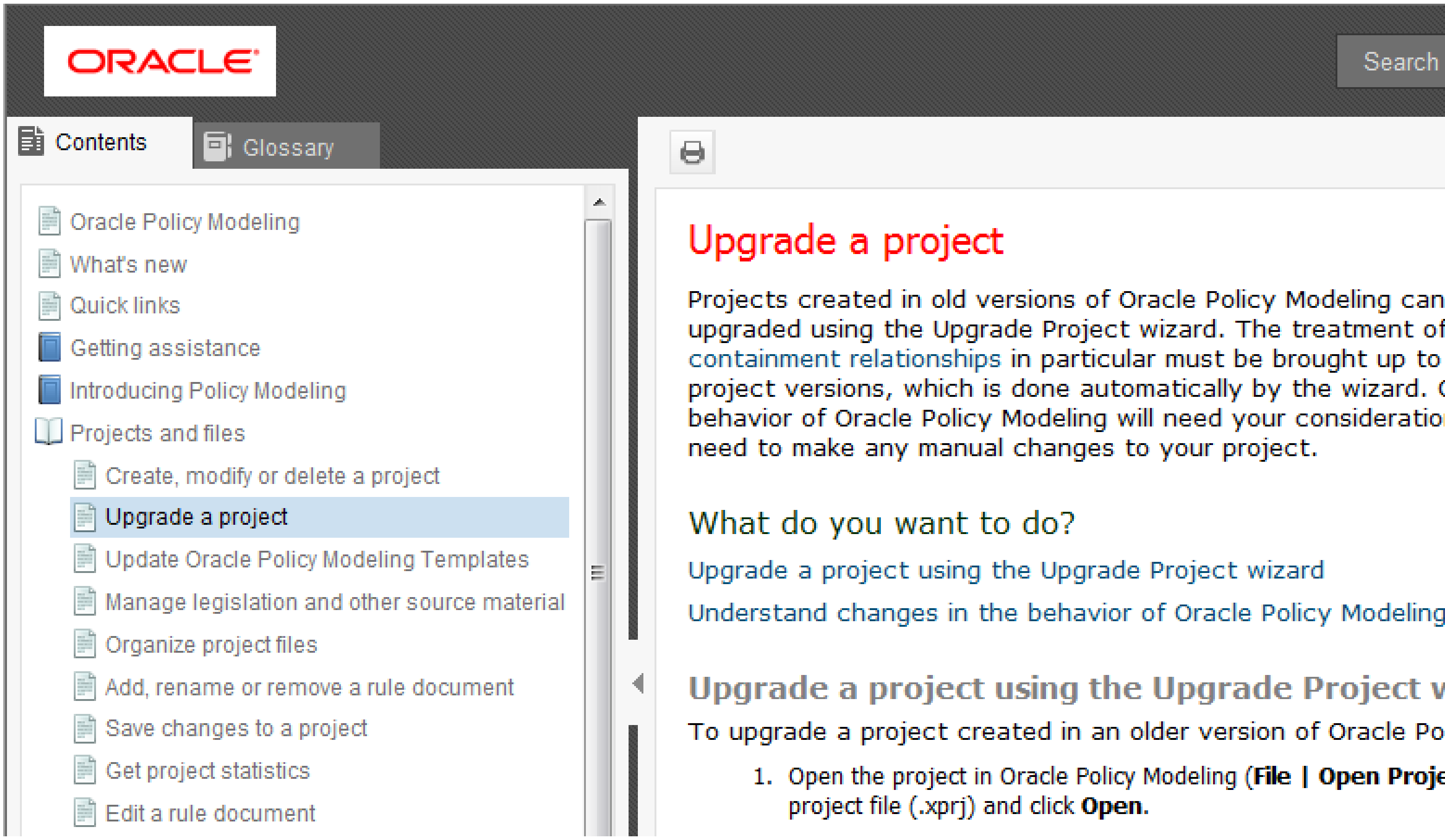
Access the Oracle Policy Modeling User's Guide in another language

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:



Upgrade a project

Projects created in old versions of Oracle Policy Modeling can
upgraded using the Upgrade Project wizard. The treatment of
containment relationships in particular must be brought up to
project versions, which is done automatically by the wizard. O
behavior of Oracle Policy Modeling will need your consideratio
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

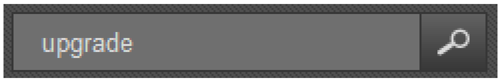
Upgrade a project using the Upgrade Project w

To upgrade a project created in an older version of Oracle Po

1. Open the project in Oracle Policy Modeling (**File | Open Proje**
project file (.xprj) and click **Open**.

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.

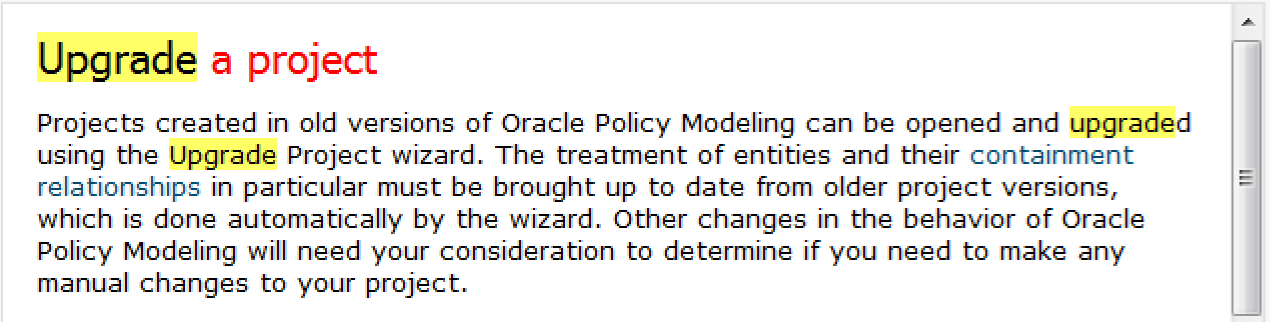


Then click the magnifying glass button, or click **Enter** to search.

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.



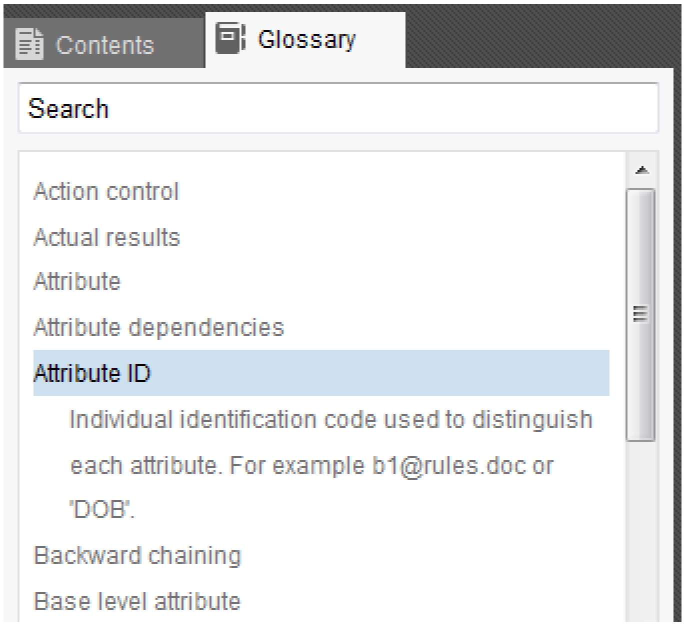
Upgrade a project

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.

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.



Individual identification code used to distinguish each attribute. For example b1@rules.doc or 'DOB'.

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\<project name>\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
  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.

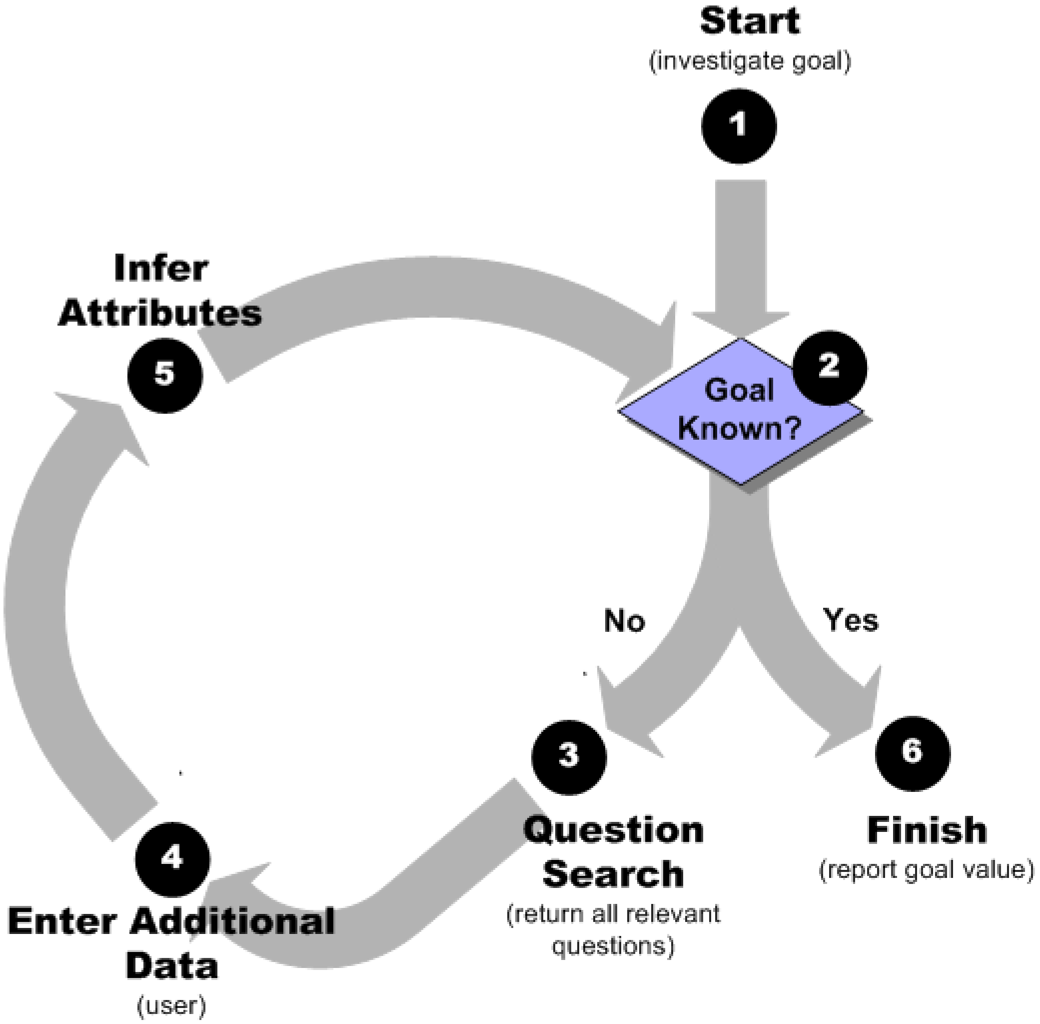
Each time you conduct a new assessment against a rulebase using the Determinations Engine (for example, creating a new interview in Web Determinations) you are creating what is called a **rulebase session**. The Determinations Engine does not maintain the state of its rulebase sessions, so each time 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 rule-base.

What do you want to do?

Create a new project

Open an existing project

Add existing files to a project

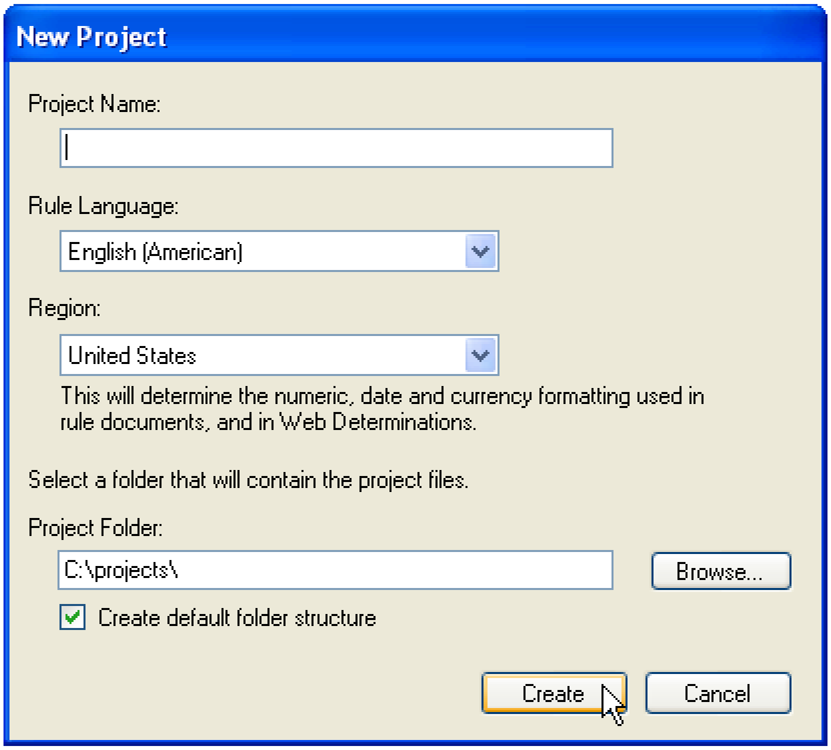
Add new files to a project

Delete a project

Create a new project

To create a new rulebase project:

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



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: C:\projects\

☑ 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 attributes in Excel

If you want to use an attribute's value in the condition or conclusion of a text attribute in an Excel rule table, you now need to put the attribute text in parentheses. Existing projects that use attributes in such rule talbes will need to have parentheses added, otherwise the attribute text 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

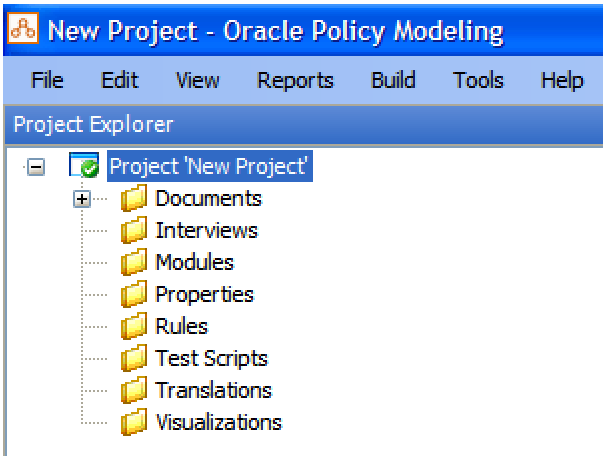
Move files between folders

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

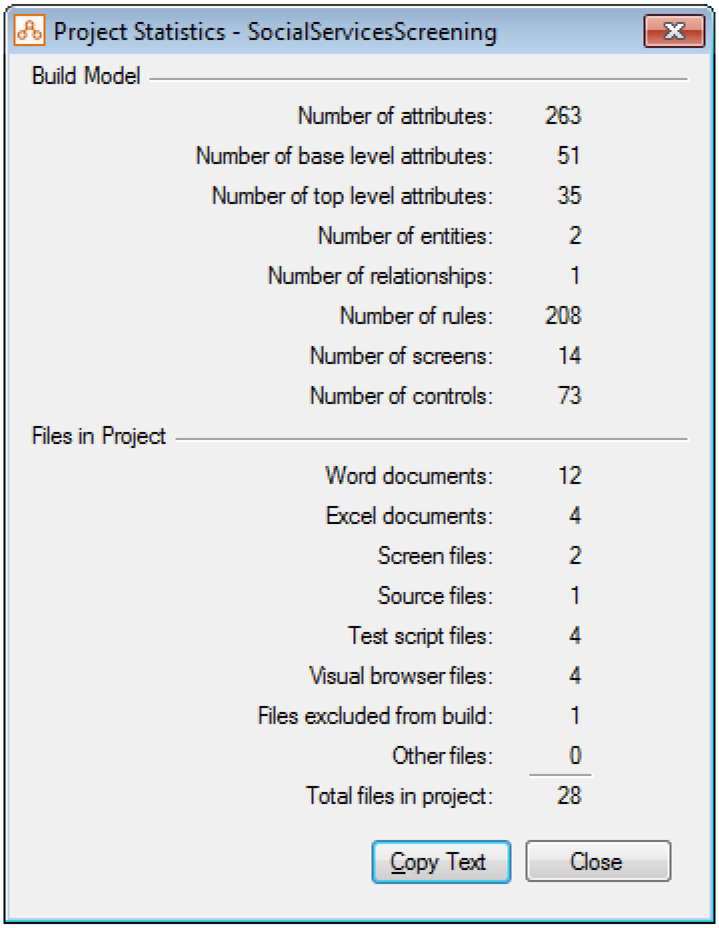
To save all changes to the project:

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

Get project statistics

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

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



Project Statistics - SocialServicesScreening

Build Model

Number of attributes:	263
Number of base level attributes:	51
Number of top level attributes:	35
Number of entities:	2
Number of relationships:	1
Number of rules:	208
Number of screens:	14
Number of controls:	73

Files in Project

Word documents:	12
Excel documents:	4
Screen files:	2
Source files:	1
Test script files:	4
Visual browser files:	4
Files excluded from build:	1
Other files:	0
Total files in project:	28

To copy the text of this dialog, 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

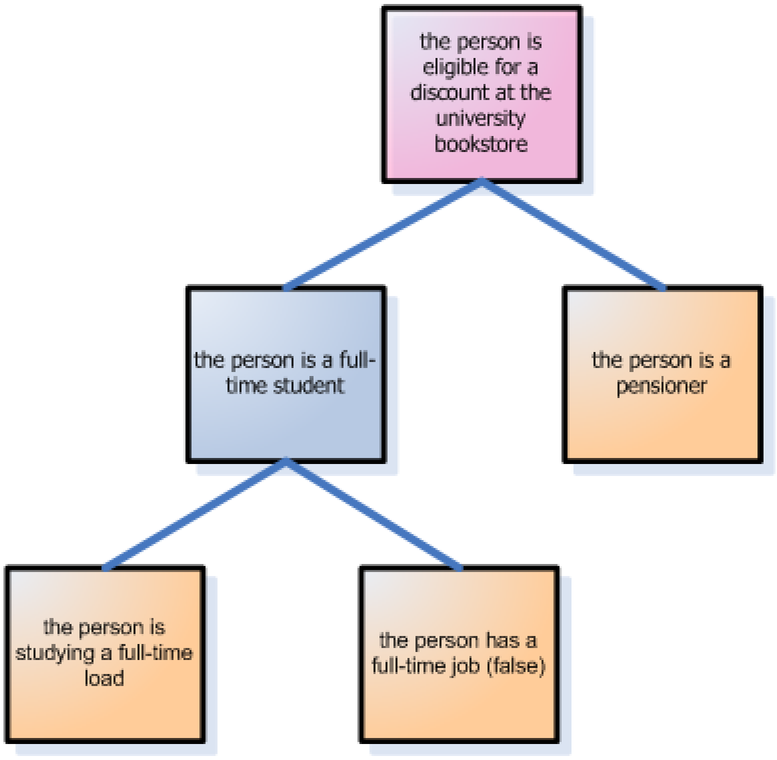
Rule 2

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

Example 1	Example 2
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:

a. intermediate logic is not required in the decision report* and
b. 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

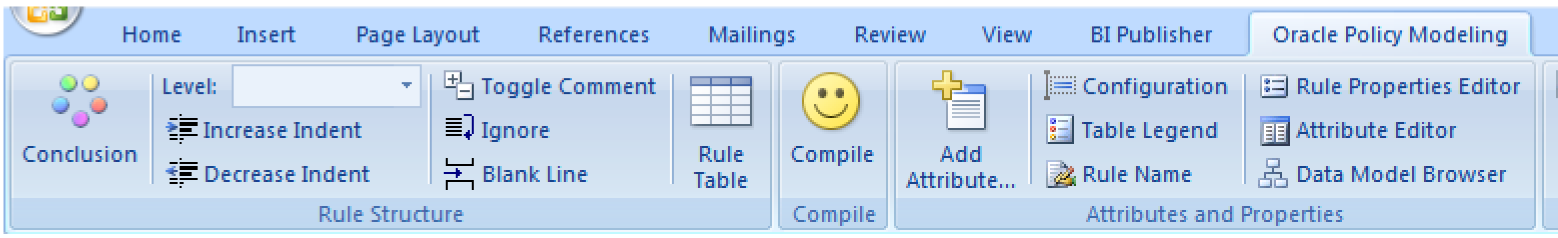
the pet's species = "dog"

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

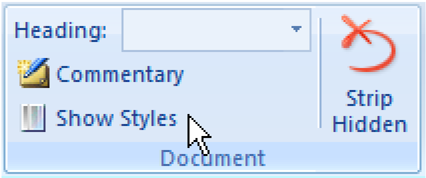
Measurement Units and Style Area

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.)).

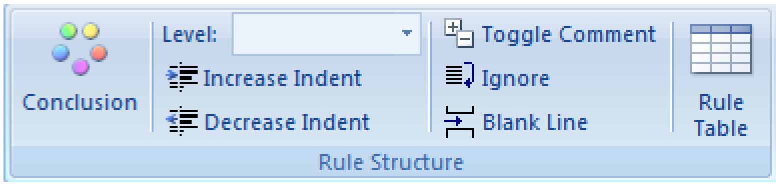
ii. For extremely complex projects containing either very large rule documents (70+ pages) or large numbers of rule documents, you should also turn off auto-saving, backup and background repagination to improve the performance of Microsoft Word with Oracle Policy Modeling.

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:

conclusion	
value	condition
value	condition
value	**otherwise**

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

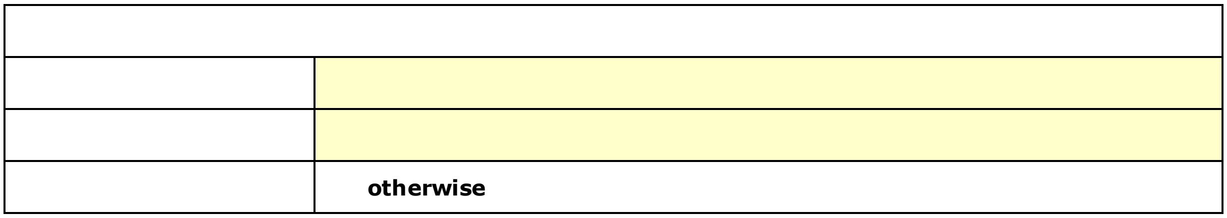
The left hand column is used to specify values (includes mathematical expressions) which will set the value of the conclusion attribute if the condition in the right hand column of the same row is satisfied.

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.



	otherwise

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

the passenger's favorite color	
	otherwise

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
	otherwise

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
uncertain	**otherwise**

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**
Conclusion Heading	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
Condition Heading	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.
Else	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.

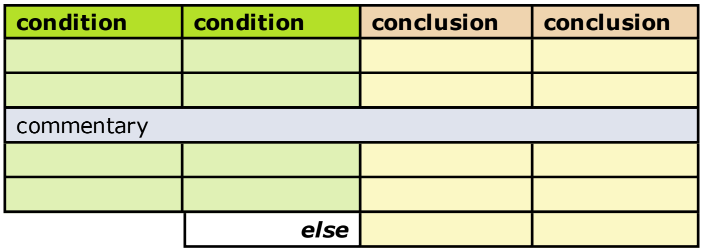
The heading cells are optional. Similarly, the order of cells is irrelevant since each style is unique - as long as the necessary styles are used with valid cell contents.

NOTES:

1. 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.
2. 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.
3. 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.
4. 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.
5. 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:

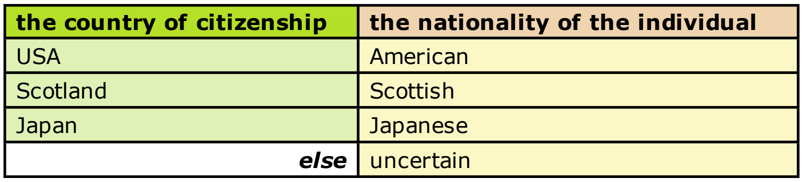


condition	**condition**	**conclusion**	**conclusion**
commentary			
	else		

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.

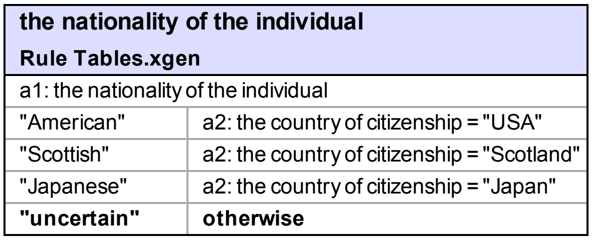
1. Replace the text **condition** in the second column with "the country of citizenship". This cell is already in the correct **Condition Heading** style. As we will only be having one set of conditions you can delete the first **condition** column.
2. Replace the text **conclusion** with "the nationality 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
else	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**.)

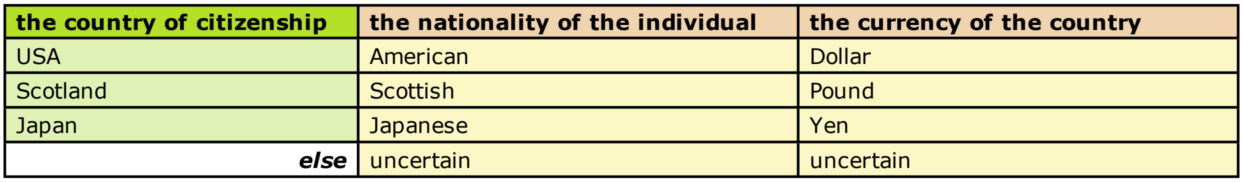


the nationality of the individual **Rule Tables.xgen**	
a1: the nationality of the individual	
"American"	a2: the country of citizenship = "USA"
"Scottish"	a2: the country of citizenship = "Scotland"
"Japanese"	a2: the country of citizenship = "Japan"
"uncertain"	**otherwise**

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:

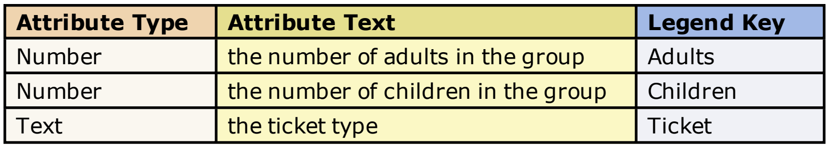


the country of citizenship	the nationality of the individual	the currency of the country
USA	American	Dollar
Scotland	Scottish	Pound
Japan	Japanese	Yen
else	uncertain	uncertain

Prove the same set of conclusions using multiple conditions

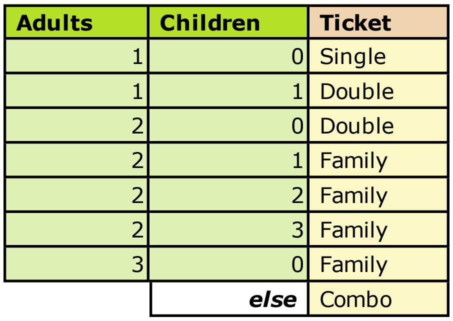
You can specify multiple conditions for a particular conclusion in Excel, merging the conclusion cells if appropriate to influence the way the rule is evaluated.

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



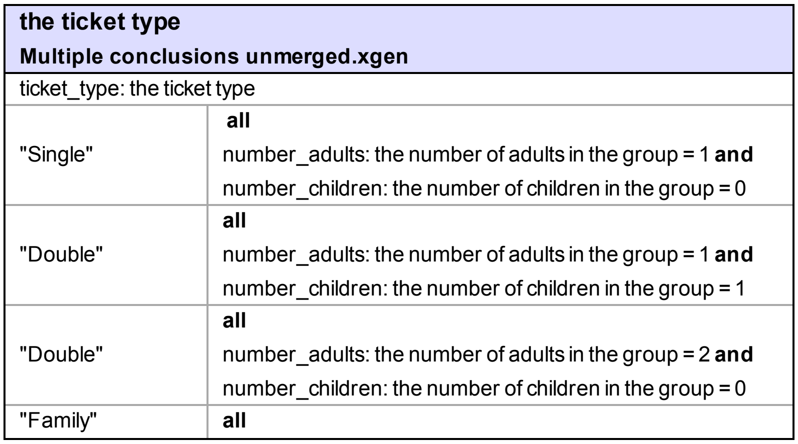
Attribute Type	Attribute Text	Legend Key
Number	the number of adults in the group	Adults
Number	the number of children in the group	Children
Text	the ticket type	Ticket

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
	else	Combo

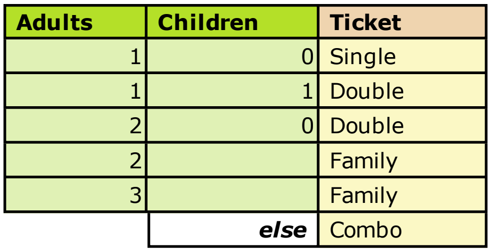
The rule generated for this table in Oracle Policy Modeling will look like the following:



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

the ticket type **Multiple conclusions unmerged.xgen**	
	number_adults: the number of adults in the group = 2 **and** number_children: the number of children in the group = 1
"Family"	**all** number_adults: the number of adults in the group = 2 **and** number_children: the number of children in the group = 2
"Family"	**all** number_adults: the number of adults in the group = 2 **and** number_children: the number of children in the group = 3
"Family"	**all** number_adults: the number of adults in the group = 3 **and** number_children: the number of children in the group = 0
"Combo"	**otherwise**

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.

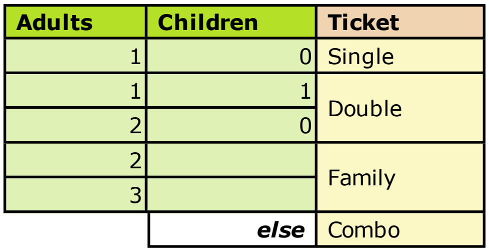


Adults	Children	Ticket
1	0	Single
1	1	Double
2	0	Double
2		Family
3		Family
	else	Combo

This will simplify the logic, and the rule generated:

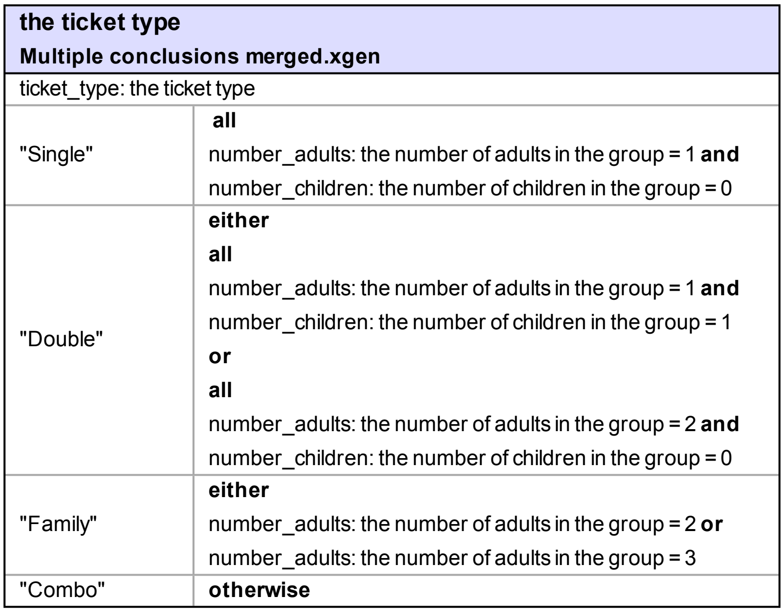
the ticket type **Multiple conclusions simplified.xgen**	
ticket_type: the ticket type	
"Single"	**all** number_adults: the number of adults in the group = 1 **and** number_children: the number of children in the group = 0
"Double"	**all** number_adults: the number of adults in the group = 1 **and** number_children: the number of children in the group = 1
"Double"	**all** number_adults: the number of adults in the group = 2 **and** 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"	**otherwise**

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		
	else	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"	**all** number_adults: the number of adults in the group = 1 **and** number_children: the number of children in the group = 0
"Double"	**either** **all** number_adults: the number of adults in the group = 1 **and** number_children: the number of children in the group = 1 **or** **all** number_adults: the number of adults in the group = 2 **and** number_children: the number of children in the group = 0
"Family"	**either** number_adults: the number of adults in the group = 2 **or** number_adults: the number of adults in the group = 3
"Combo"	**otherwise**

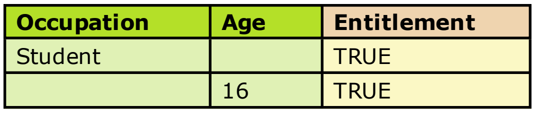
TIP: To see an example of a complete rulebase with merged condition and conclusion cells, open and run the Insurance Fraud Score example rulebase project provided in the Examples folder in the Oracle Policy Modeling installation folder.

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.

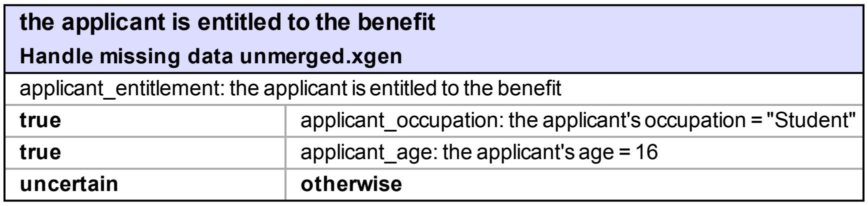
In some cases, this may not be the most useful way for the rule to evaluate. If a single conclusion is proved in multiple ways, you can merge a single conclusion cell across all of the different condition rows. Oracle Policy Modeling will then allow any of those condition rows to prove the conclusion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



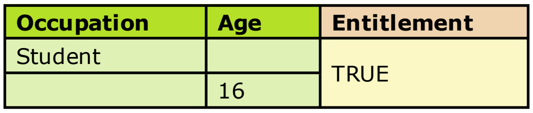
Occupation	**Age**	**Entitlement**
Student		TRUE
	16	TRUE

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.



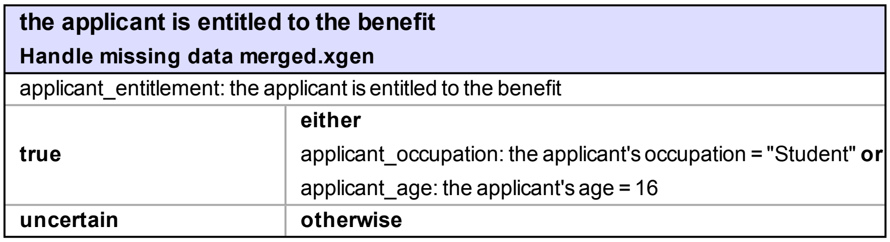
<table>
<tr><th colspan="2">the applicant is entitled to the benefit
Handle missing data unmerged.xgen</th></tr>
<tr><td colspan="2">applicant_entitlement: the applicant is entitled to the benefit</td></tr>
<tr><td>true</td><td>applicant_occupation: the applicant's occupation = "Student"</td></tr>
<tr><td>true</td><td>applicant_age: the applicant's age = 16</td></tr>
<tr><td>uncertain</td><td>otherwise</td></tr>
</table>

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.



<table>
<tr><th>Occupation</th><th>Age</th><th>Entitlement</th></tr>
<tr><td>Student</td><td></td><td rowspan="2">TRUE</td></tr>
<tr><td></td><td>16</td></tr>
</table>

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.



<table>
<tr><th colspan="2">the applicant is entitled to the benefit
Handle missing data merged.xgen</th></tr>
<tr><td colspan="2">applicant_entitlement: the applicant is entitled to the benefit</td></tr>
<tr><td>true</td><td>either
applicant_occupation: the applicant's occupation = "Student" or
applicant_age: the applicant's age = 16</td></tr>
<tr><td>uncertain</td><td>otherwise</td></tr>
</table>

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
			else	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
		else	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
else	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
else	14

2005-2006

Ticket	Price
Adult	16
Concession	12
Child	8
else	16

2006-2007

Ticket	Price
Adult	20
Concession	15
Child	10
else	20

This will create the following rule in Oracle Policy Modeling:

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

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	
	else	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
else	$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)
else	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
else	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
	else	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
	else	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.**

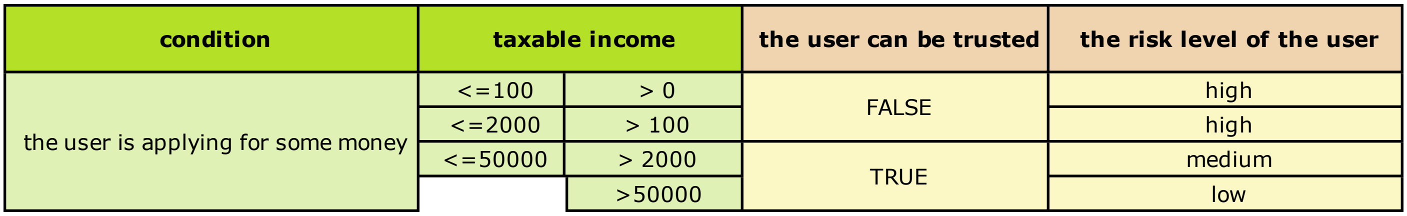
Adults	Children	Ticket
1	0	Single
	1	Double
2	0	Double
		Family
3		Family
	else	Combo

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.

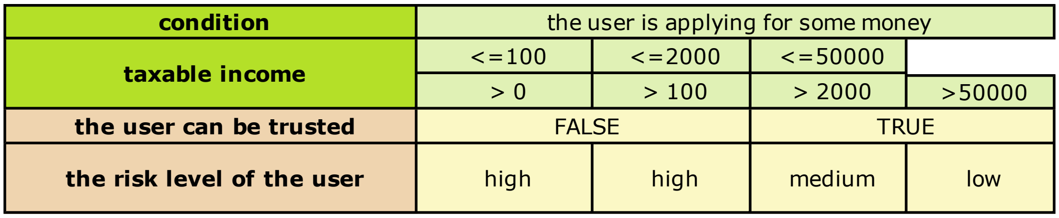
Change rule table orientation

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

the claimant lives in Australia	
true	the claimant lives in Sydney
true	the claimant lives in Canberra
false	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 that 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 coexist. (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
Numerical functions	Used with number and currency variables to perform basic and complex arithmetic calculations, trigonometric calculations and maximum/minimum calculations
Date functions	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
Time of day functions	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 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

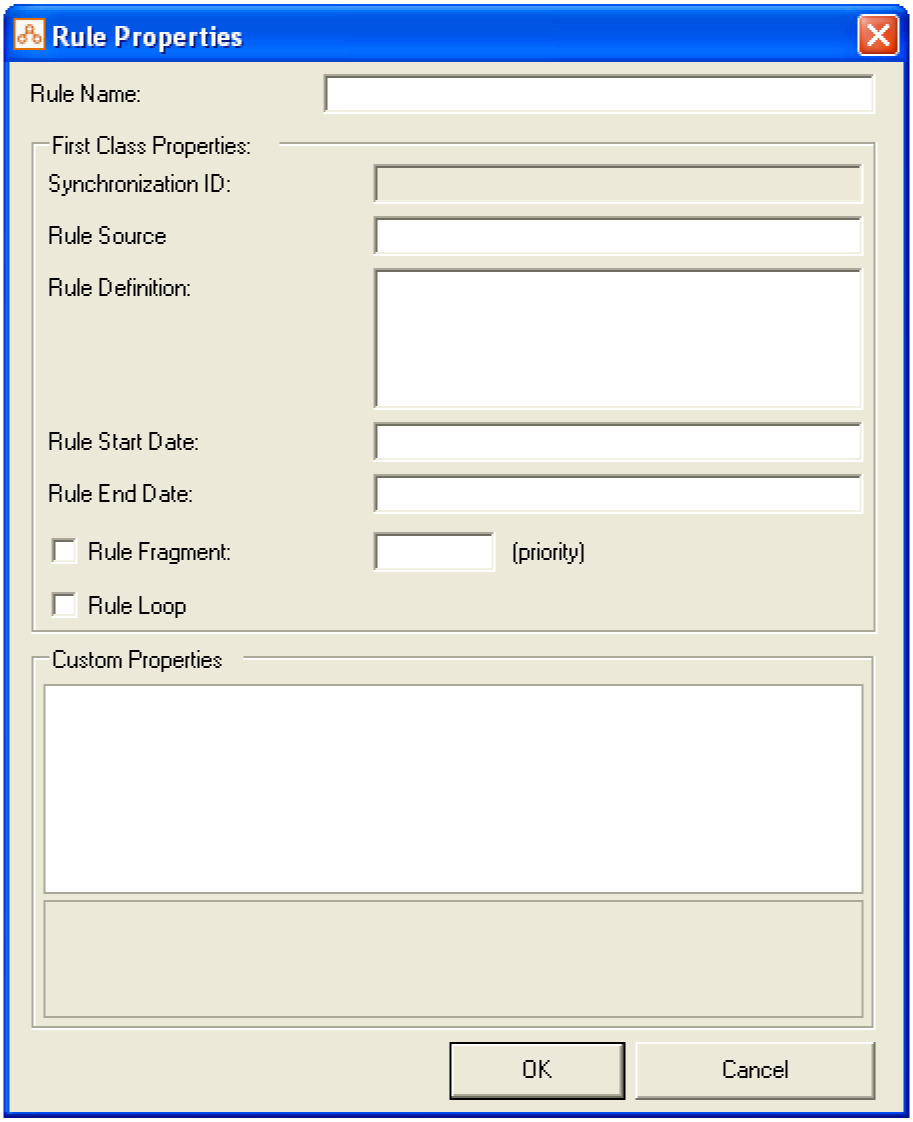
The following rule metadata can be added to a rule:

- 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.



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

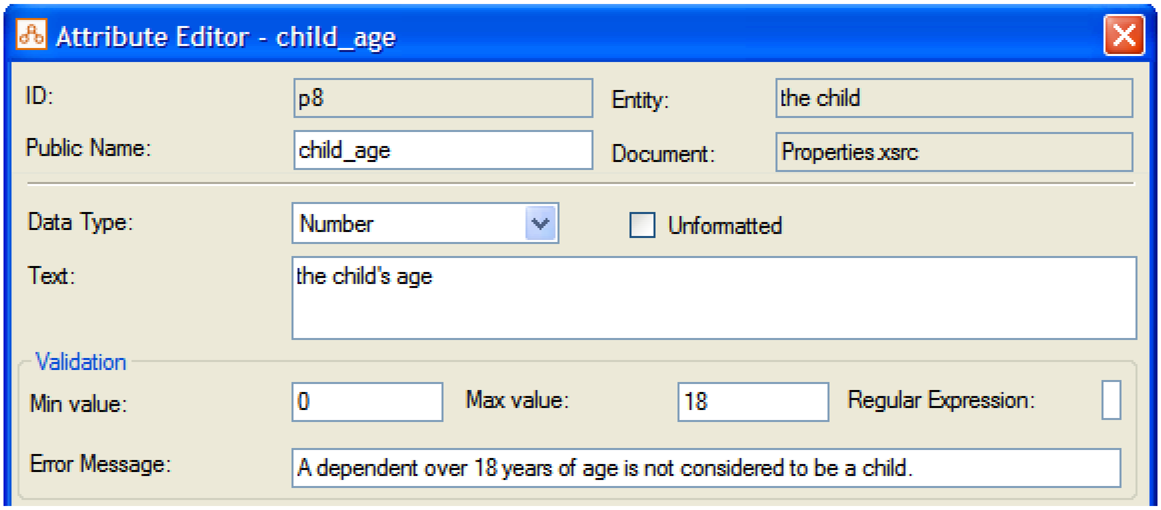
Minimum and maximum values can be specified for number, currency, date, time of day, and date/time variables to ensure that data entered by the user falls within a certain range. Values must be specified in the correct format.

- 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").



Attribute Editor - child_age

ID: p8

Entity: the child

Public Name: child_age

Document: Properties.xsrc

Data Type: Number

☐ Unformatted

Text: the child's age

Validation

Min value: 0

Max value: 18

Regular Expression:

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

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	^[\w-]+(?:\.[\w-]+)*@(?:[\w-]+\.)+[a-zA-Z]{2,7}$
To check the data entered follows a certain format: 2 letters 6 numbers 1 letter (eg AB123456C)	^[A-Za-z]{2}[0-9]{6}[A-Za-z]$
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 spaces separating the sections.	^(\(0[0-9])\) ?[1-9][0-9]{3} ?[0-9]{4}$
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.	^\s*(\S+(\s+\S+){0,N}\s*)?$
To check a National Insurance Number (NINO)	^[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

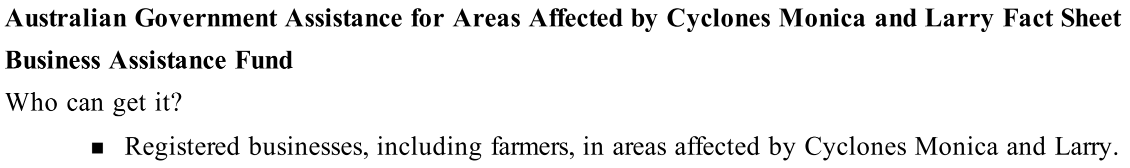
Once the scope of the rulebase has been determined and documented, you need to create rules documents for the sections of the material that are in scope. One of the primary activities in interpreting source material is to identify the logical foundation blocks in that material – the logical operators 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:

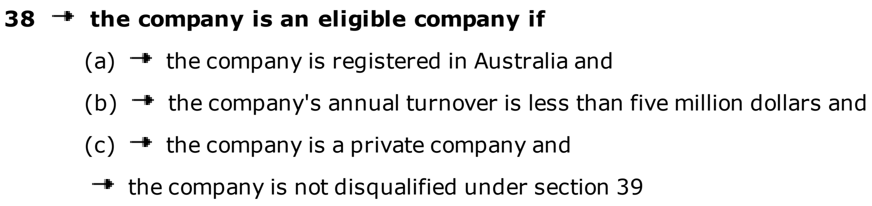
```
Replace[<text to be replaced>, <replacement text including structural elements ^x and ^entity>]
```

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 to 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

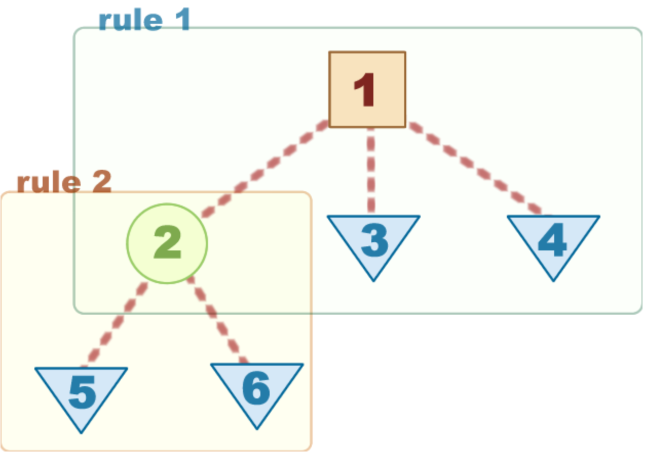
Rule Element	Comment
1: The person is eligible for a licence if:	This is the goal proposition or conclusion of the rule.
2: The person is qualified to drive a car	This is one of the rule conditions but is also the conclusion for Rule 2 (below).

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**.

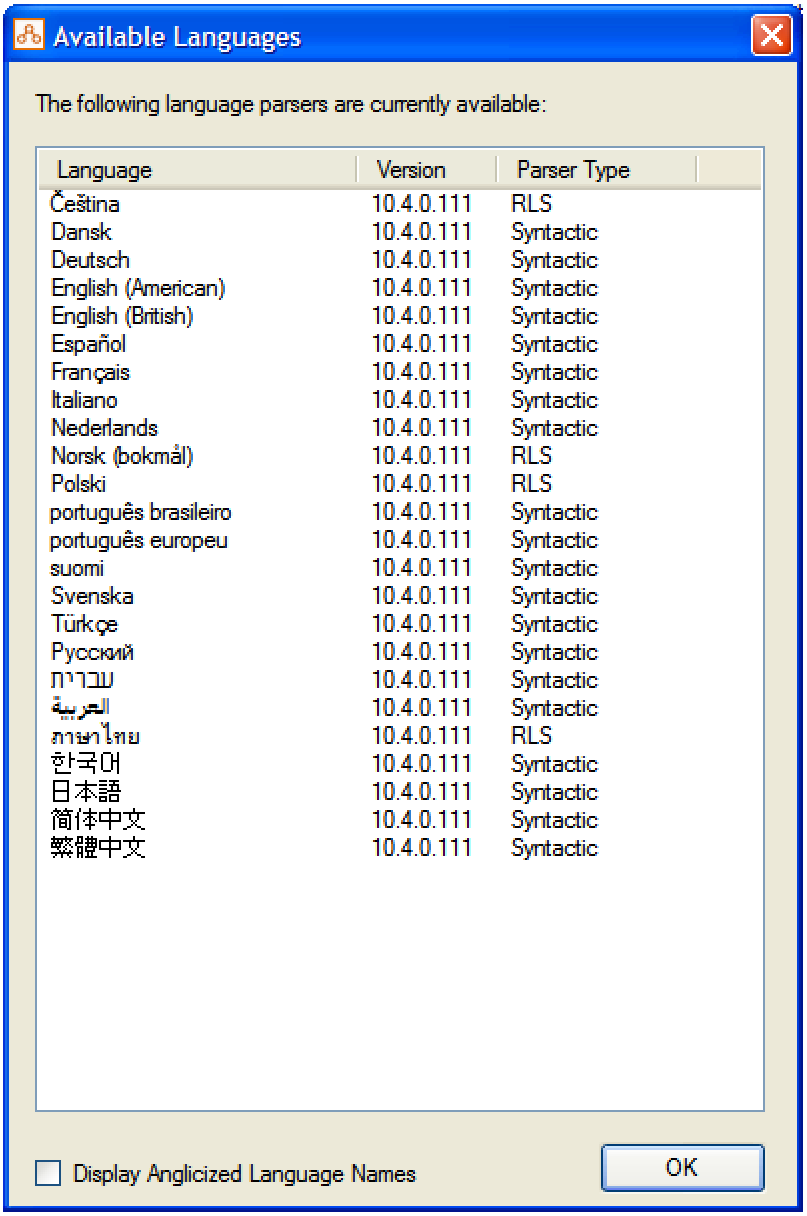
Once you have created your project and commenced rulebase development (ie once rules or attributes have been created), the rule language and region are locked and you will not be able to change these settings.

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 following language parsers are currently available:

Language	Version	Parser Type
Čeština	10.4.0.111	RLS
Dansk	10.4.0.111	Syntactic
Deutsch	10.4.0.111	Syntactic
English (American)	10.4.0.111	Syntactic
English (British)	10.4.0.111	Syntactic
Español	10.4.0.111	Syntactic
Français	10.4.0.111	Syntactic
Italiano	10.4.0.111	Syntactic
Nederlands	10.4.0.111	Syntactic
Norsk (bokmål)	10.4.0.111	RLS
Polski	10.4.0.111	RLS
português brasileiro	10.4.0.111	Syntactic
português europeu	10.4.0.111	Syntactic
suomi	10.4.0.111	Syntactic
Svenska	10.4.0.111	Syntactic
Türkçe	10.4.0.111	Syntactic
Русский	10.4.0.111	Syntactic
עברית	10.4.0.111	Syntactic
العربية	10.4.0.111	Syntactic
ภาษาไทย	10.4.0.111	RLS
한국어	10.4.0.111	Syntactic
日本語	10.4.0.111	Syntactic
简体中文	10.4.0.111	Syntactic
繁體中文	10.4.0.111	Syntactic

☐ Display Anglicized Language Names

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

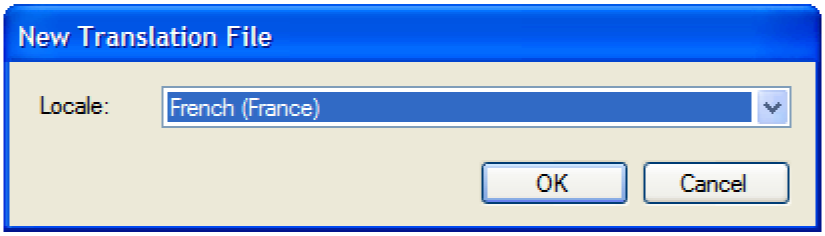
Check for untranslated text in a rulebase

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.

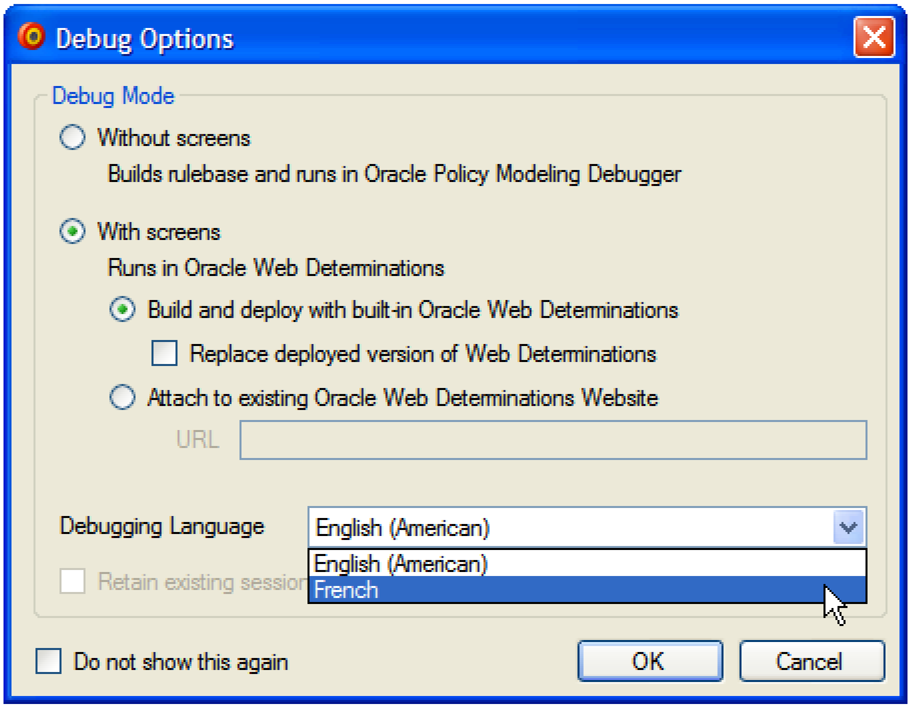
1. Right-click on the **Translations** folder in the Project Explorer of your rulebase project, and select **Add New Translation Document**.
2. In the **New Translation File** dialog, select the appropriate **Locale** for your new rulebase translation, based on the relevant language and region, and click **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.


New Translation File

Locale: French (France)

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.



Debug Options

Debug Mode

○ Without screens

Builds rulebase and runs in Oracle Policy Modeling Debugger

● With screens

Runs in Oracle Web Determinations

● Build and deploy with built-in Oracle Web Determinations

☐ Replace deployed version of Web Determinations

○ Attach to existing Oracle Web Determinations Website

URL

Debugging Language: English (American)

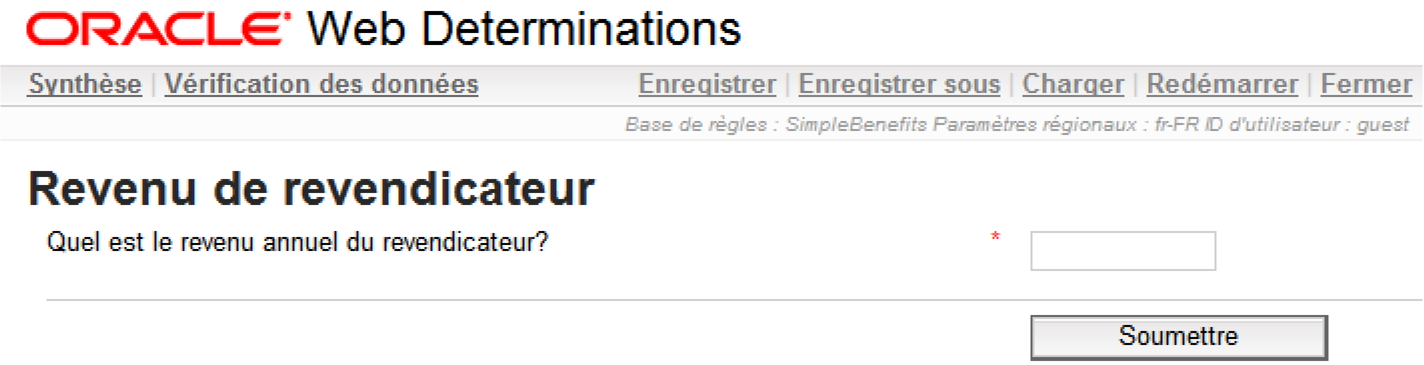
English (American)
French

☐ Retain existing session

☐ Do not show this again

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

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

Revenu de revendicateur

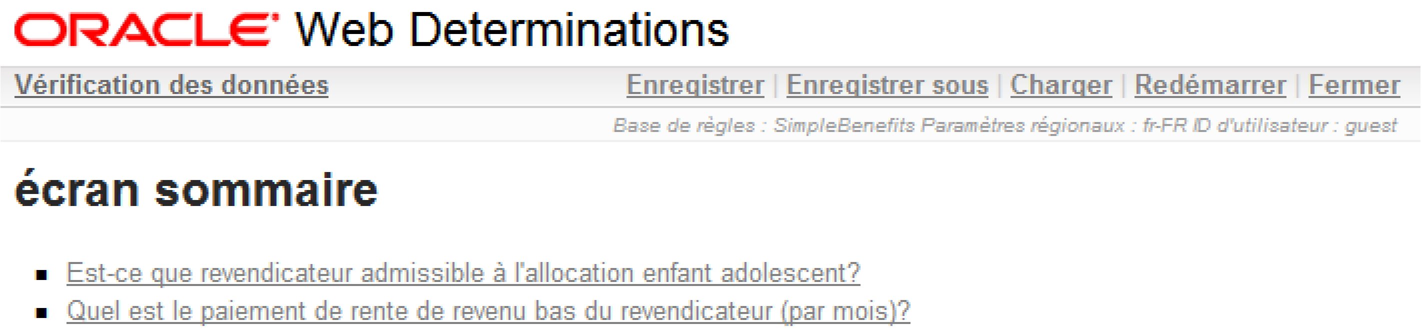
Quel est le revenu annuel du revendicateur? *

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.



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.

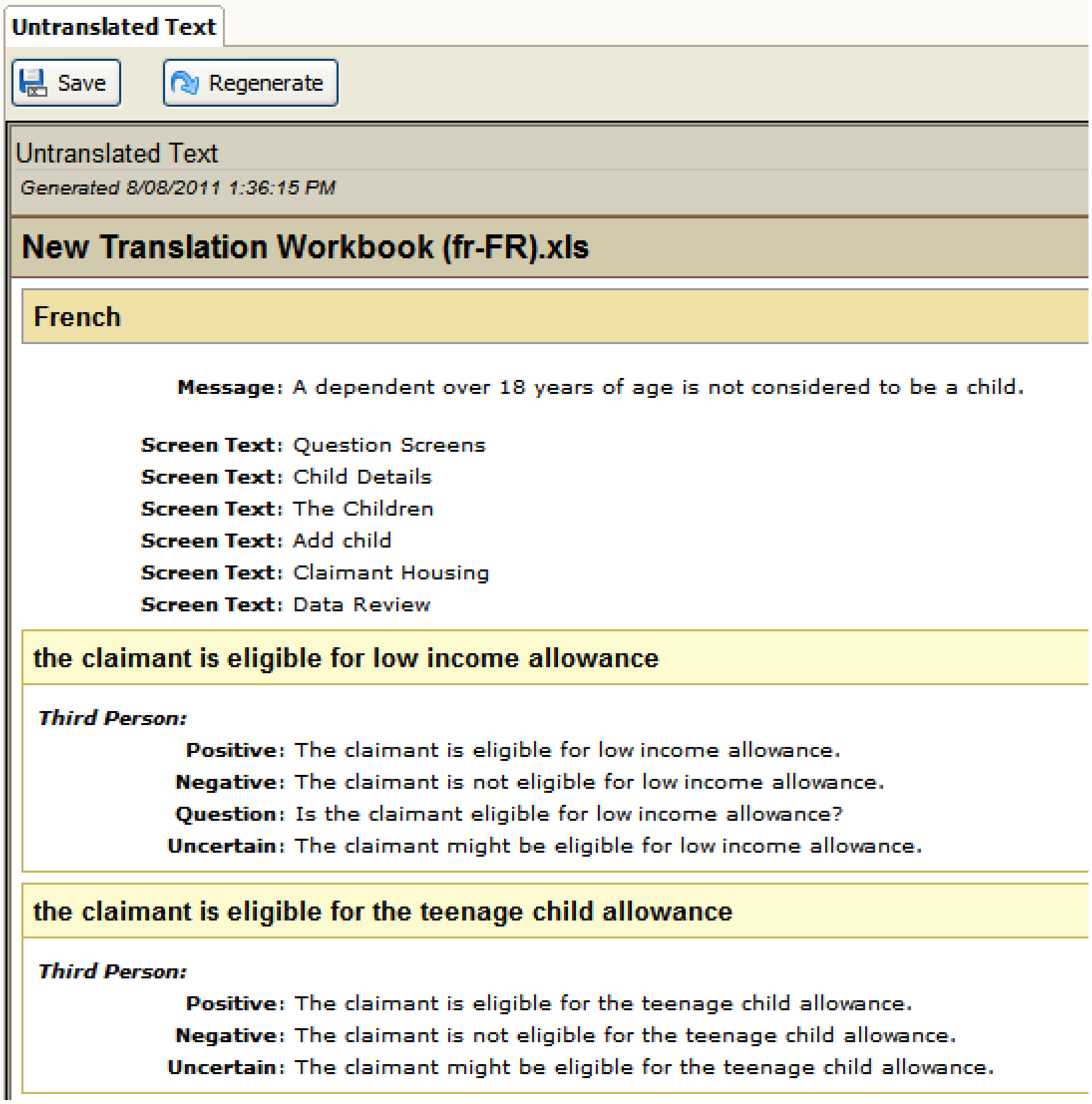
You will need to create a configuration file to run the translation you selected in Oracle Web Determinations. To do this:

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

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.

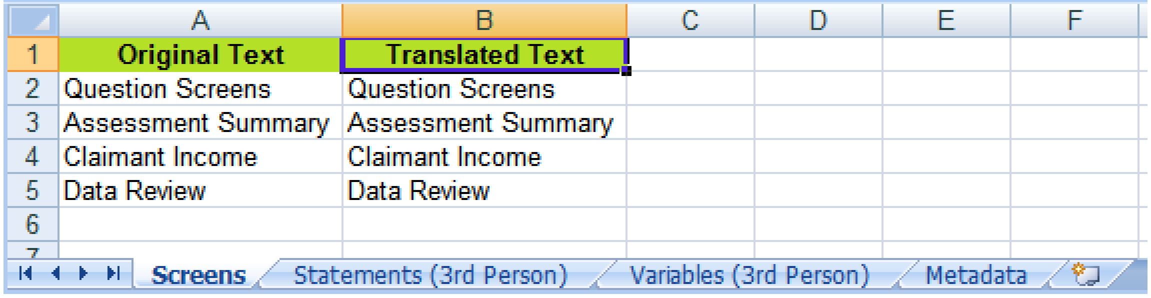
3. Use this report to complete any missing translations, then click **Regenerate** at the top of the report to verify that all translations have been completed.
4. The **Error List** will also show a warning if your rulebase has missing translations. Click on **View | Error List** to display this. Double-click 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	**Original Text**	**Translated Text**				
2	Question Screens	Question Screens				
3	Assessment Summary	Assessment Summary				
4	Claimant Income	Claimant Income				
5	Data Review	Data Review				
6						

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 rulebase 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).

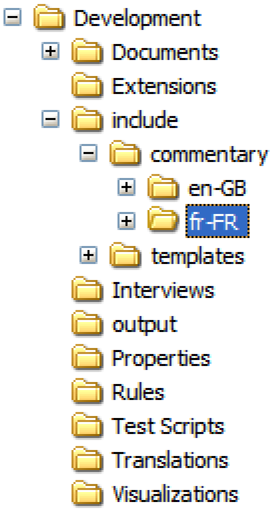
See also:

- 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.

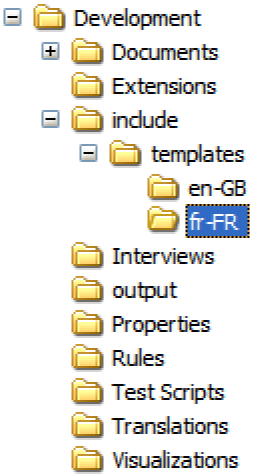
See also:

- 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:

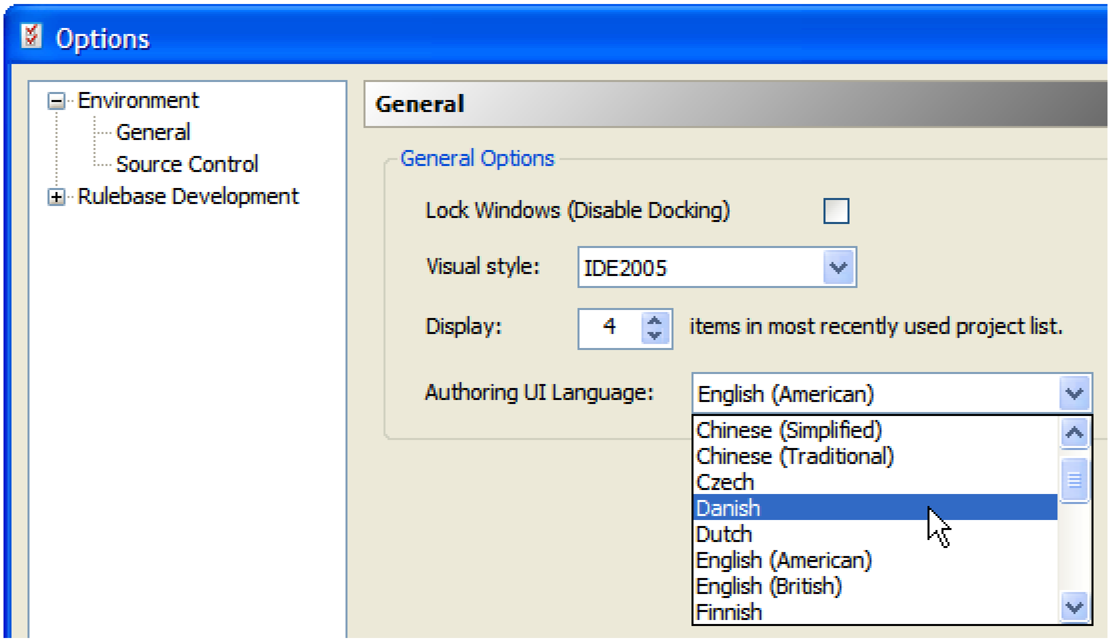
- Create, update or delete an interview document
- Develop a template for an interview document

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.



Lock Windows (Disable Docking) ☐

Visual style: IDE2005

Display: 4 items in most recently used project list.

Authoring UI Language: English (American)

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[1] 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

---

[1] The conjugations are the different forms the verbs take depending on what person and tense they are.

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**.

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 [1]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] 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.

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.

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[1] (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:

---

[1] The symbol marking the point between the whole and decimal parts of a number.

- 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
- 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. استخدمand اكل). The default verbs list has explicit Hamzated versions of these verbs with Hamza above Alef (e.g. أستخدم and أكل), 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 arthrite, 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 ("אינך", "אינך", "אינו", "אינה", "אינכם", "אינכן", "אינם", "אינן") 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
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* (なかった) and *kawa nakatta* (買った/買わなかった) 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 rulebase. 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 (なければならない) 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 사과을 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 to한다).

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 rulebase. 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 сделает 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 *был**и** обязан**ы** 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?

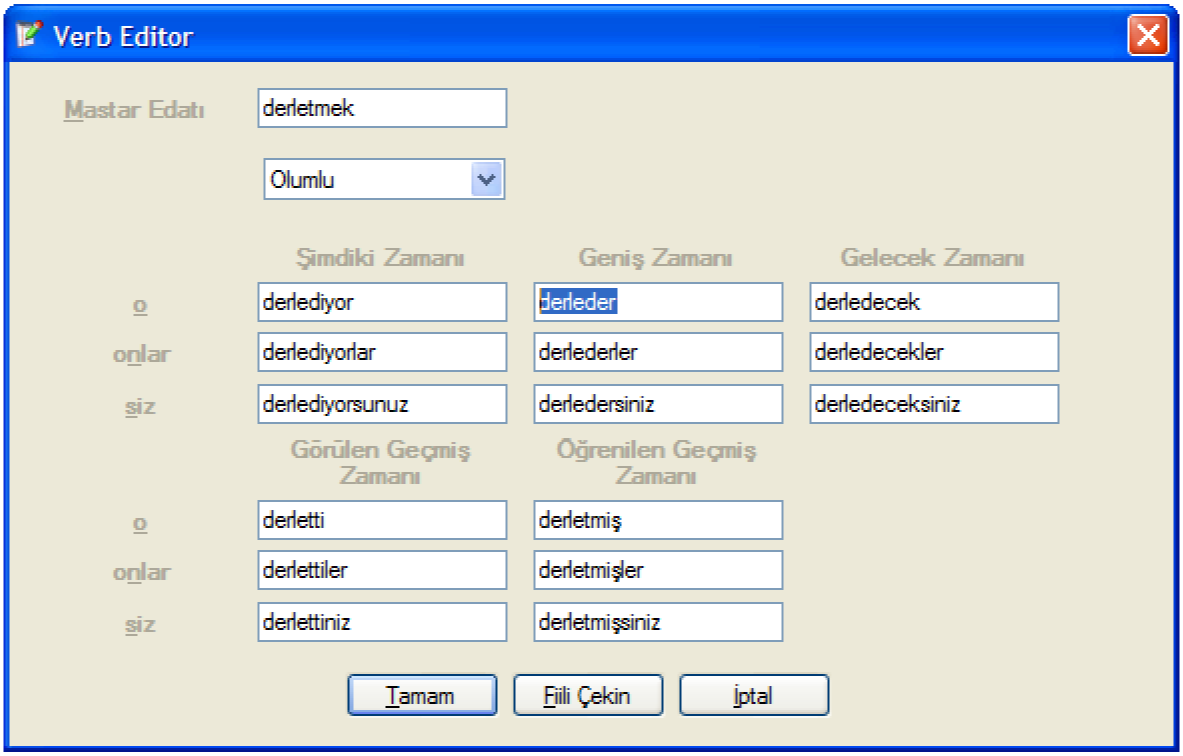
Currently, Oracle Policy Modeling will not perform the contraction.

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:



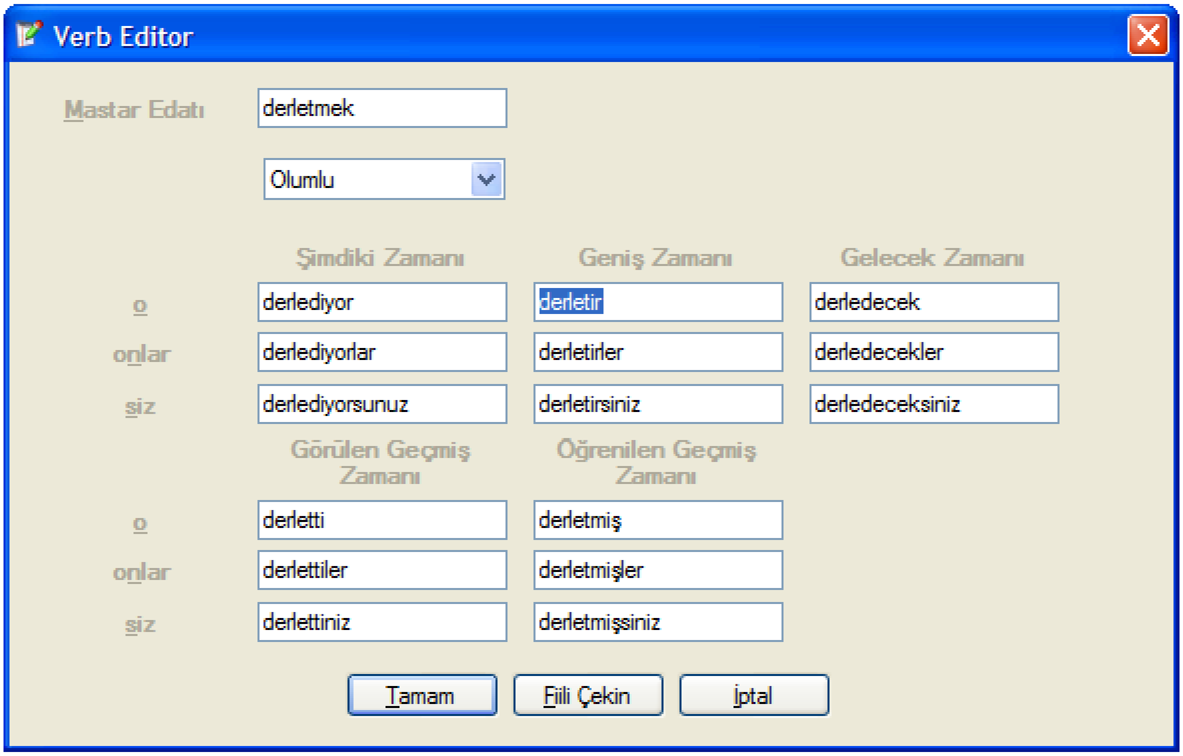
Mastar Edatı: derletmek

Olumlu

	Şimdiki Zamanı	Geniş Zamanı	Gelecek Zamanı
o	derlediyor	derleder	derledecek
onlar	derlediyorlar	derlederler	derledecekler
siz	derlediyorsunuz	derledersiniz	derledeceksiniz

	Görülen Geçmiş Zamanı	Öğrenilen Geçmiş Zamanı
o	derletti	derletmiş
onlar	derlettiler	derletmişler
siz	derlettiniz	derletmişsiniz

You can update the 3rd person singular Present Aorist positive form with the appropriate verb conjugation:



Mastar Edatı: derletmek

Olumlu

	Şimdiki Zamanı	Geniş Zamanı	Gelecek Zamanı
o	derlediyor	derletir	derledecek
onlar	derlediyorlar	derletirler	derledecekler
siz	derlediyorsunuz	derletirsiniz	derledeceksiniz

	Görülen Geçmiş Zamanı	Öğrenilen Geçmiş Zamanı
o	derletti	derletmiş
onlar	derlettiler	derletmişler
siz	derlettiniz	derletmişsiniz

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

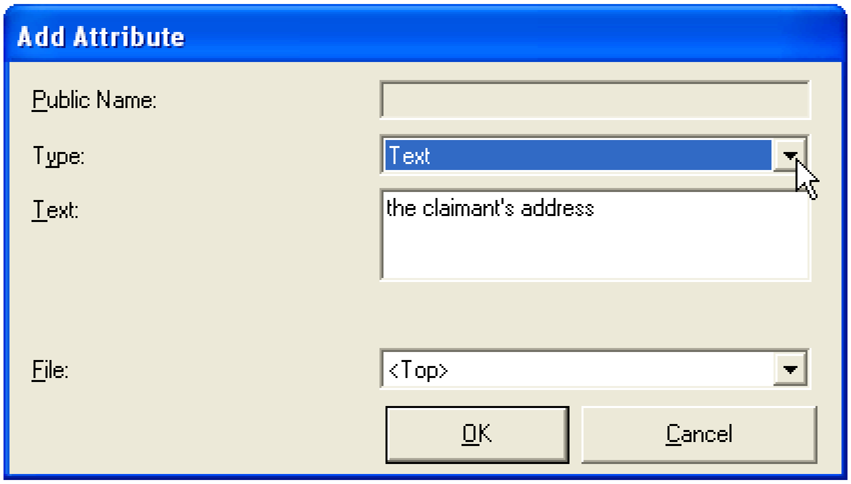
Create a new attribute in a properties file

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.



Type: Text

Text: the claimant's address

File: <Top>

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:

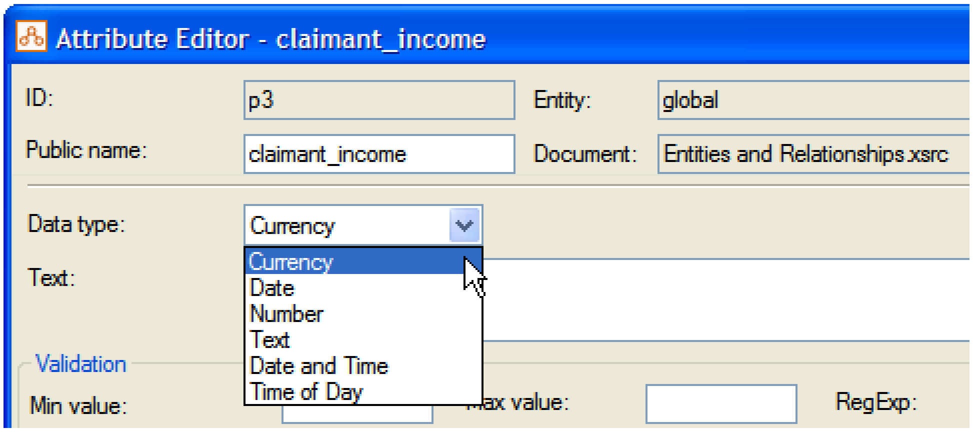
1. Open the **Declarations** worksheet.
2. In the **Attribute Type** column enter the type of attribute, and in the **Attribute Text** column enter the text of the attribute.

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.

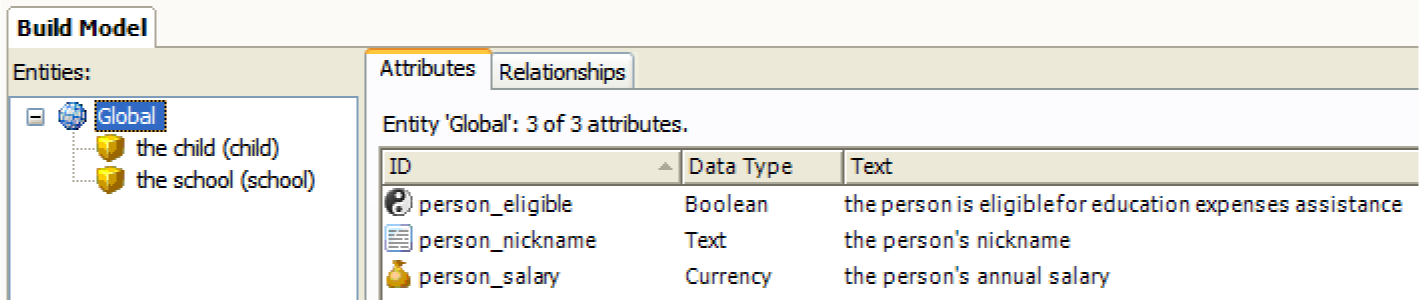
For example, assume the attributes in the following table have been added to a Word document where "the household member" has been defined as an entity in the rulebase:

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.

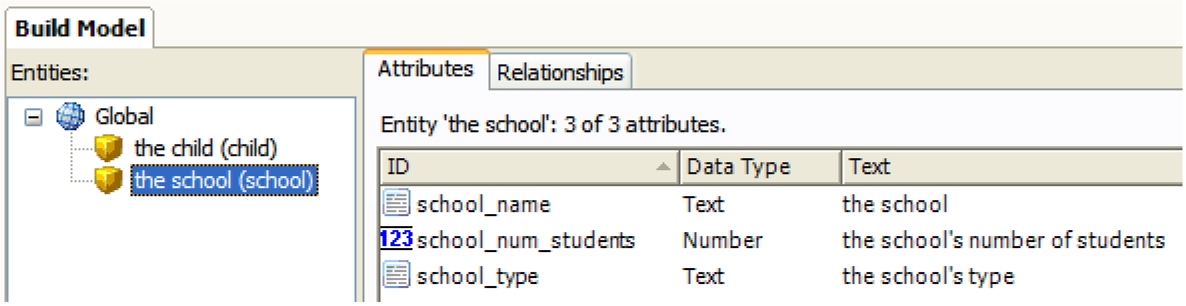
Attributes which do not contain any entity text are placed in the **Global** level. The list of global attributes are displayed in the right-hand pane:



Entity 'Global': 3 of 3 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:



Entity 'the school': 3 of 3 attributes.

ID	Data Type	Text
school_name	Text	the school
school_num_students	Number	the school's number of students
school_type	Text	the school's type

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 source entity text so that it is clear from the relationship name who the relationship is from.

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

Attribute type	Icon	When used	Example
Number	123	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
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 (>)	**the person is over 18 if** the person's age > 18
Less than (<)	**the employee is early for work if** the time the employee starts work < the specified start time for the employee
Equals (=)	**the person was born on the same day as the person's spouse if** the person's date of birth = the person's spouse's date of birth
Not equal to (<>)	**the pet is not a monkey if** the type of pet <> "monkey"
Greater than or equal to (>=)	**the applicant is eligible for a loan if** the applicant's annual income >= 50000
Less than or equal to (<=)	**the submission is valid if** 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

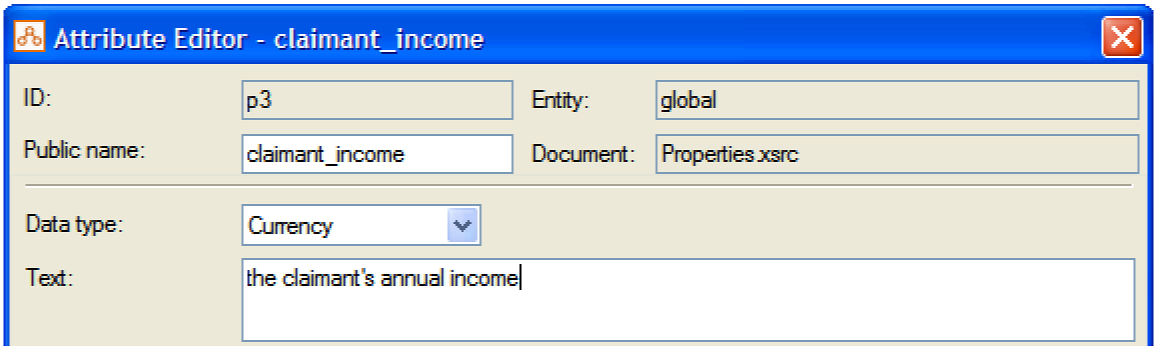
Take, for example, the following source material:

> 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".



Attribute Editor - claimant_income

ID: p3

Entity: global

Public name: claimant_income

Document: Properties.xsrc

Data type: Currency

Text: 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.	**the number of ants on the property = 15,000**	Treatment of commas ",", periods "." and spaces " " as decimal and thousand separators are based on the Region settings for the rulebase project. Scientific notation may not be used. There are also considerations in formatting of numeric values in functions.
Currency	Any number (supports decimals)	Approximately 30 significant figures may be used for	**the person's savings (in dollars) = 534.50**	Leading $ and £ symbols may be used if this enhances read-

Variable type	Format	Range	Example rule	Notes
		very small or very large numbers, otherwise approximately 15 can be used.		ability of your rules, however, note that the formatting of attributes values will be determined by the rulebase region setting when the rulebase is run.
Text	Any string of alphanumeric characters may be used, enclosed by double quotes ".	The limit to the length of a text string depends on Word/Excel and your system, and should not be a practical limitation in your rule authoring.	**the household's location = "New York"**	To enter a 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	**the date of the last interest rate rise = 2007-10-25**	Oracle Web

Variable type	Format	Range	Example rule	Notes
				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 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-

Variable type	Format	Range	Example rule	Notes
				se 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	**the store closing time = 17:30:00**	
Date and time	yyyy-MM-dd hh:mm:ss	In the ranges detailed above for date and time values	**the submission date time = 2009-08-12 17:30:00**	Time zones are 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 custom function 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 >= 18 and
> >
> > the claimant's age <= 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 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 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 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:

the numerical form of the day of the week = (the date - 1980-01-07) 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
"Saturday"	the numerical form of day of the week = 5; or the numerical form of day of the week = -2
"Sunday"	the numerical form of day of the week = 6; or the numerical form of day of the week = -1
uncertain	**otherwise**

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

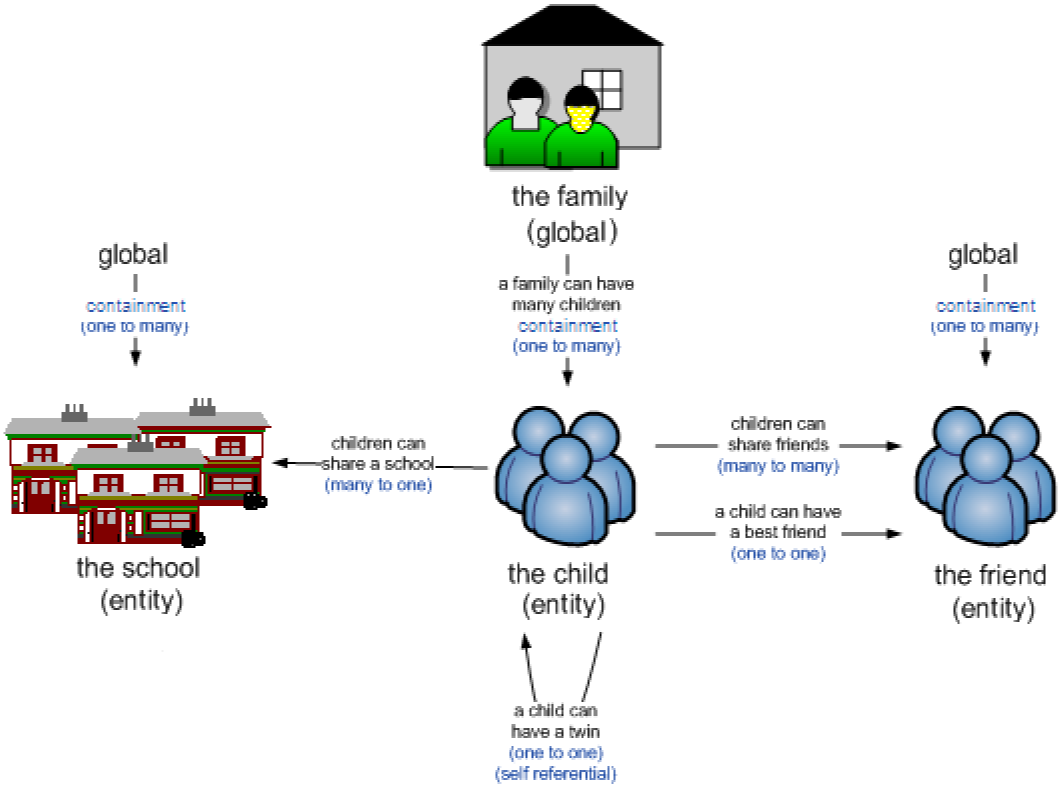
We can either model these last two relationships 'just in case' or not capture these relationships. We have left these relationships out of this example for simplicity.

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.

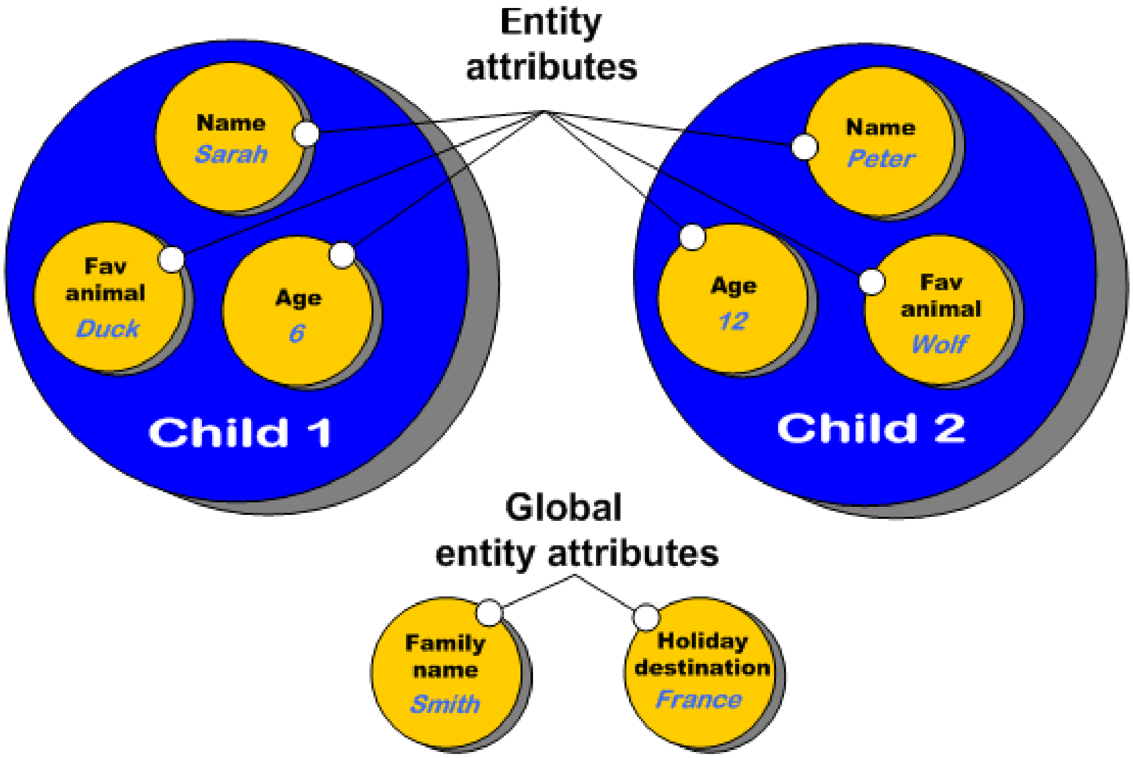
Global Entities

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:

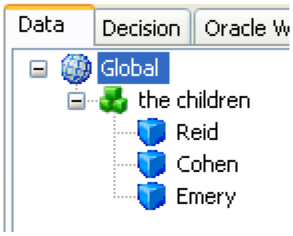
After you have defined an entity in this way, every attribute which uses the exact entity text (eg 'the child') becomes an entity-level attribute belonging to that entity.

When an entity is created:

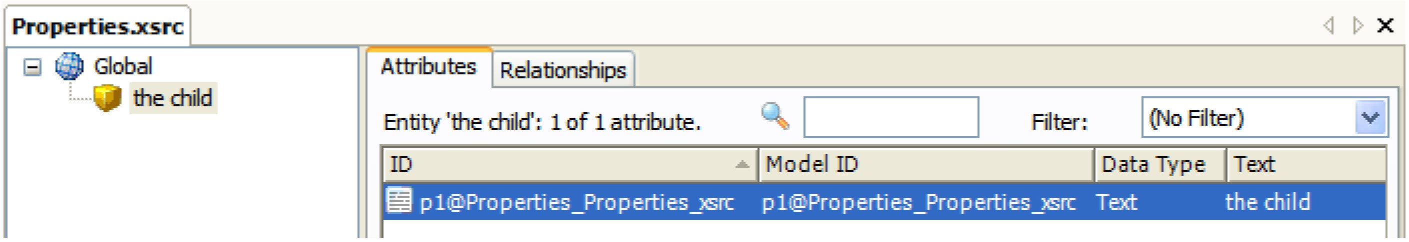
1. a containment relationship is automatically created for the new entity, to ensure that the containment structure in the rulebase data model is well-defined, and
2. an identifying attribute is automatically created for the new entity (see below).

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.



Entity 'the child': 1 of 1 attribute.

ID	Model ID	Data Type	Text
p1@Properties_Properties_xsrc	p1@Properties_Properties_xsrc	Text	the child

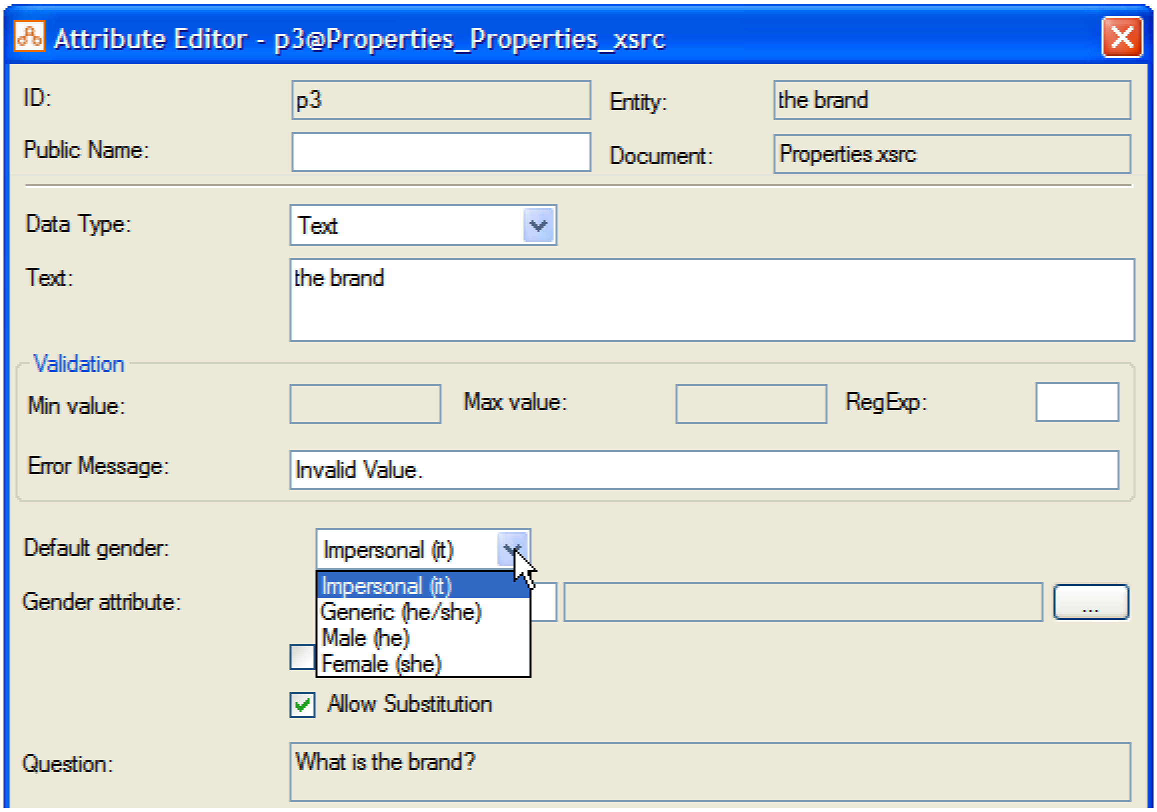
By default, this text attribute has:

- 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.



Attribute Editor - p3@Properties_Properties_xsrc

ID: p3

Entity: the brand

Public Name:

Document: Properties.xsrc

Data Type: Text

Text: the brand

Validation

Min value:

Max value:

RegExp:

Error Message: Invalid Value.

Default gender: Impersonal (it)

Gender attribute:

☑ Allow Substitution

Question: 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.

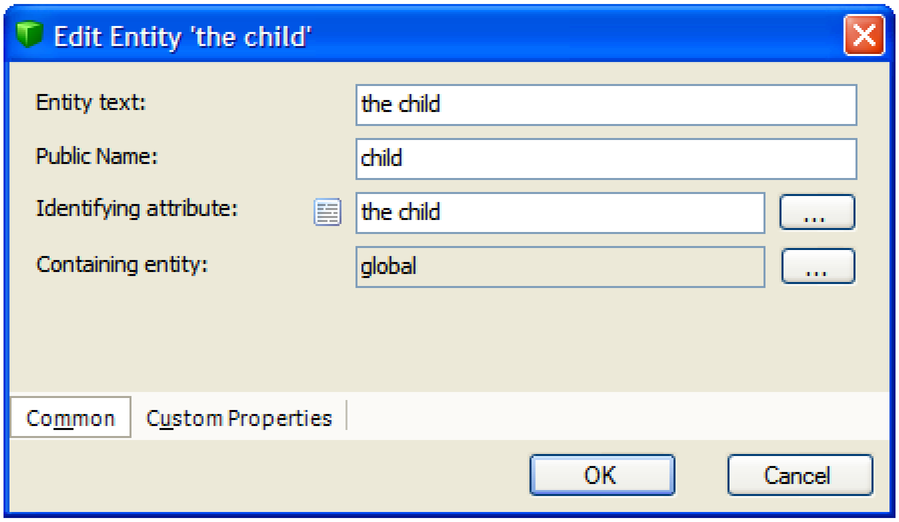
Give an entity a public name

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**.



Entity text: the child
Public Name: child
Identifying attribute: the child
Containing entity: global

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?

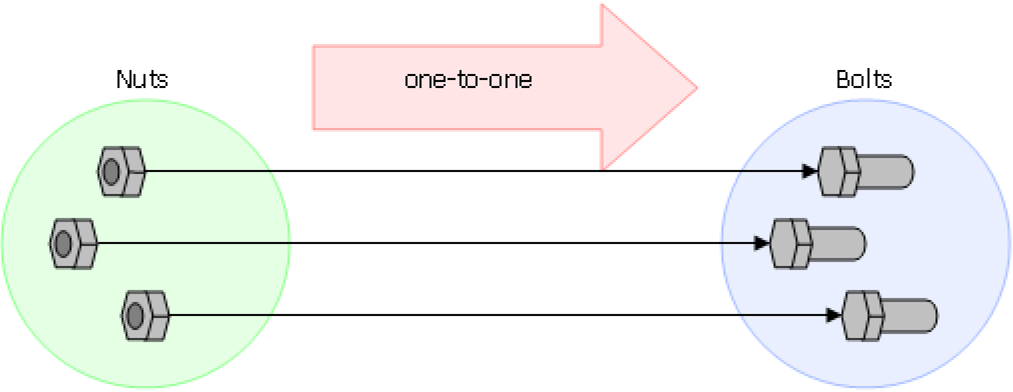
Understand the different types of relationships

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.



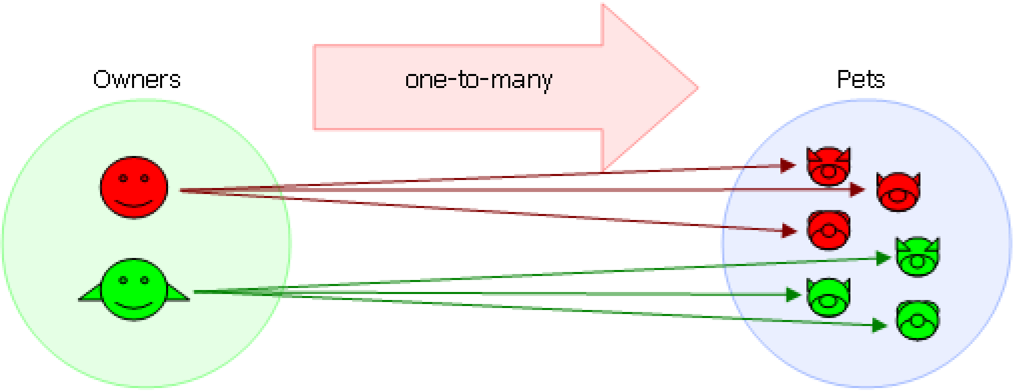
Entity 1: the nut

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.



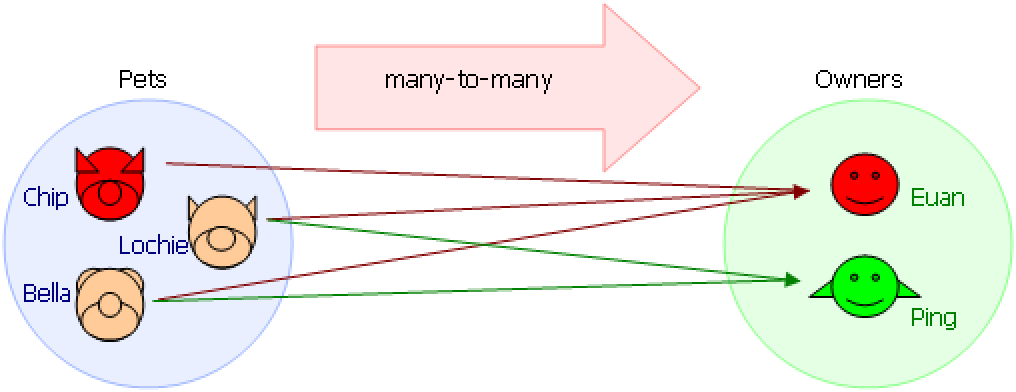
Entity 1: the owner

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

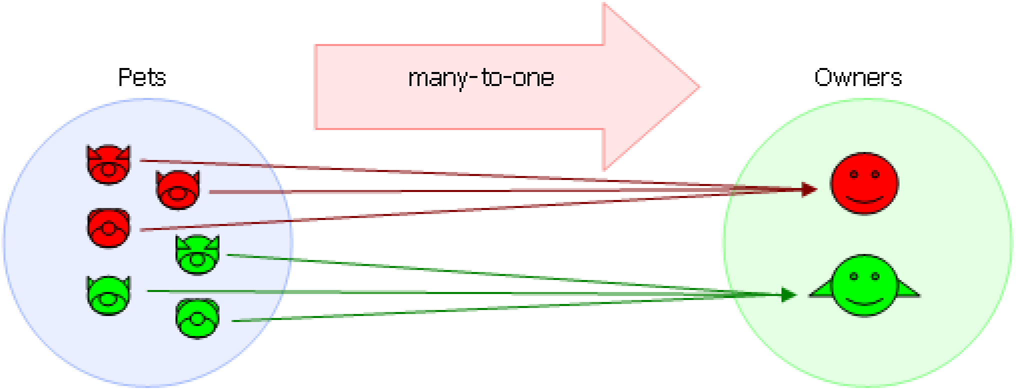
Entity 2: the owner

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

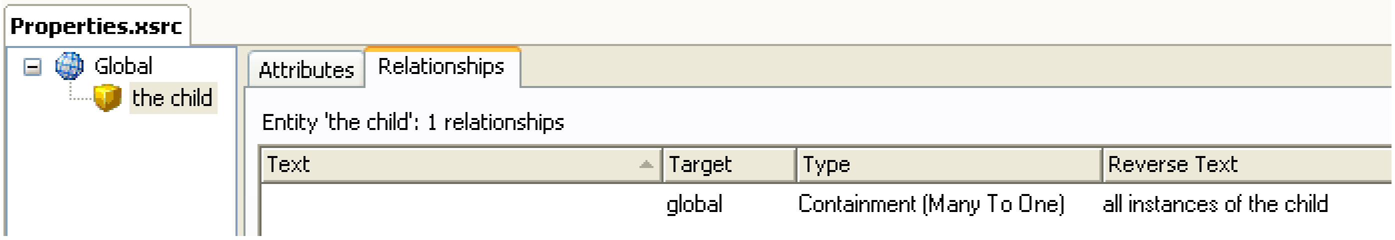
Relationships are defined in a properties file for the project, rather than in individual Word and Excel documents. This eliminates the need to re-define the same relationships in each rule file.

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:

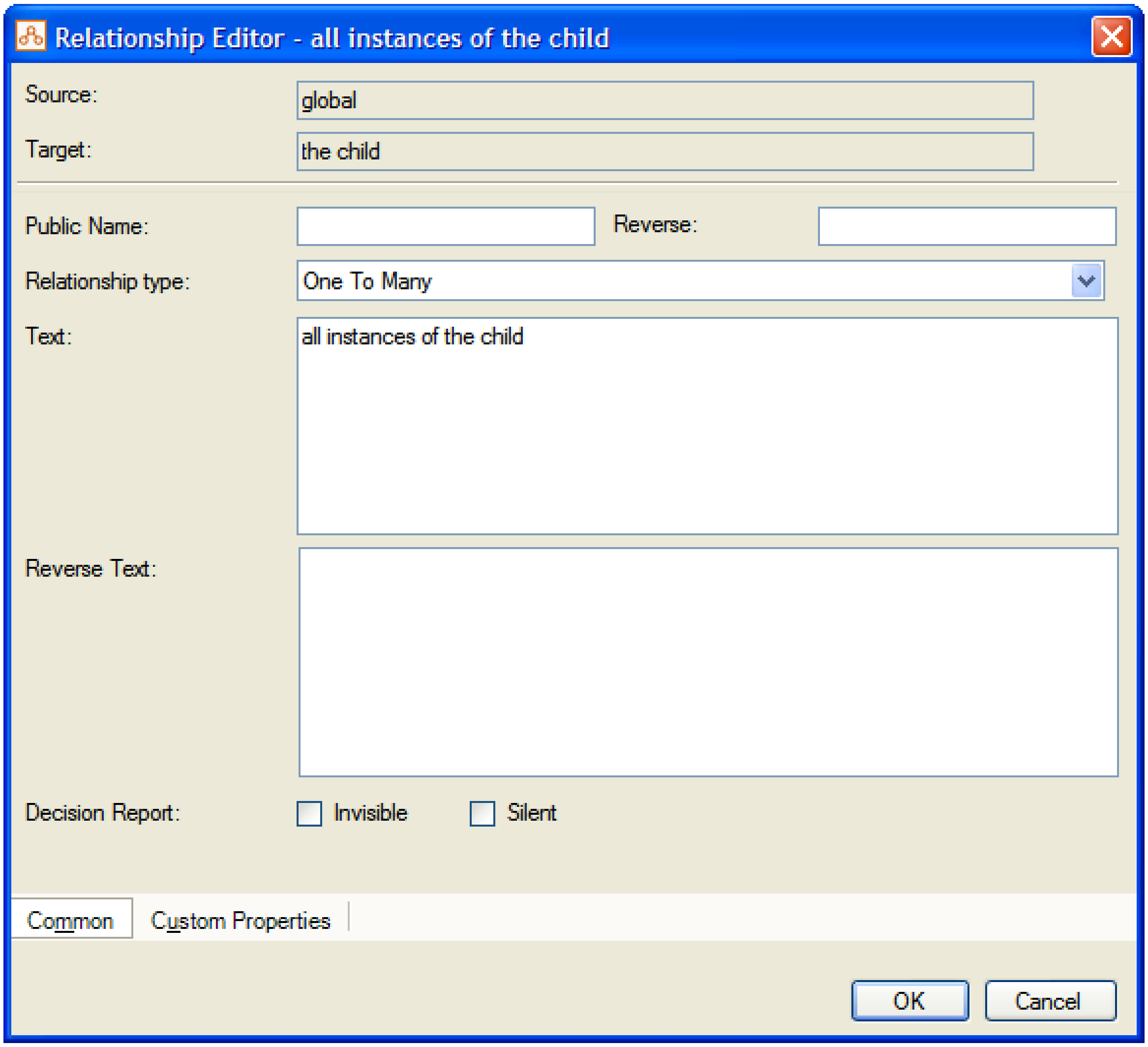
1. In Oracle Policy Modeling, double click your properties file to open it. Select the entity that you want to edit the containment relationship for. The containment relationships already created for the entity will be shown on the **Relationships** tab.

   Entity 'the child': 1 relationships

Text	Target	Type	Reverse Text
	global	Containment (Many To One)	all instances of the child

2. Double click on the relationship to open the **Relationship Editor** dialog box.



Source: global

Target: the child

Public Name:

Reverse:

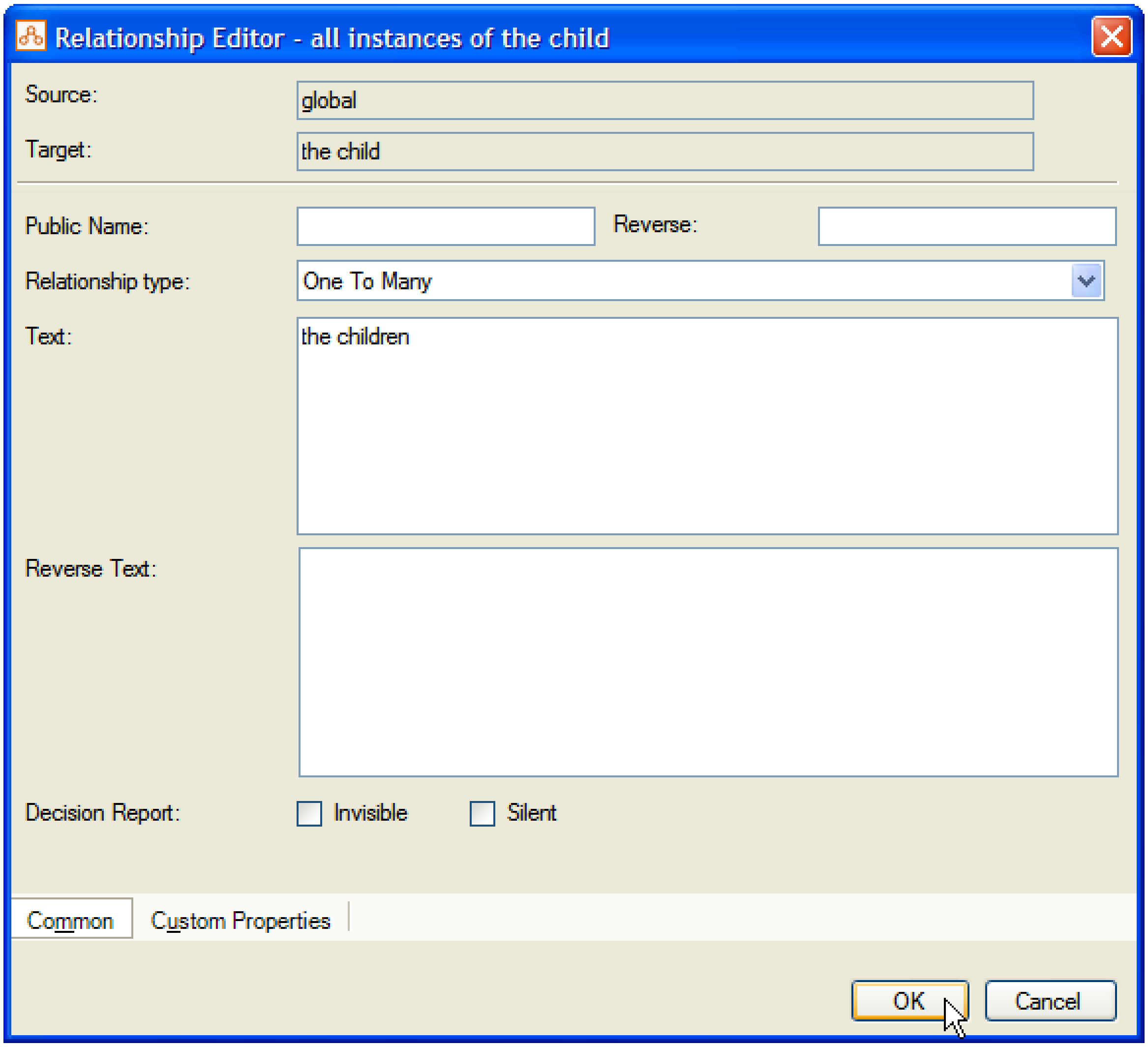
Relationship type: One To Many

Text: all instances of the child

Reverse Text:

Decision Report: ☐ Invisible ☐ Silent

3. Change the **Text** for the relationship.



Source: global

Target: the child

Public Name:

Reverse:

Relationship type: One To Many

Text: the children

Reverse Text:

Decision Report: ☐ Invisible ☐ Silent

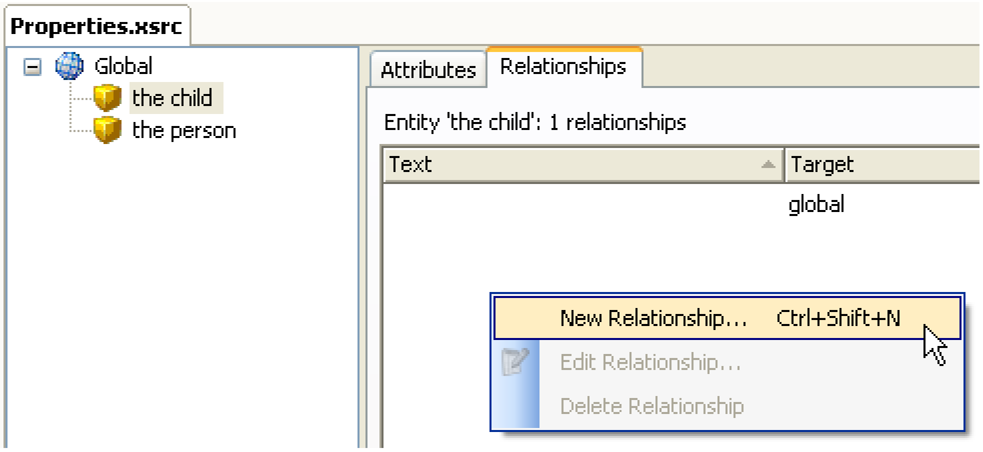
Click **OK**. You can now use your new relationship text in rules.

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.

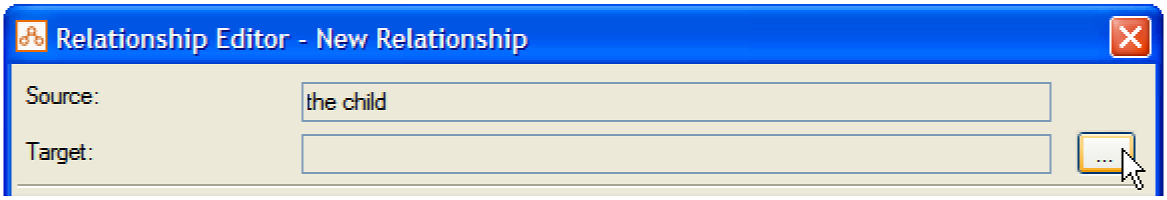
2. In the Relationships window, right-click and select **New Relationship...**



Entity 'the child': 1 relationships

Text	Target
	global

3. In the **Relationship Editor** dialog box, select the browse button next to the **Target** field to select the target entity.



Source: the child

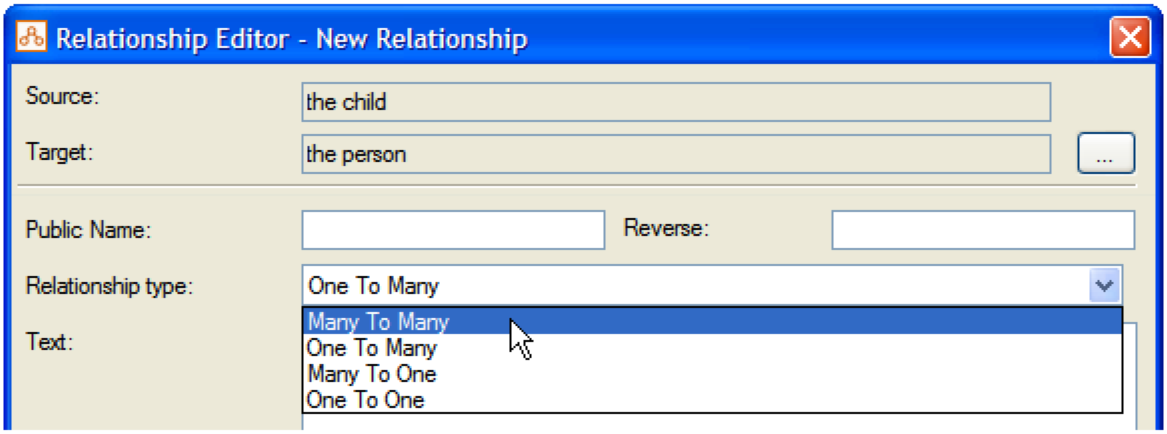
Target:

4. In the **Entity Selector** dialog box select the target entity.

the person
Defined in document: Properties.xsrc

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

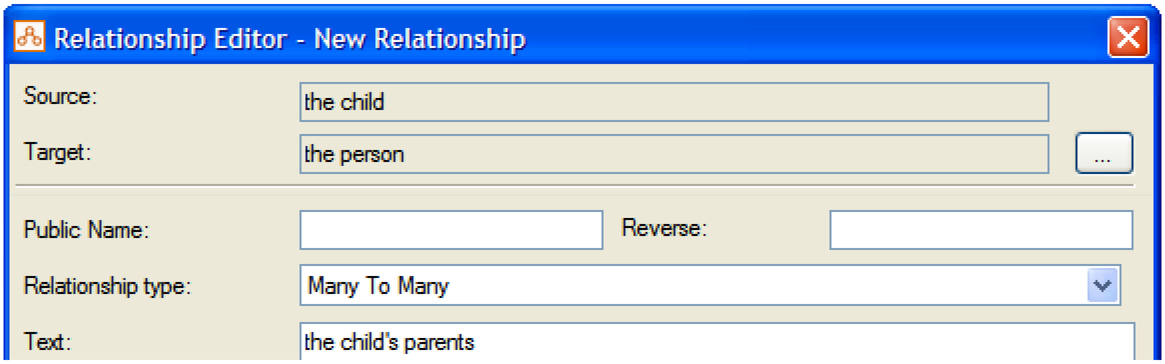
Public Name:

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

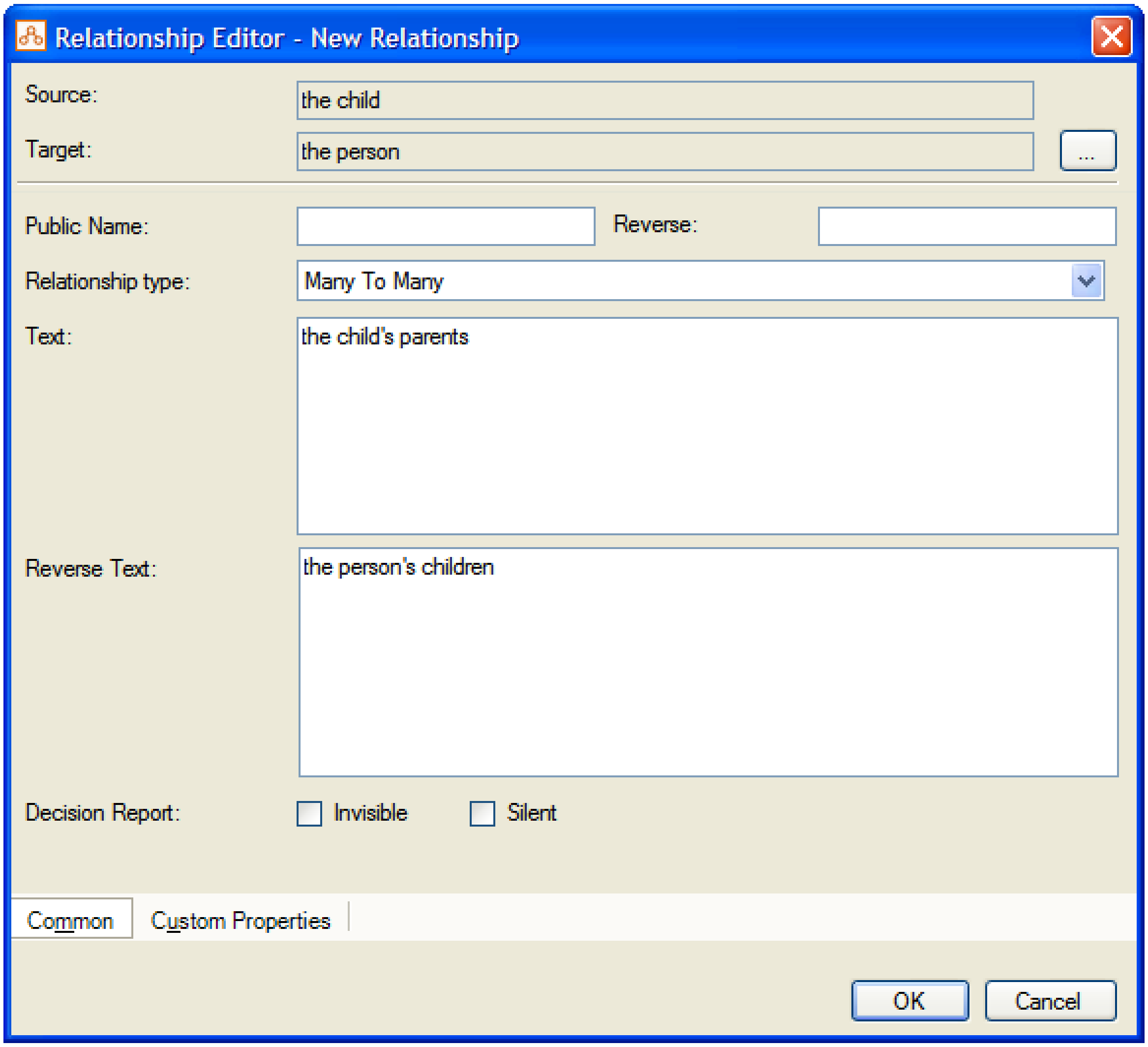
Public Name:

Reverse:

Relationship type: Many To Many

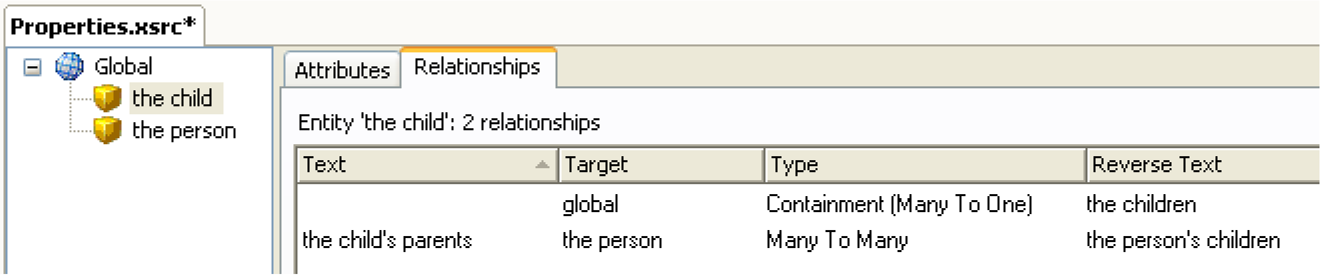
Text: the child's parents

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.



Source: the child
Target: the person
Public Name:
Reverse:
Relationship type: Many To Many
Text: the child's parents
Reverse Text: the person's children
Decision Report: ☐ Invisible ☐ Silent

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.



Entity 'the child': 2 relationships

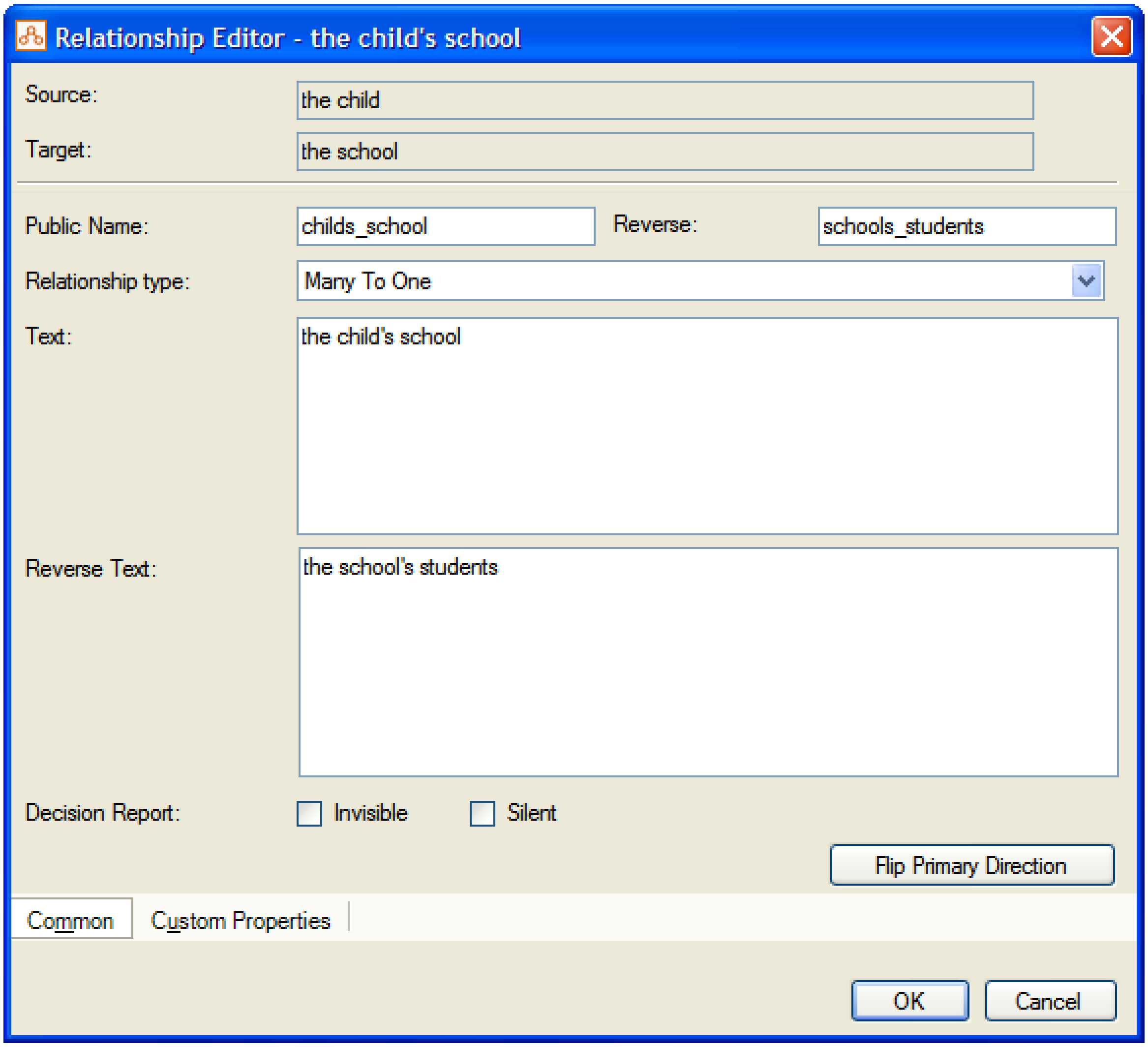
Text	Target	Type	Reverse Text
	global	Containment (Many To One)	the children
the child's parents	the person	Many To Many	the person's children

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**.



Relationship Editor - the child's school

Source: the child

Target: the school

Public Name: childs_school

Reverse: schools_students

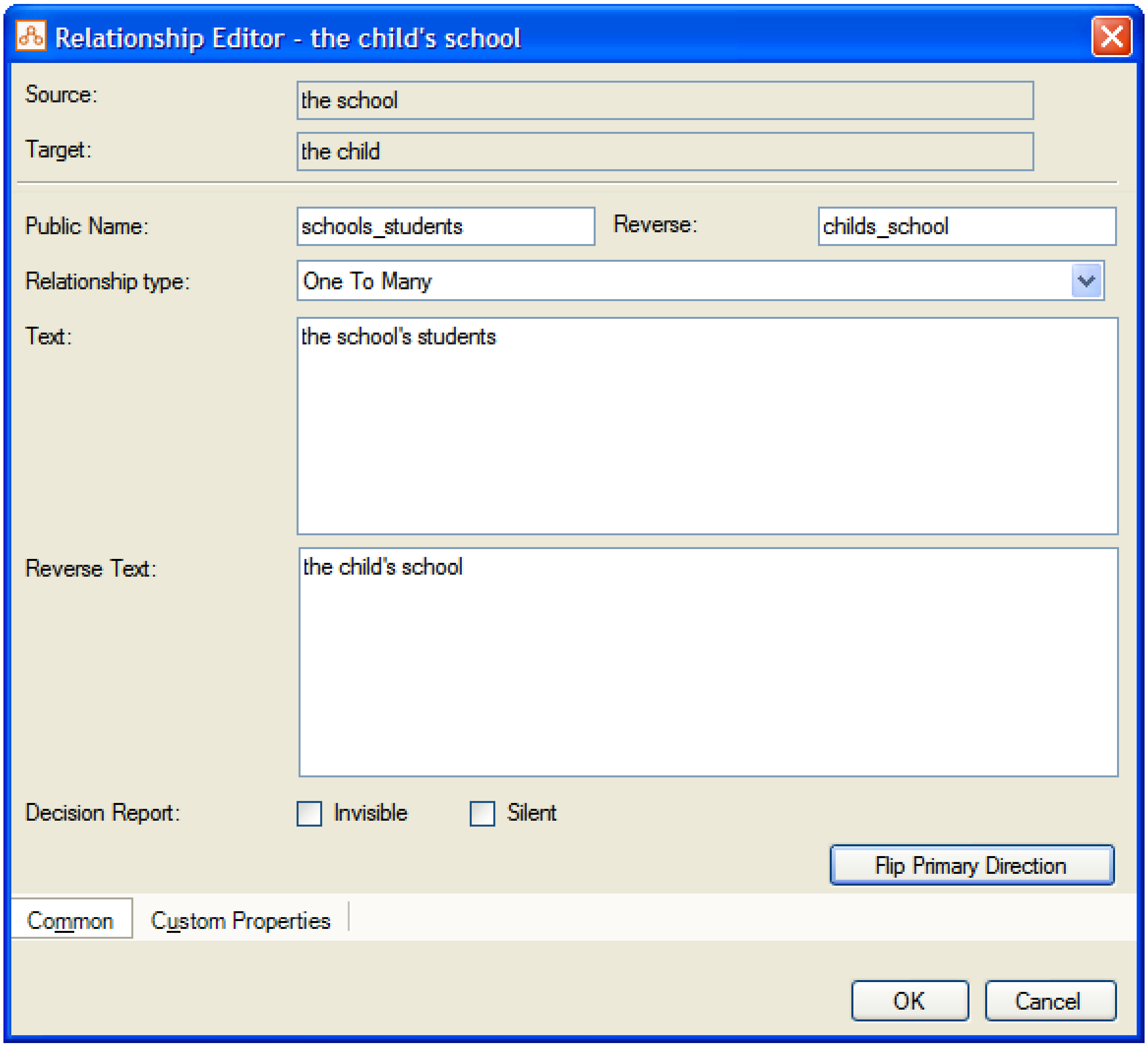
Relationship type: Many To One

Text: the child's school

Reverse Text: the school's students

Decision Report: ☐ Invisible ☐ Silent

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.



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

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 source entity text so that it is clear from the relationship name who the relationship is from.

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	123	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
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
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:

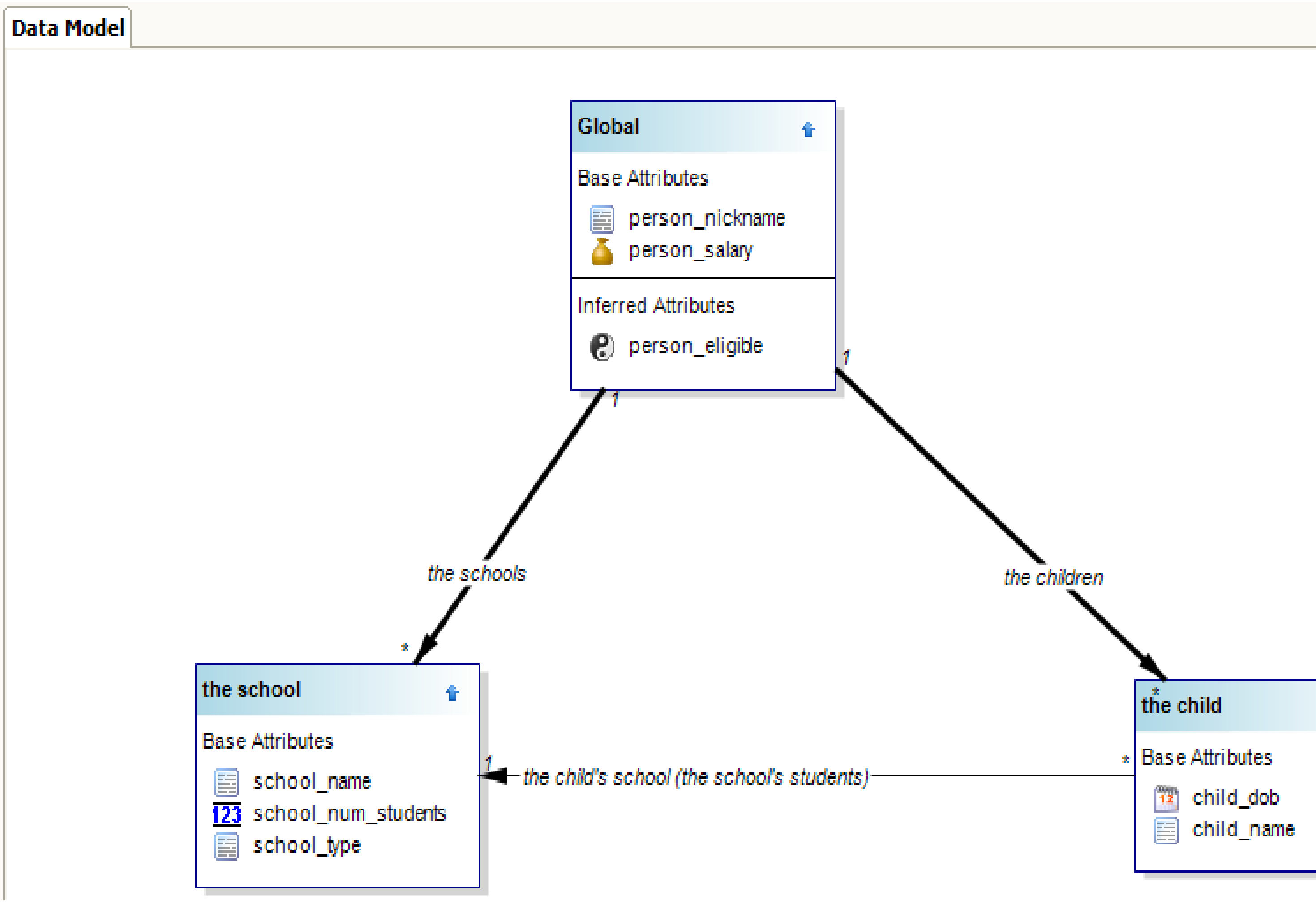
1. In Oracle Policy Modeling, select **Tools | Clean Up Unused Attributes and Relationships**.
2. In the **Clean Up Unused Attributes and Relationships** dialog box, select the unused relationships that you want to delete.
3. Click **OK**.

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

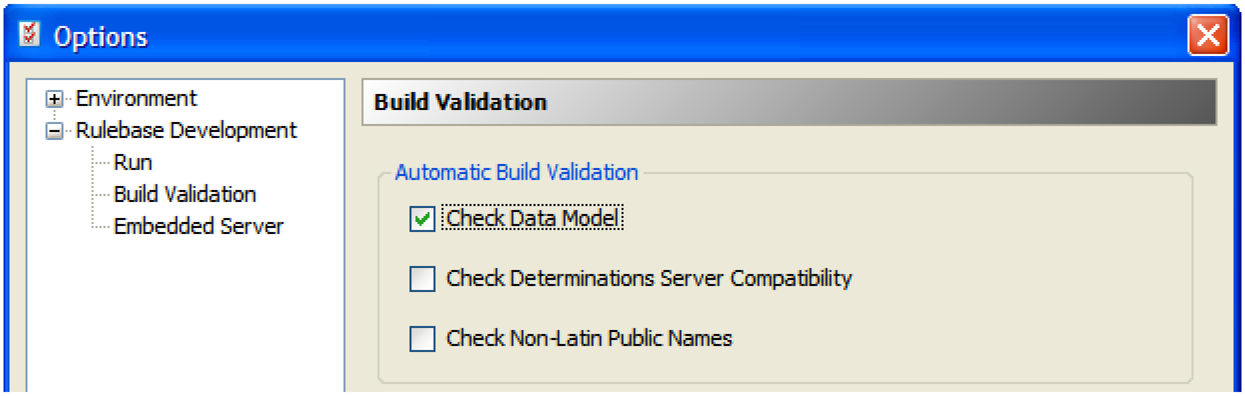
Create an external data model file for use with Oracle Policy Modeling

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**.



☑ Check Data Model

☐ Check Determinations Server Compatibility

☐ Check Non-Latin Public Names

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.

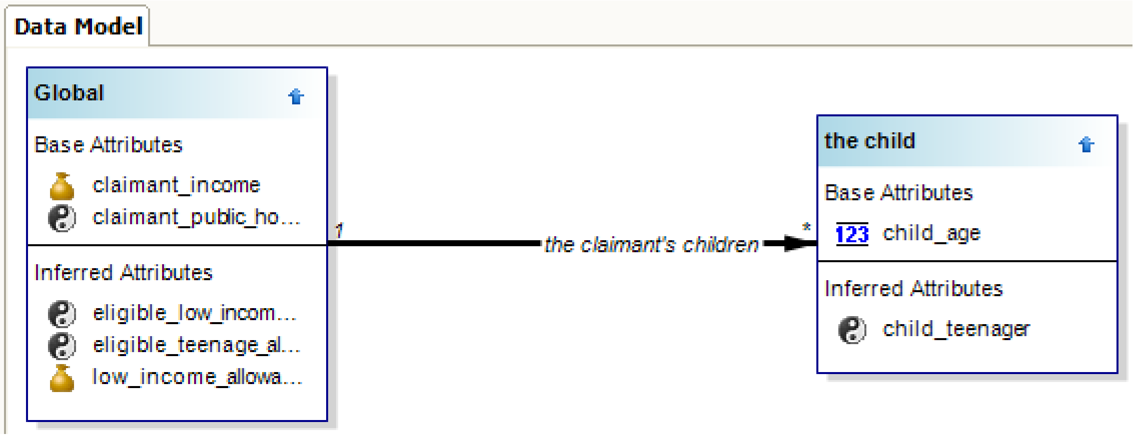
Creating the data model file from within Oracle Policy Modeling requires having a rulebase which already reflects the desired data model. This means creating all the entities and attributes you need in a blank new project or, if the rules are already quite well progressed, making sure the rulebase you 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).

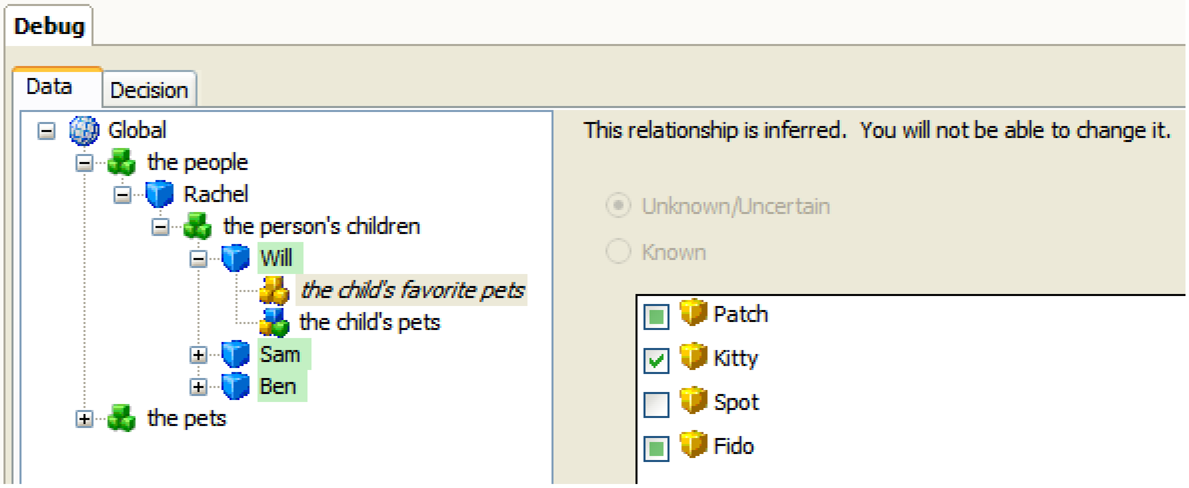
Entity completion status and inferred relationships

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:



This relationship is inferred. You will not be able to change it.

● Unknown/Uncertain

○ Known

🟩 Patch

☑ Kitty

☐ Spot

🟩 Fido

Here the relationship 'the child's favorite pets' is being examined for the child Will. The relationship editor is showing that:

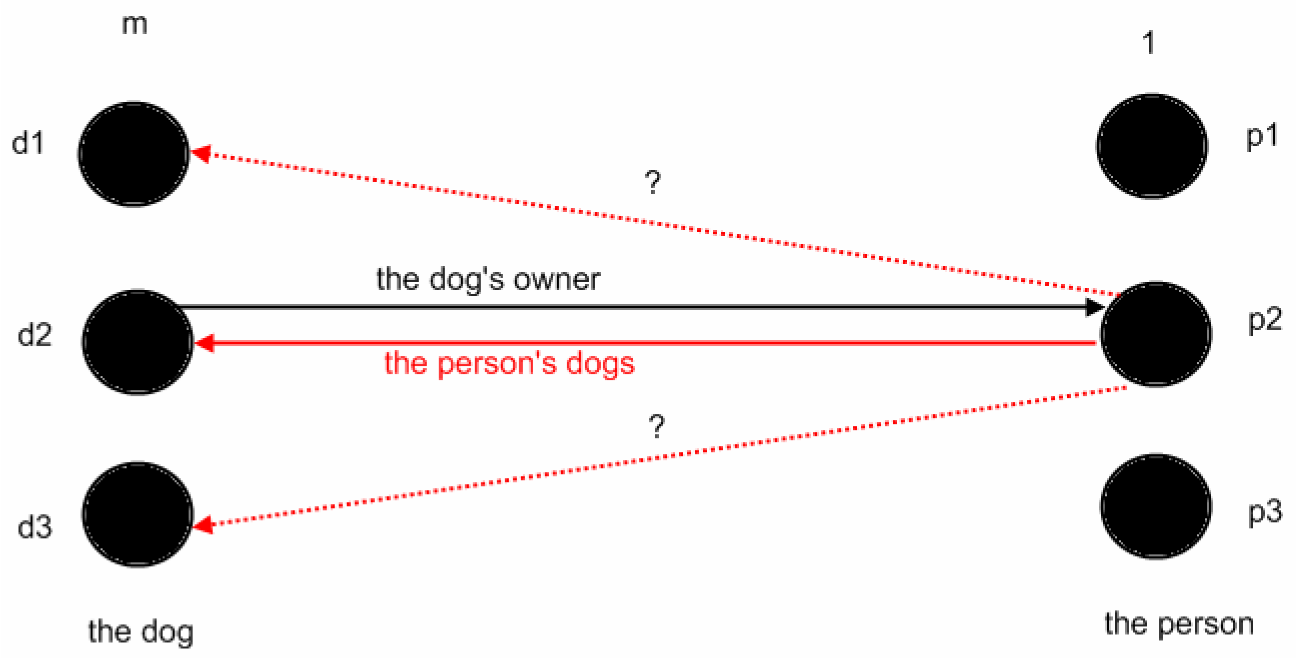
- the relationship is unknown - therefore this is a partial knowledge situation
- the pet Kitty is known to be a target of the relationship
- the pet Spot is known to *not* be a target of the relationship
- the pets Patch and Fido may or may not be targets of the relationship - this cannot be determined 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.

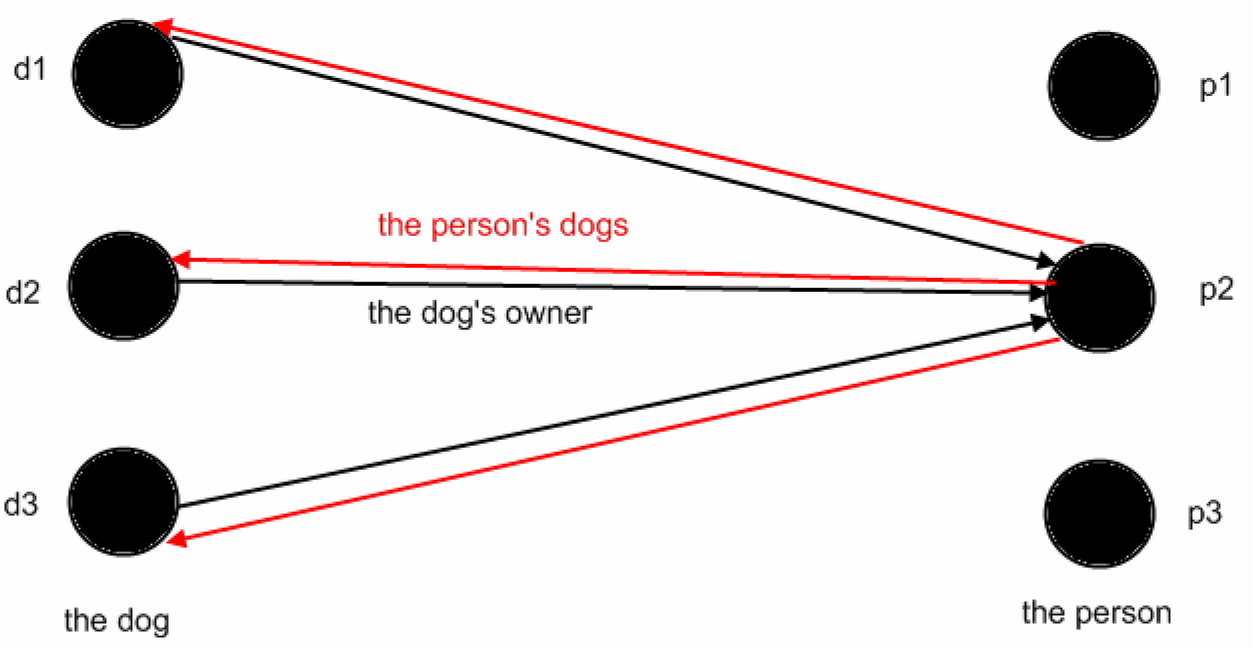
This leads to a situation of partial knowledge. For p2, the person's dogs is a partially known relationship. There is one known target, d2, and d1 and d3 may or may not be members of the relationship.

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:



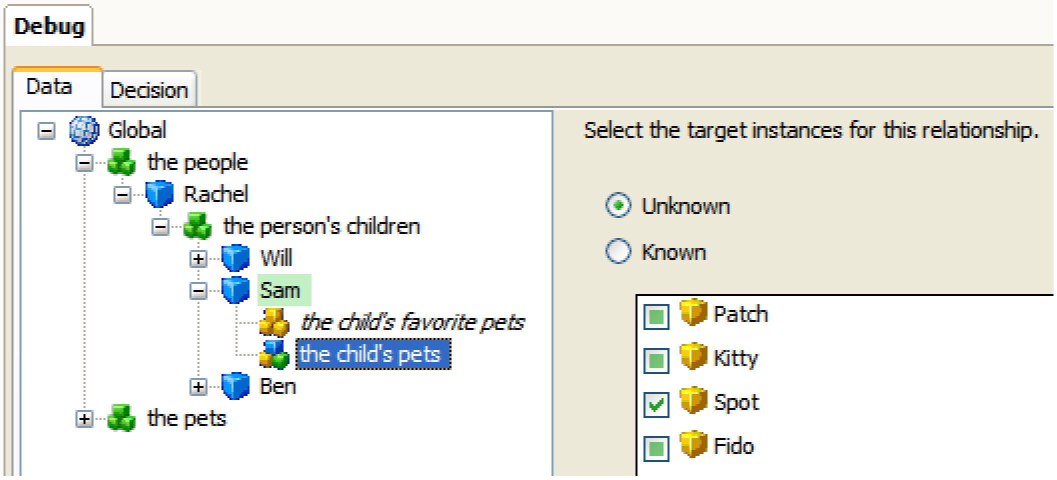
Here:

- 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:



Select the target instances for this relationship.

● Unknown
○ Known

▣ Patch
▣ Kitty
☑ Spot
▣ Fido

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.

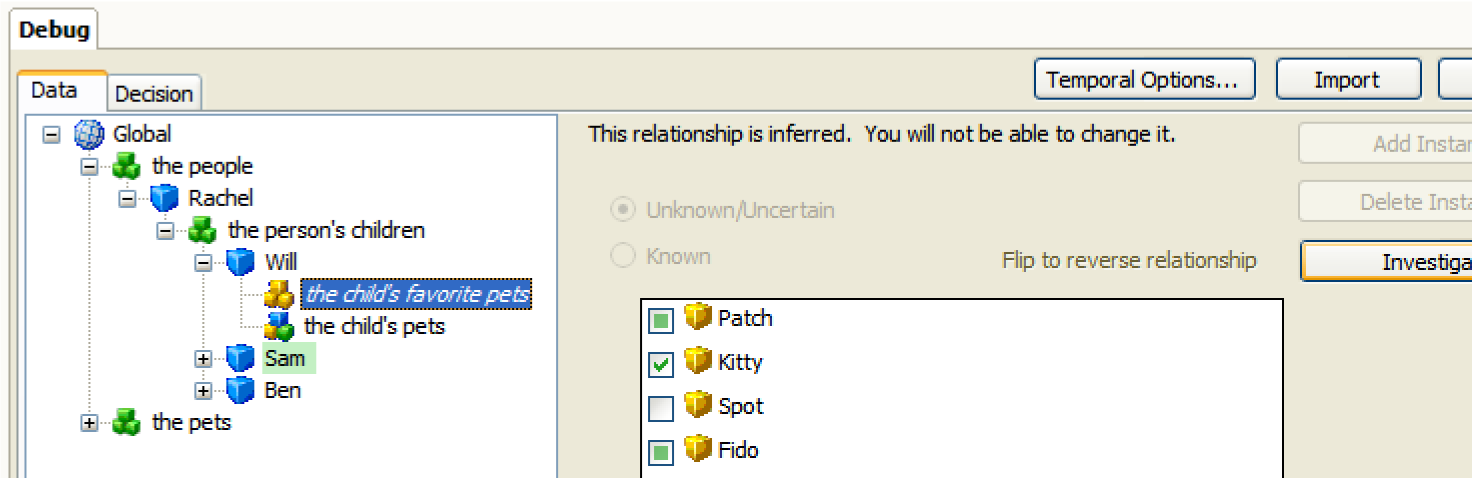
This is because the rule engine does not support "not targets" for static relationships.

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.



This relationship is inferred. You will not be able to change it.

● Unknown/Uncertain

○ Known

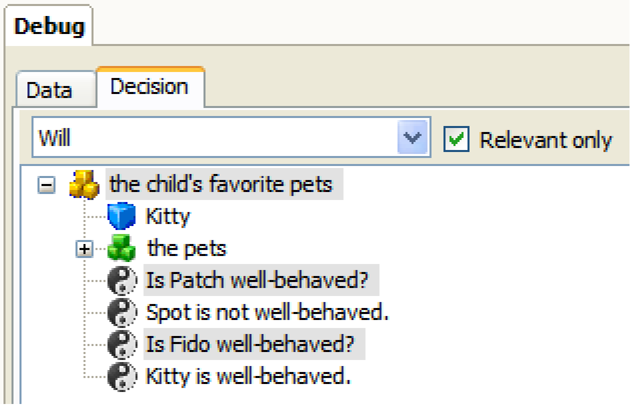
🟩 Patch

☑ Kitty

☐ Spot

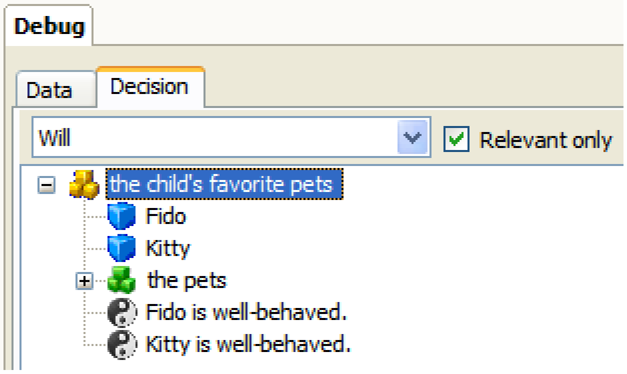
🟩 Fido

3. The Decision view will be shown with any relevant unknown attributes or relationships highlighted.

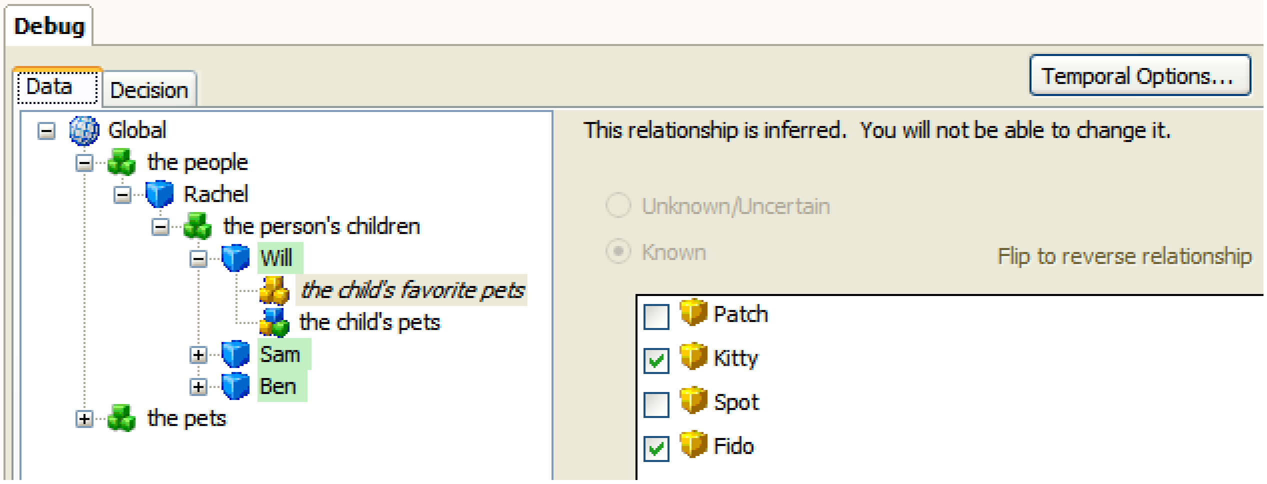


(If any unknown containment relationships are highlighted, right-click and choose **Edit Relationship**, to go to the relationship editor in the Data view. Complete the relationship by adding entity instances or using the **Containment Complete** option on the relationship in the Data view, then return 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.



This relationship is inferred. You will not be able to change it.

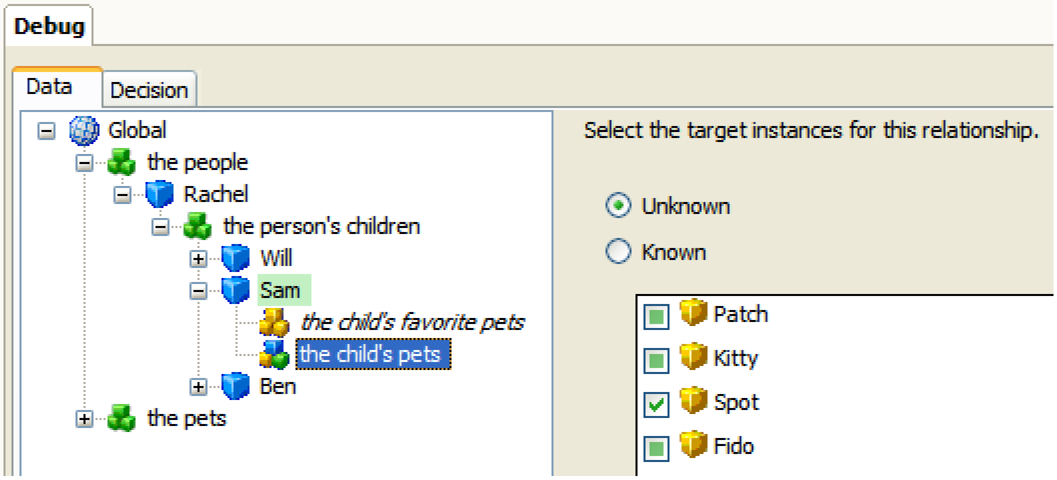
○ Unknown/Uncertain
● Known

☐ Patch
☑ Kitty
☐ Spot
☑ Fido

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.

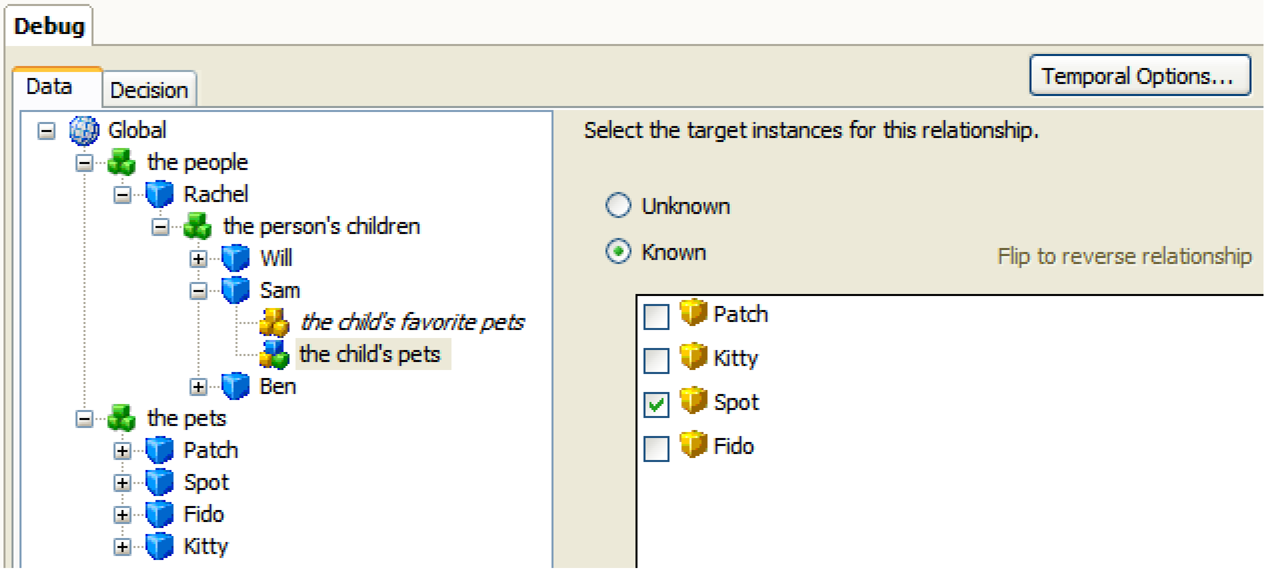


Select the target instances for this relationship.

● Unknown
○ Known

🟩 Patch
🟩 Kitty
☑ Spot
🟩 Fido

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.)



Select the target instances for this relationship.

○ Unknown

● Known

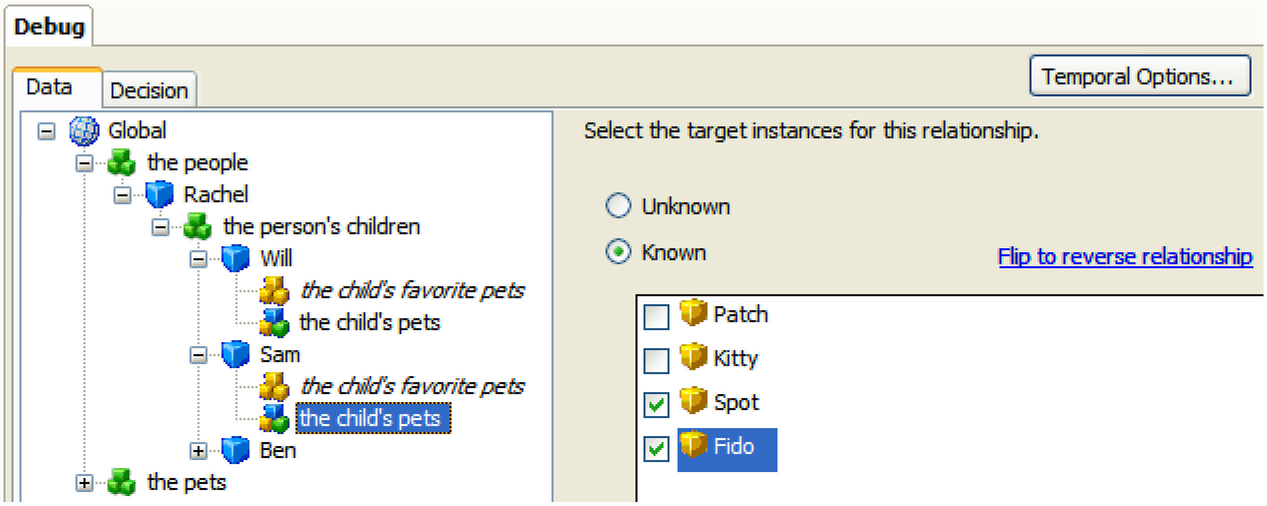
☐ Patch

☐ Kitty

☑ Spot

☐ Fido

3. Select the check box for any existing entity instances that you want to associate with that relationship.



Select the target instances for this relationship.

○ Unknown

● Known

☐ Patch

☐ Kitty

☑ Spot

☑ Fido

Understand containment relationships and entity completion

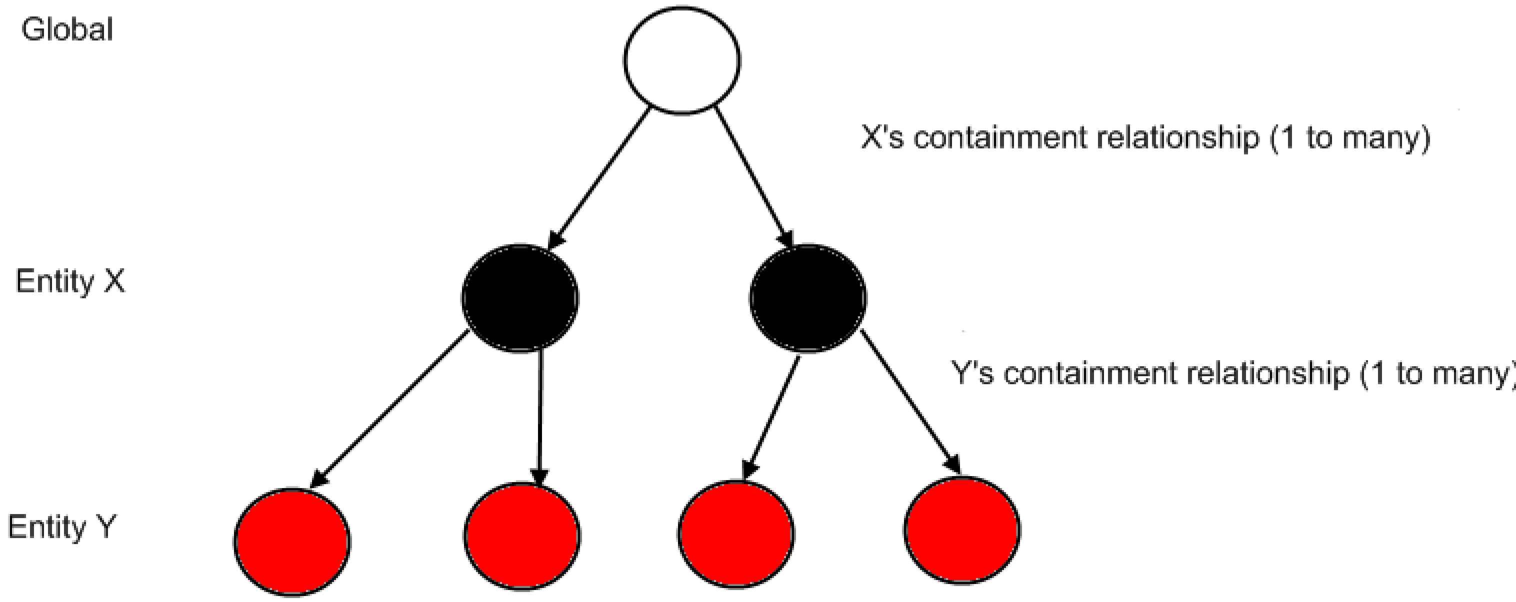
When an entity is considered to be *complete*, the rule engine assumes that it knows about the entire set of instances for that entity. An entity's completion status (whether or not it is considered to be complete) is of major importance when determining whether or not a relationship is partially known. 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.

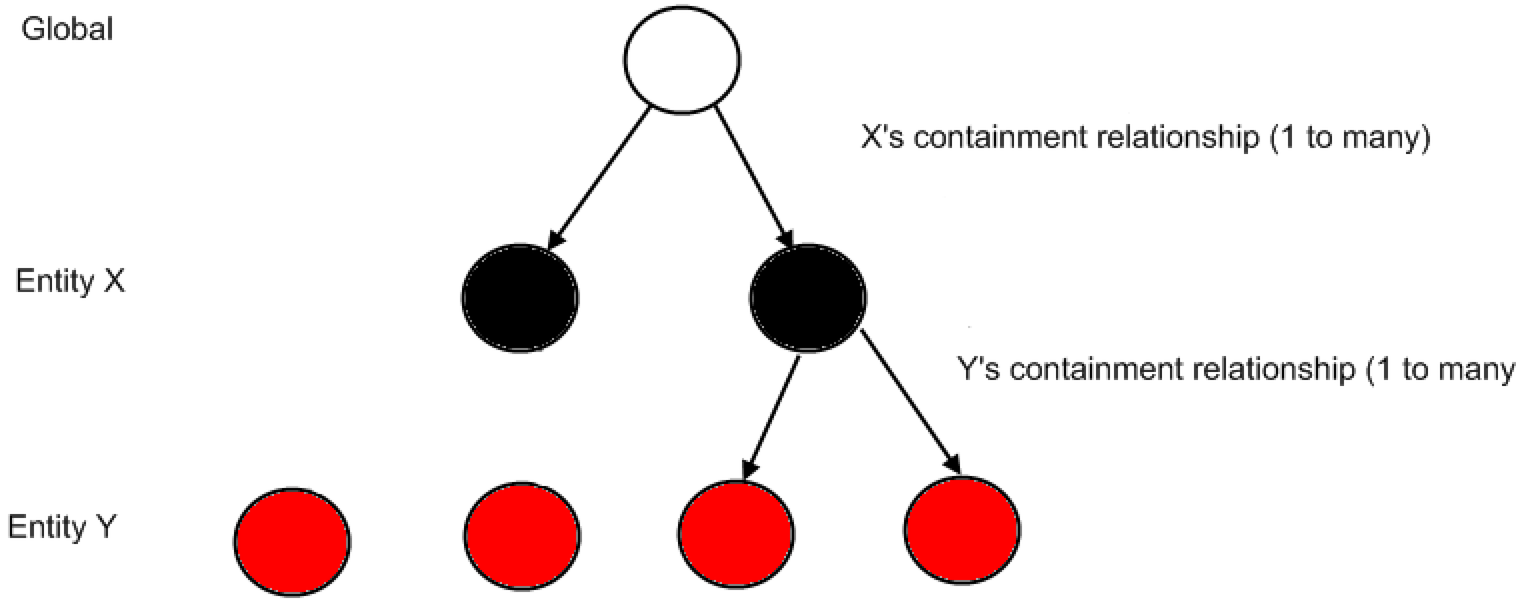
Consider the example provided in the following diagram:



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.

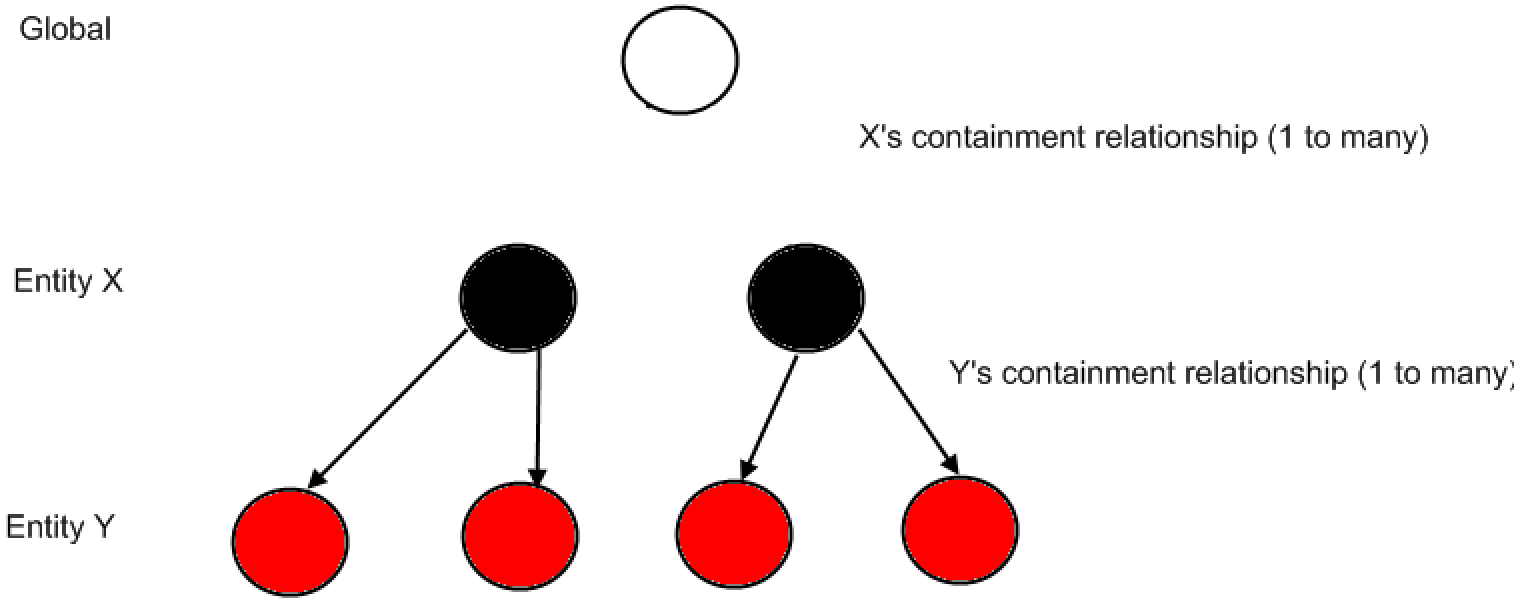
Consider this second scenario:



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:

1. 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.)
2. 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
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 and 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 workplace" is a text attribute in the entity "the person", and "the person's co-workers" is a many-to-many relationship whose source is "the person", and whose target is also "the 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

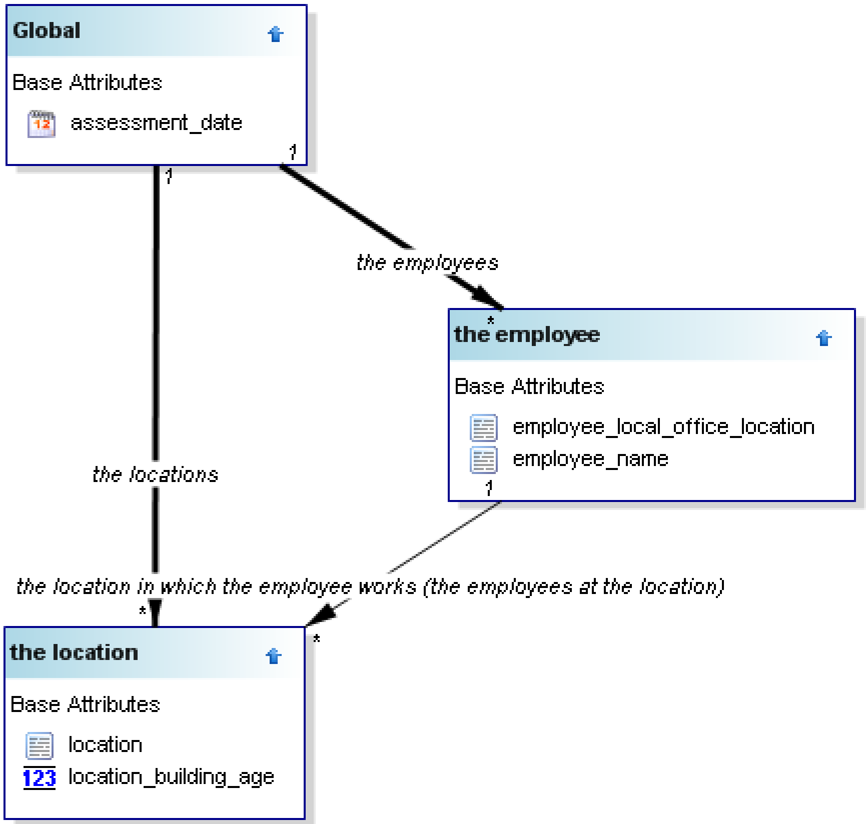
or in table form (where multiple instances are needed):

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:

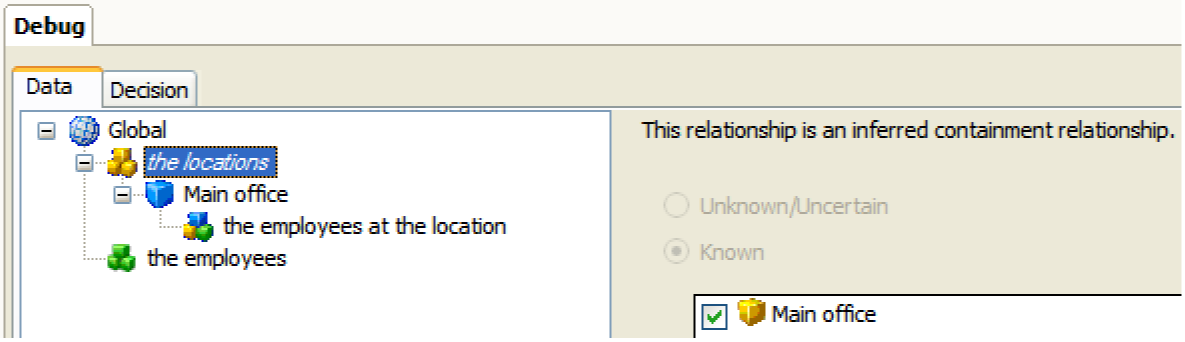


Example 1: Creating a single instance

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.



This relationship is an inferred containment relationship.

○ Unknown/Uncertain
● Known

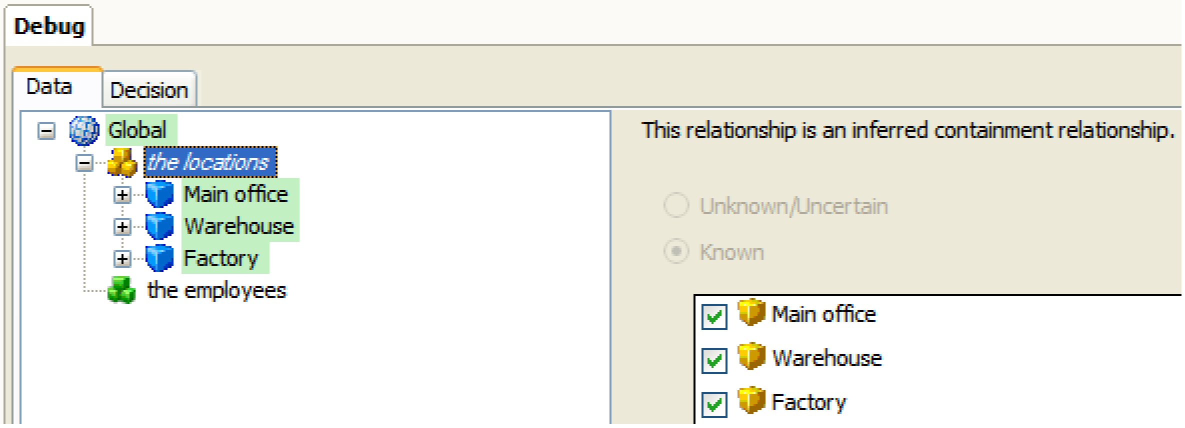
☑ Main office

Example 2: Creating multiple instances using a rule table

Writing the rule:

the locations	
"Main office"	the assessment date >= 2009-10-01
"Warehouse"	the assessment date >= 2000-01-15
"Factory"	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.



This relationship is an inferred containment relationship.

○ Unknown/Uncertain
● Known

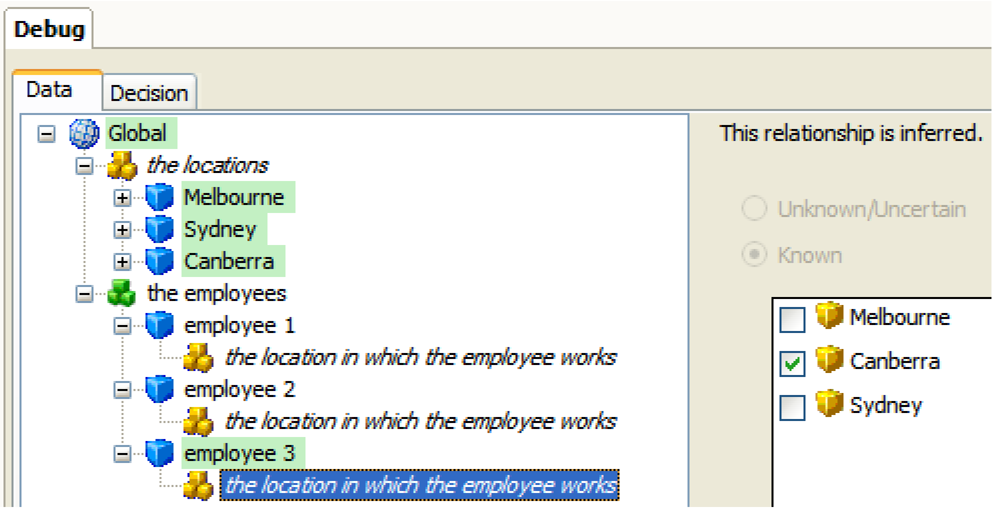
☑ Main office
☑ Warehouse
☑ Factory

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.



This relationship is inferred.

○ Unknown/Uncertain

● Known

☐ Melbourne

☑ Canberra

☐ Sydney

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

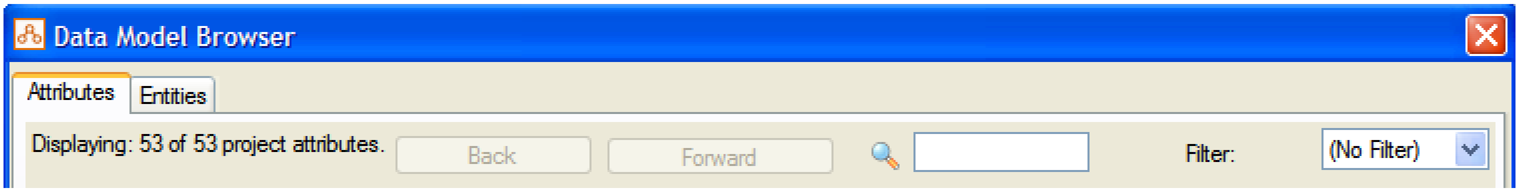
Edit an entity from within Word

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.



Displaying: 53 of 53 project attributes.

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.

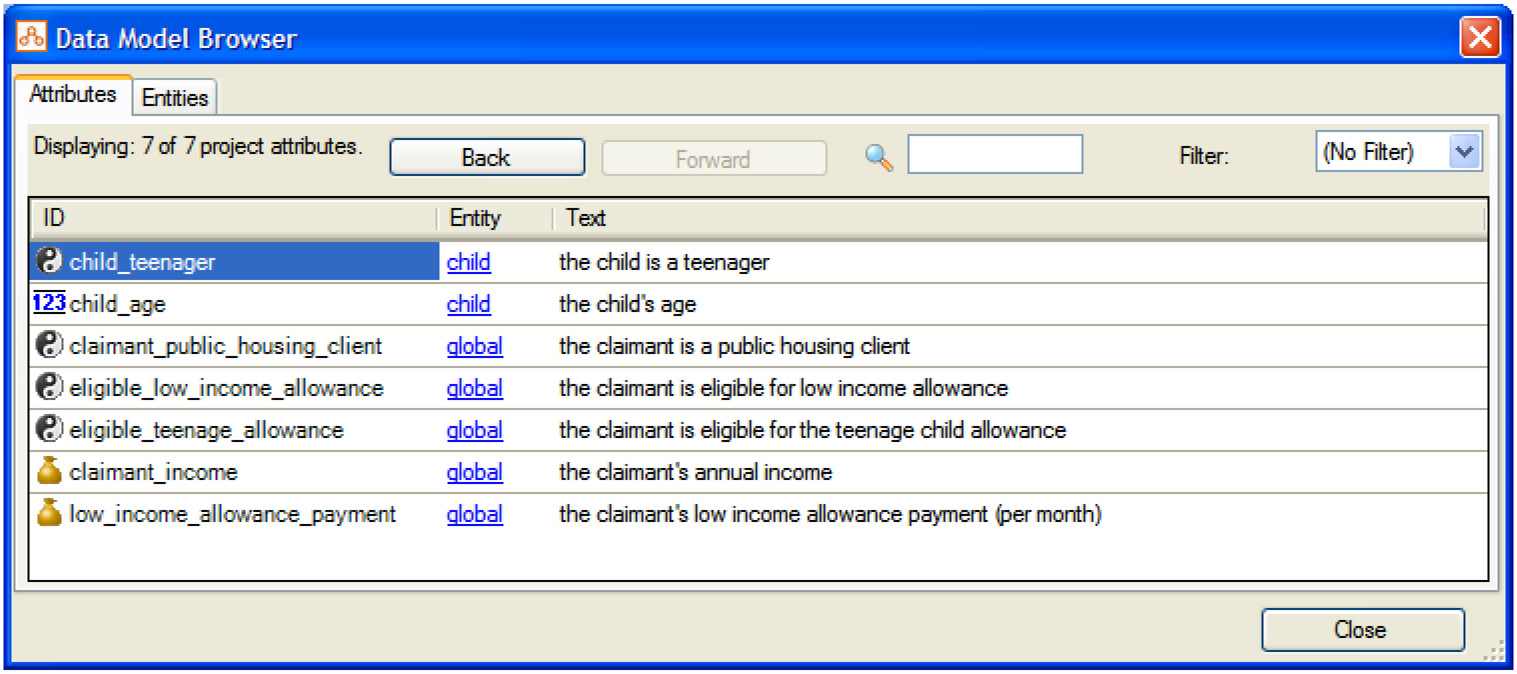
You can also sort lists by clicking on the column header - this will alternate between ascending and descending order.

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.



Displaying: 7 of 7 project attributes.

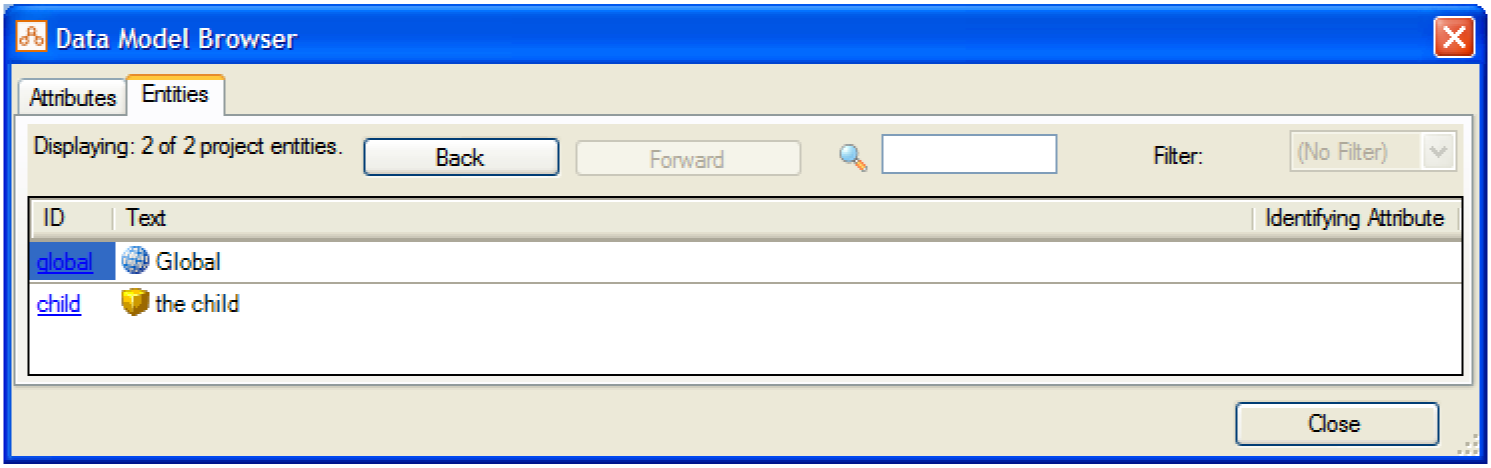
ID	Entity	Text
child_teenager	child	the child is a teenager
123 child_age	child	the child's age
claimant_public_housing_client	global	the claimant is a public housing client
eligible_low_income_allowance	global	the claimant is eligible for low income allowance
eligible_teenage_allowance	global	the claimant is eligible for the teenage child allowance
claimant_income	global	the claimant's annual income
low_income_allowance_payment	global	the claimant's low income allowance payment (per month)

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).



Displaying: 2 of 2 project entities.

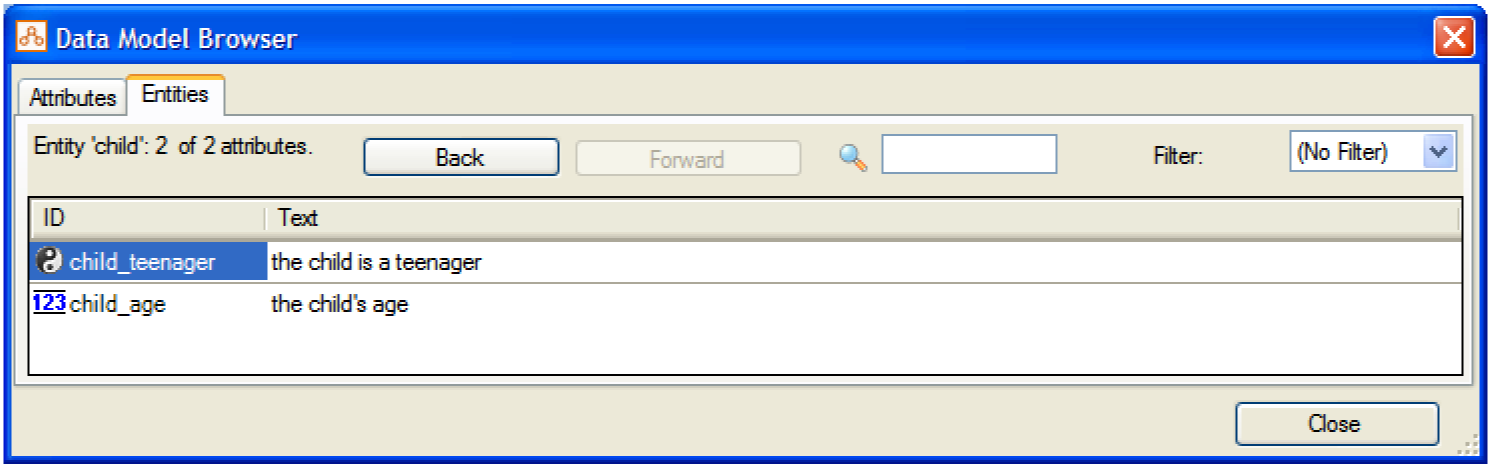
ID	Text	Identifying Attribute
global	Global	
child	the child	

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.



Entity 'child': 2 of 2 attributes.

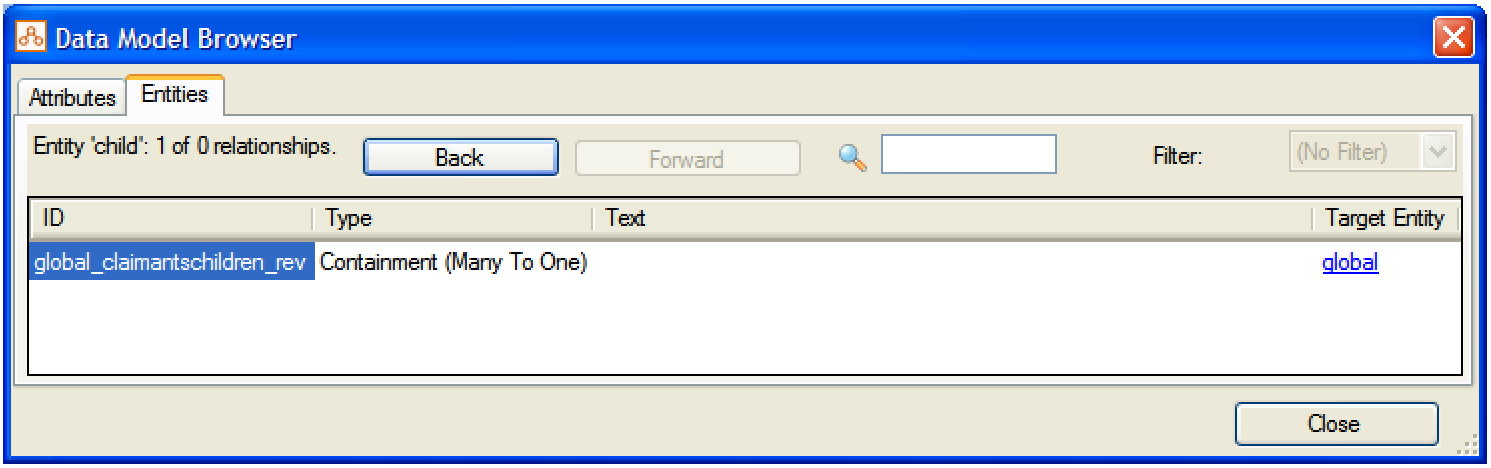
ID	Text
child_teenager	the child is a teenager
123 child_age	the child's age

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.



Entity 'child': 1 of 0 relationships.

ID	Type	Text	Target Entity
global_claimantschildren_rev	Containment (Many To One)		global

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 timeframe (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

the person's daily entitlement for pension	
the standard daily rate of benefit	the person is not married and the person satisfies the age requirement
the standard daily rate of benefit * 1.5	the person is married and the person satisfies the age requirement
0	**otherwise**

Standard daily rate

the standard daily rate of benefit	
5	TemporalBefore(2006-07-01)
6	TemporalOnOrAfter(2006-07-01) and the person is not married
7	TemporalOnOrAfter(2006-07-01) and the person is married
0	**otherwise**

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:

the standard daily rate of benefit	
5	TemporalBefore(2006-07-01)
6	TemporalOnOrAfter(2006-07-01) and the person is not married
7	TemporalOnOrAfter(2006-07-01) and the person is married
0	**otherwise**

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

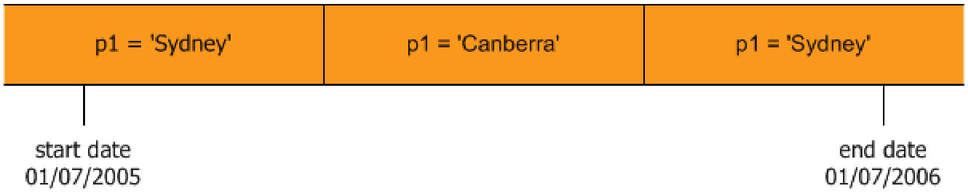
The Interval Count Distinct function counts the number of known distinct values for a variable, in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

- 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

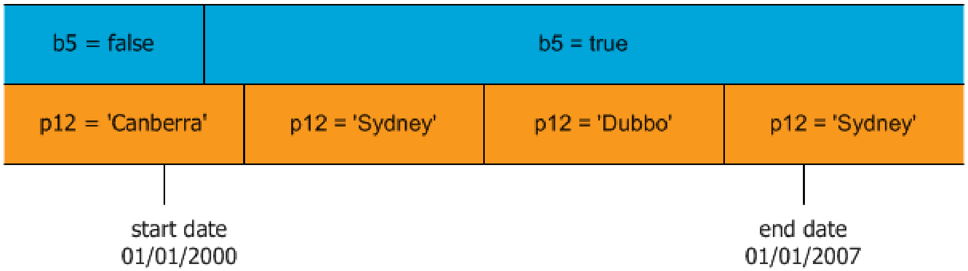
The Interval Count Distinct If function counts the number of known distinct values for an attribute, in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true. The syntax for this function is:

- 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

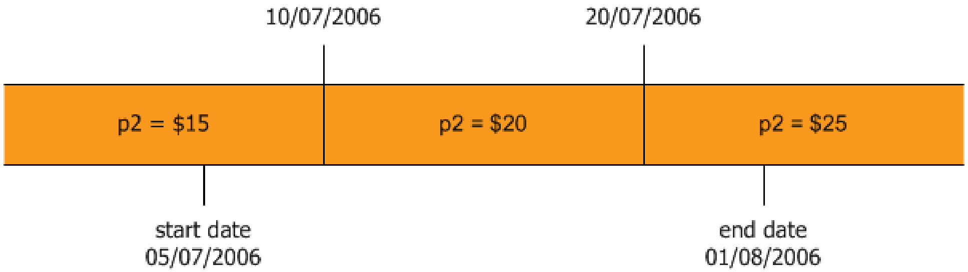
The Interval Daily Sum function calculates the sum of a currency or number variable, in the interval from the specified start date (inclusive) to the end date (exclusive). The attribute is assumed to be a daily quantity. The syntax for this function is:

- 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

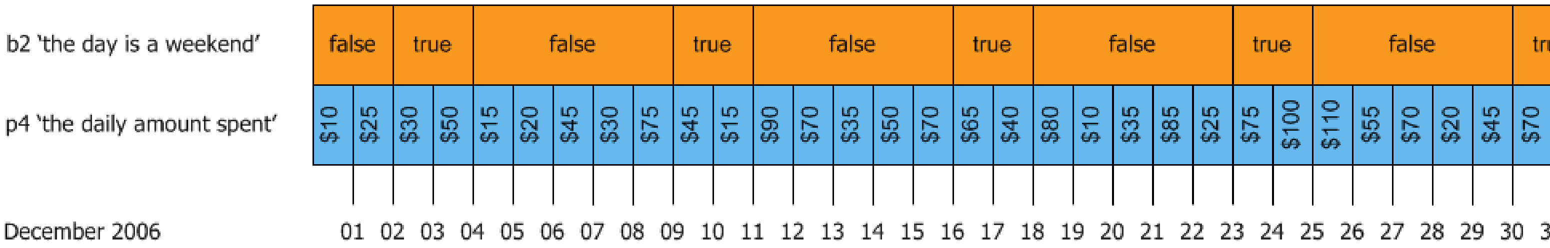
The Interval Daily Sum If function calculates the sum of all the daily values for a currency or number variable, in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true. The syntax for this function is:

- 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:



That is, $30 + $50 + $45 + $15 + $65 + $40 + $75 + $100 + $70 + $40 = $530

Calculate the average value of a variable in a time period

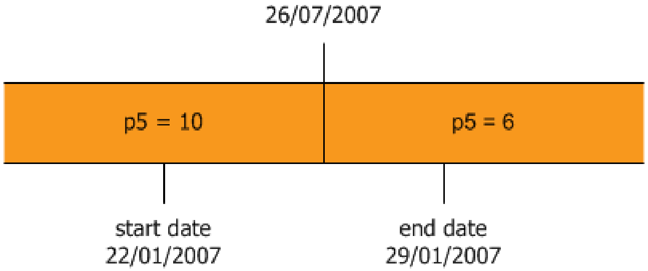
The Weighted Average function calculates the average value of a currency or number variable in the interval from the specified start date (inclusive) to the end date (exclusive) weighted by the time span to which each value applies. The syntax for this function is:

- 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

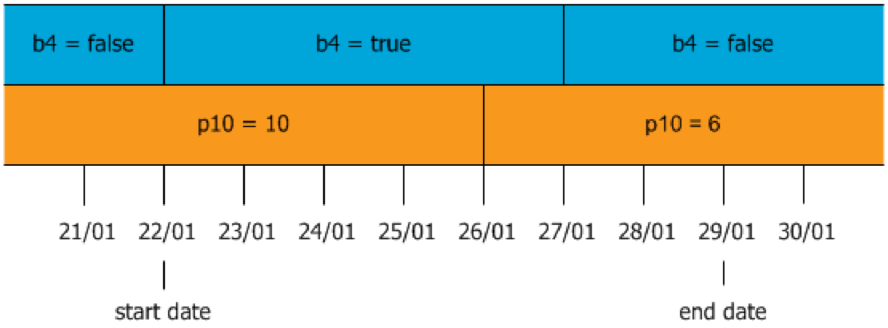
The Weighted Average If function calculates the average value of a currency or number variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true (weighted by the time span to which each value applies and where the 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

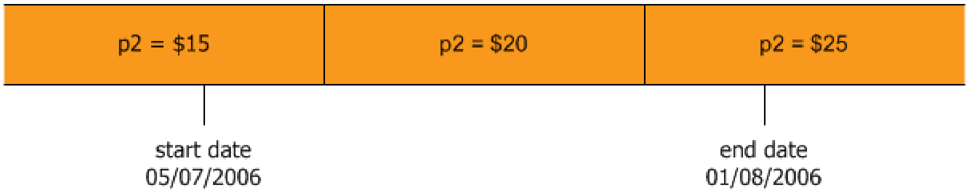
To find the maximum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive) you use the Interval Maximum function. The syntax for this function is:

- 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

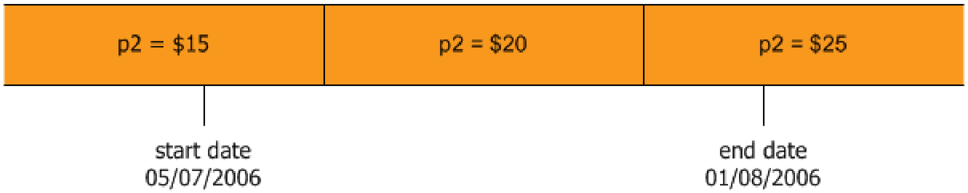
To find the minimum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive) you use the Interval Minimum function. The syntax for this function is:

- 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

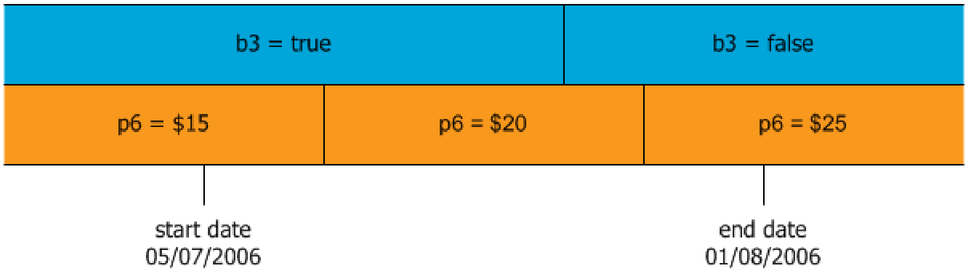
To find the maximum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true, you use the Interval Maximum If function. The syntax for this function is:

- 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

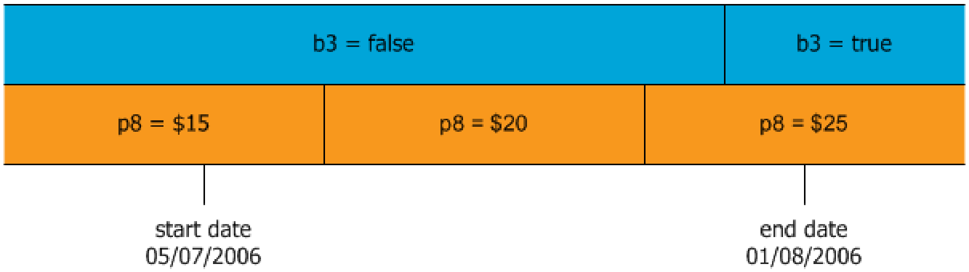
To find the minimum value of a variable in the interval from the specified start date (inclusive) to the end date (exclusive), only including times when a boolean filter is true, you use the Interval Minimum If function. The syntax for this function is:

- 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)**

This function returns a value of $25 for 'the minimum rate of benefit payable during the assessment period' (p8) where b3 is 'the client is also eligible for the benefit' for the following data:



Check if a condition is true within a time period

To check if a condition is true within a time period you use the Interval Verifier functions (Interval Always, Interval Sometimes, Interval At Least Days, Interval Consecutive Days) and the Temporal Verifier functions (Temporal Always Days, Temporal Consecutive Days, Temporal Sometimes Days).

The Interval Verifier functions work on boolean attributes and verify whether the attribute is true always, sometimes or for a consecutive number of days within a time span.

The Temporal Verifier functions operate on boolean attributes and return a boolean attribute that varies over time. They verify if the boolean attribute is true always, sometimes or for a consecutive number of days within a specified number of preceding days.

What do you want to do?

Check if a condition is true at all times in the time period

Check if a condition is ever true in the time period

Check if a condition is true for at least the specified number of days in the time period

Check if a condition is true for at least the specified number of consecutive days in the time period

Check if a condition is true for all of a specified number of preceding days

Check if a condition is true for at least the specified number of consecutive preceding days

Check if a condition is ever true within a specified number of preceding days

Check if a condition is true at all times in the time period

The Interval Always function returns true if and only if the attribute is true at all times in the interval from the specified start date (inclusive) to the end date (exclusive). The syntax for this function is:

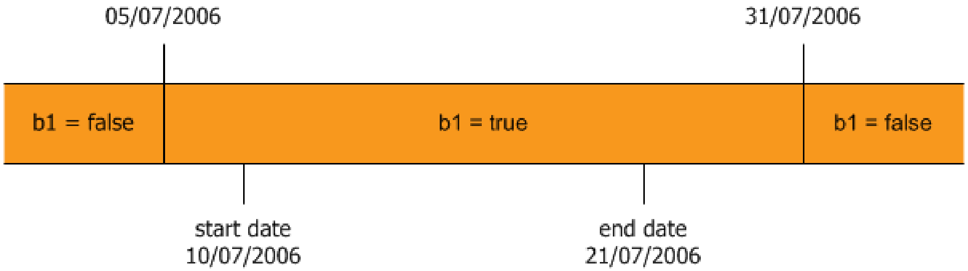
- IntervalAlways(<start date>,<end date>,<boolean>)

For example, the Interval Always function could be used to determine whether the client was in jail at all times between 10 July 2006 and 20 July 2006 (inclusive). In Word you would write this rule as:

the client was in jail at all times during the assessment period if

IntervalAlways(2006-07-10,2006-07-21,the client was in jail)

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:

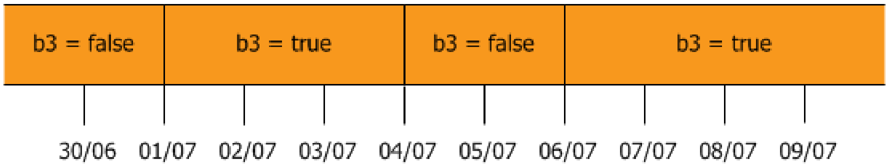
- IntervalAtLeastDays(<start date>,<end date>,<number>,<boolean>)

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:

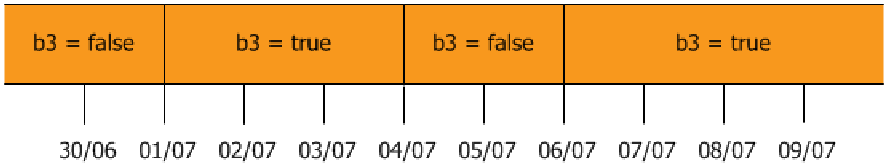
- IntervalConsecutiveDays(<start date>,<end date>,<number>,<boolean>)

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:

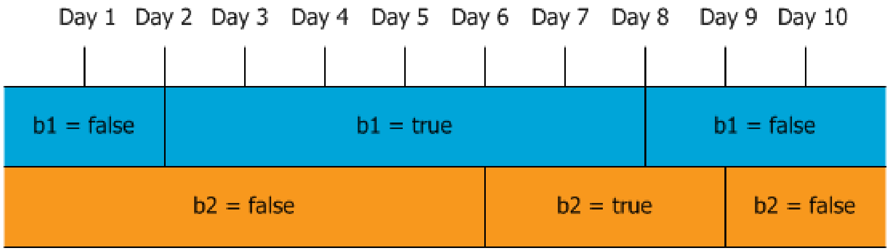
- TemporalAlwaysDays(<number>,<boolean>)

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:

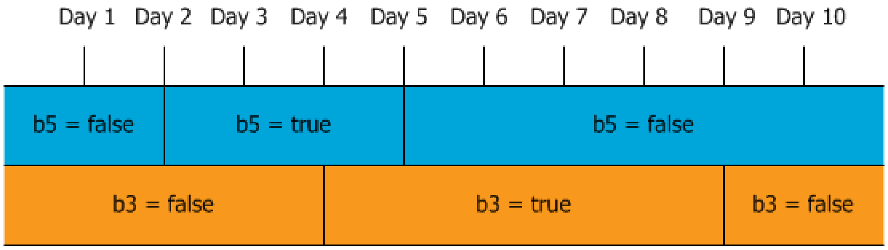
- TemporalConsecutiveDays(<mindays number>,<daycount number>,<boolean>)

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:

a. 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:

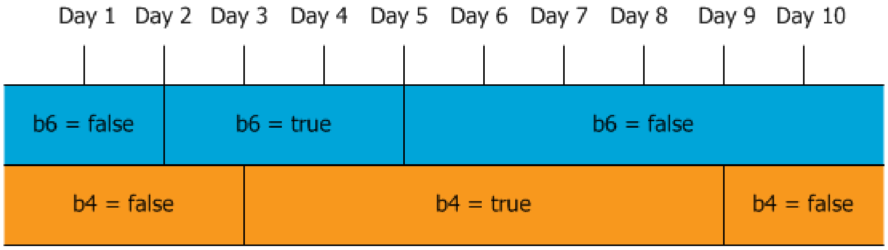
- TemporalSometimesDays(<number>,<boolean>)

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'.)

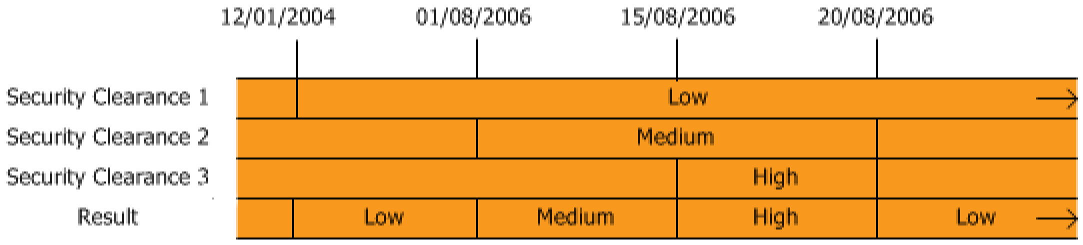
If this rule is applied to the sample data below, the result is that 'the person's effective security clearance' is {uncertain, Low from 01/07/2005, Medium from 01/07/2006, High from 01/07/2007, uncertain from 01/07/2008}.

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):

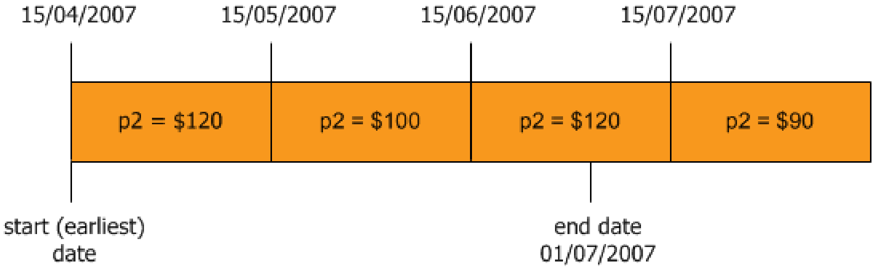
> 15/4/2007: $120, 15/5/2007: $100, 15/6/2007: $120, 15/7/2007: $90

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()

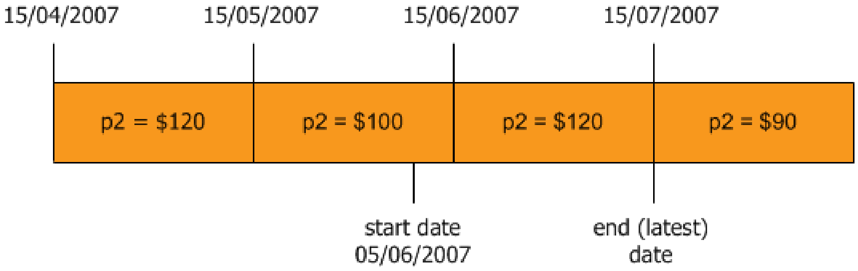
For example, an applicant is paid a benefit monthly, and the amount of the payment is a temporal value with values on specific days as follows (and 0 elsewhere):

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

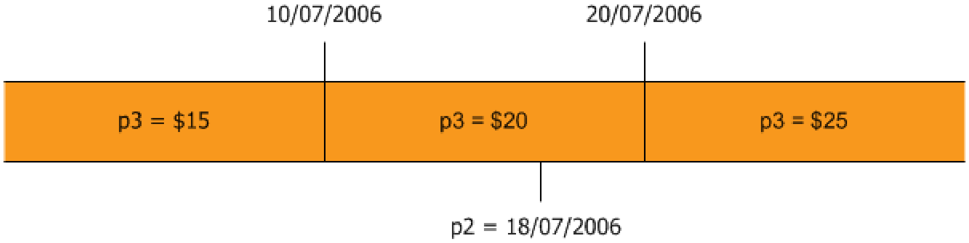
To determine the value of an attribute at a particular date you use the Value At function. This function has the following syntax:

- 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

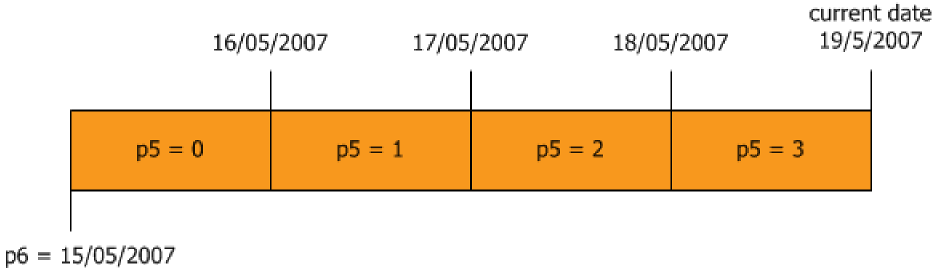
To calculate the number of full days since a given date, you use the Temporal Days Since function. Note that this function will return a number variable that varies every day. The function has the following syntax:

- 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

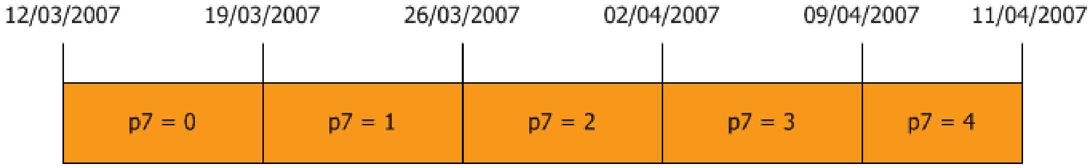
To calculate the number of full weeks since a given date, you use the Temporal Weeks Since function. Note that this function will return a number variable that varies every week. The function has the following syntax:

- 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:

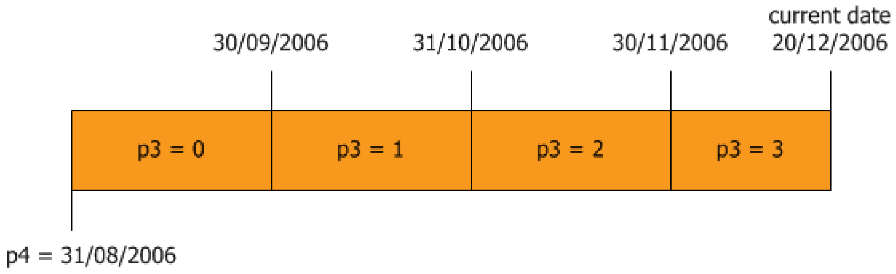
- TemporalMonthsSince(<start date>,<end date>)

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

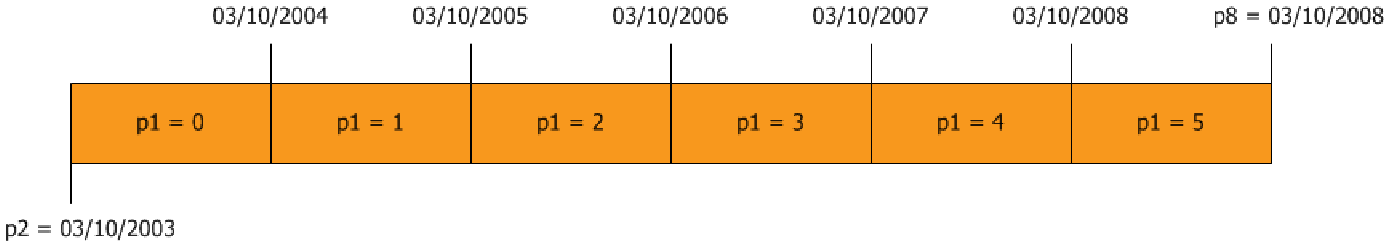
To calculate the number of full years since a given date, you use the Temporal Years Since function. Note that this function will return a number variable that varies every year. The function has the following syntax:

- 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:

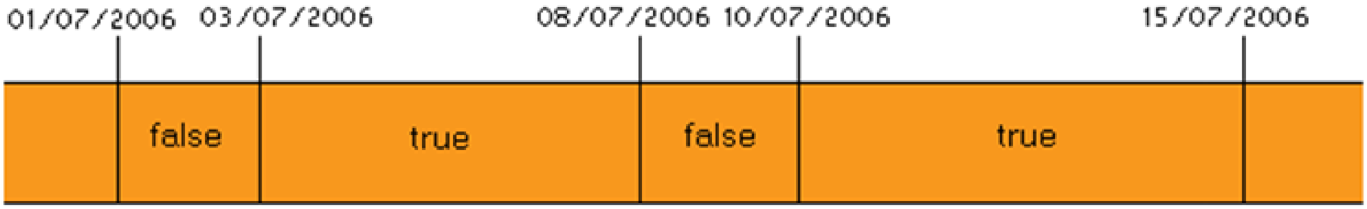
- TemporalIsWeekday(<start date>,<end date>)

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:

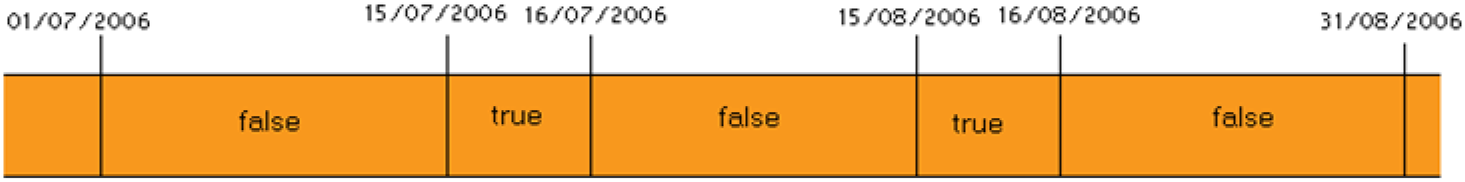
- TemporalOncePerMonth(<start date>,<end date>,<day-of-month>)

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

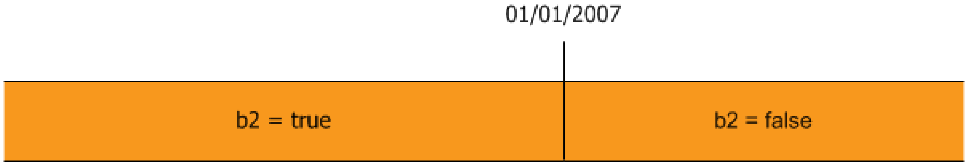
The Temporal Before function returns a boolean attribute that varies over time and is true before a given date and false on and afterwards. The syntax for this function is:

- 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

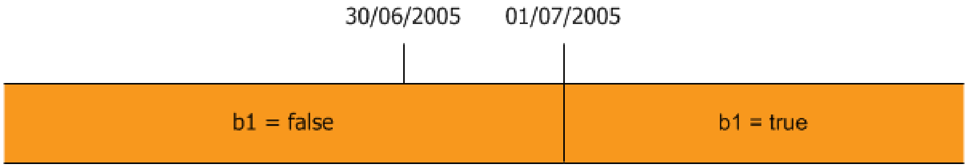
The Temporal After function returns a boolean attribute that varies over time and is true after a given date and false on and before. The syntax for this function is:

- 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:

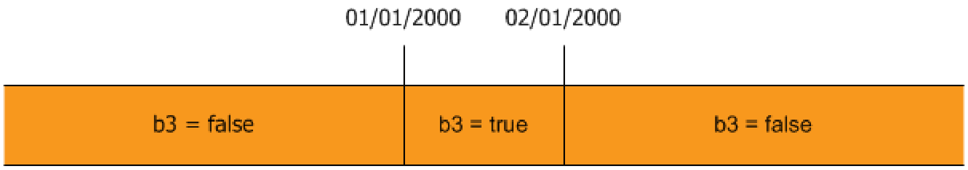
- TemporalOn(<date>)

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:

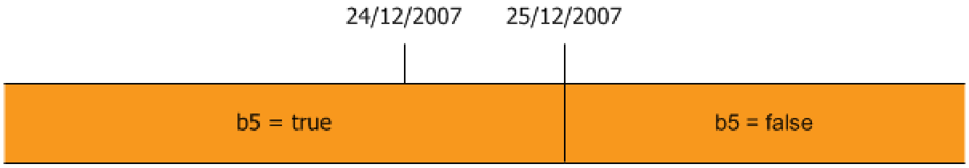
- TemporalOnOrBefore(<date>)

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:

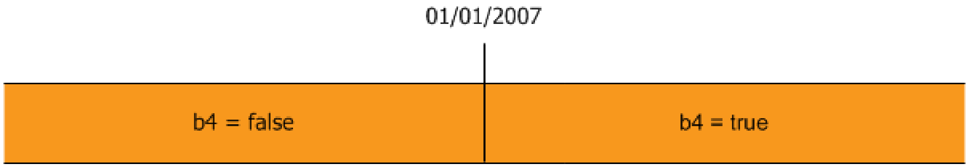
- TemporalOnOrAfter(<date>)

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.



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.

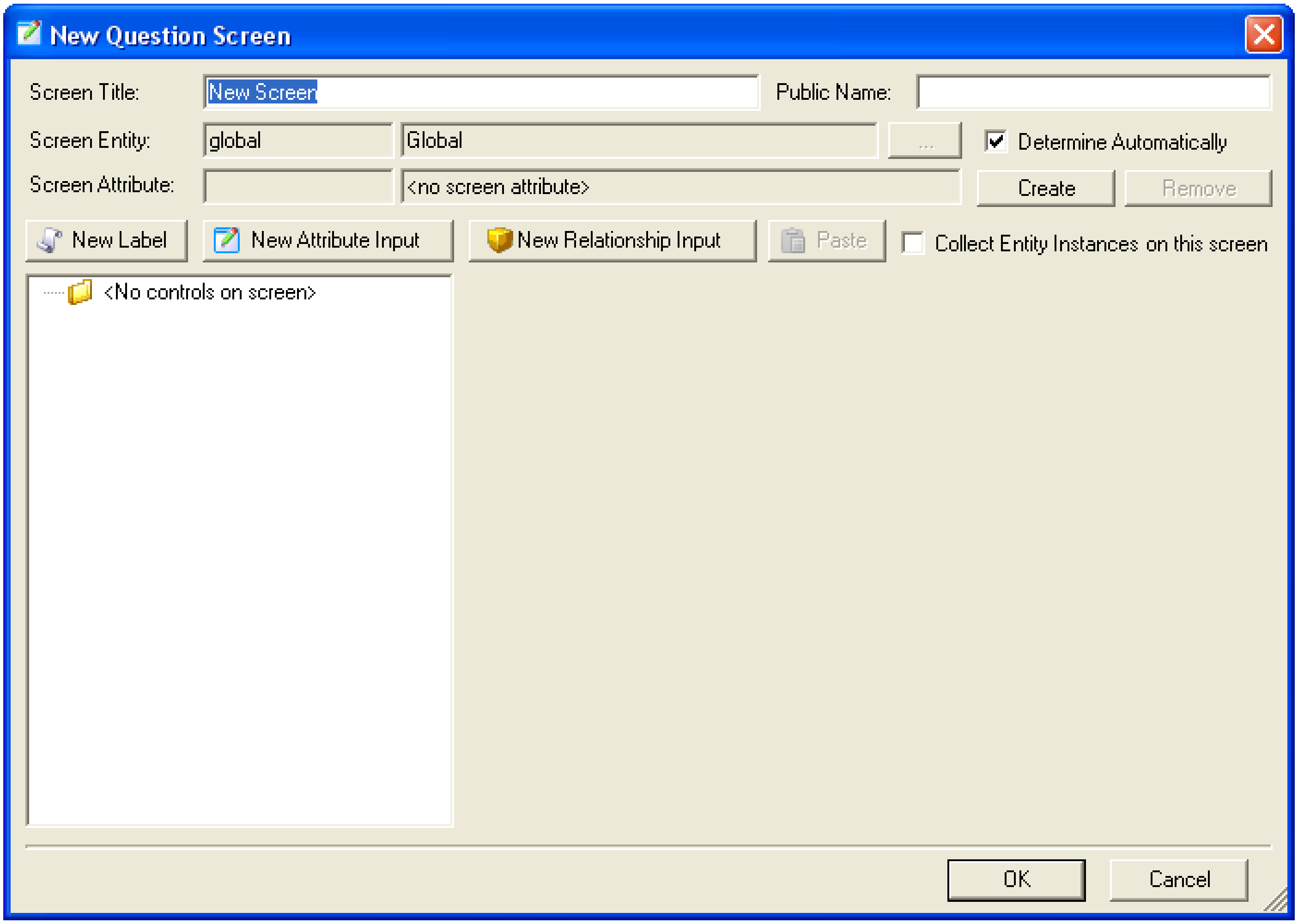
TIP: Question screen folders in a screen order are used to define the 'stages' or groupings that are displayed at the top of an interview to indicate progress through the investigation.

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:



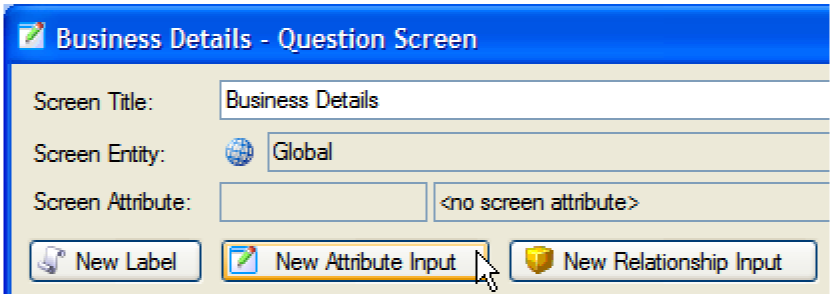
3. Enter an appropriate name for the screen in the **Screen Title** text box.
   TIP: Screen names are used as the first heading on a screen. If you use a clear screen name which makes it easy to understand the point of the current screen, the user will be much more receptive to the application. Screen 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.

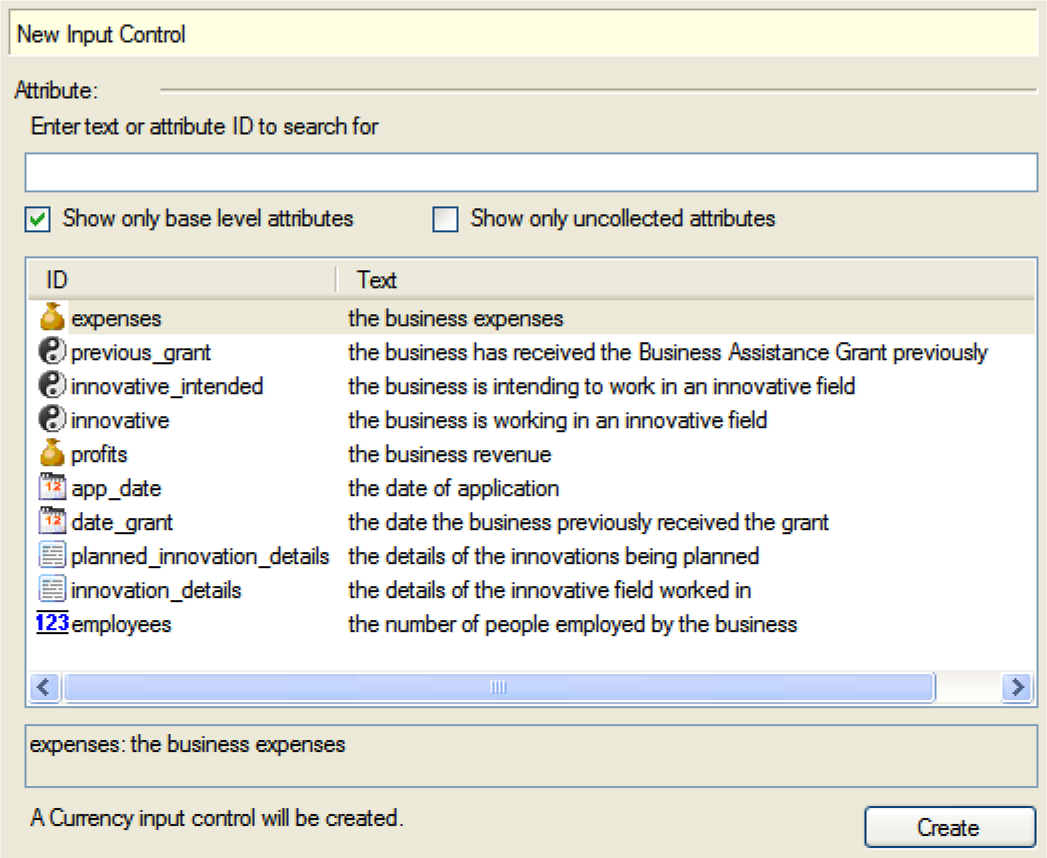
To add a new input (question) control to your screen:

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.



Screen Title: Business Details
Screen Entity: Global
Screen Attribute: <no screen attribute>

The right hand side of the question screen dialog will show the following controls:



Enter text or attribute ID to search for

☑ Show only base level attributes ☐ Show only uncollected attributes

ID	Text
expenses	the business expenses
previous_grant	the business has received the Business Assistance Grant previously
innovative_intended	the business is intending to work in an innovative field
innovative	the business is working in an innovative field
profits	the business revenue
app_date	the date of application
date_grant	the date the business previously received the grant
planned_innovation_details	the details of the innovations being planned
innovation_details	the details of the innovative field worked in
employees	the number of people employed by the business

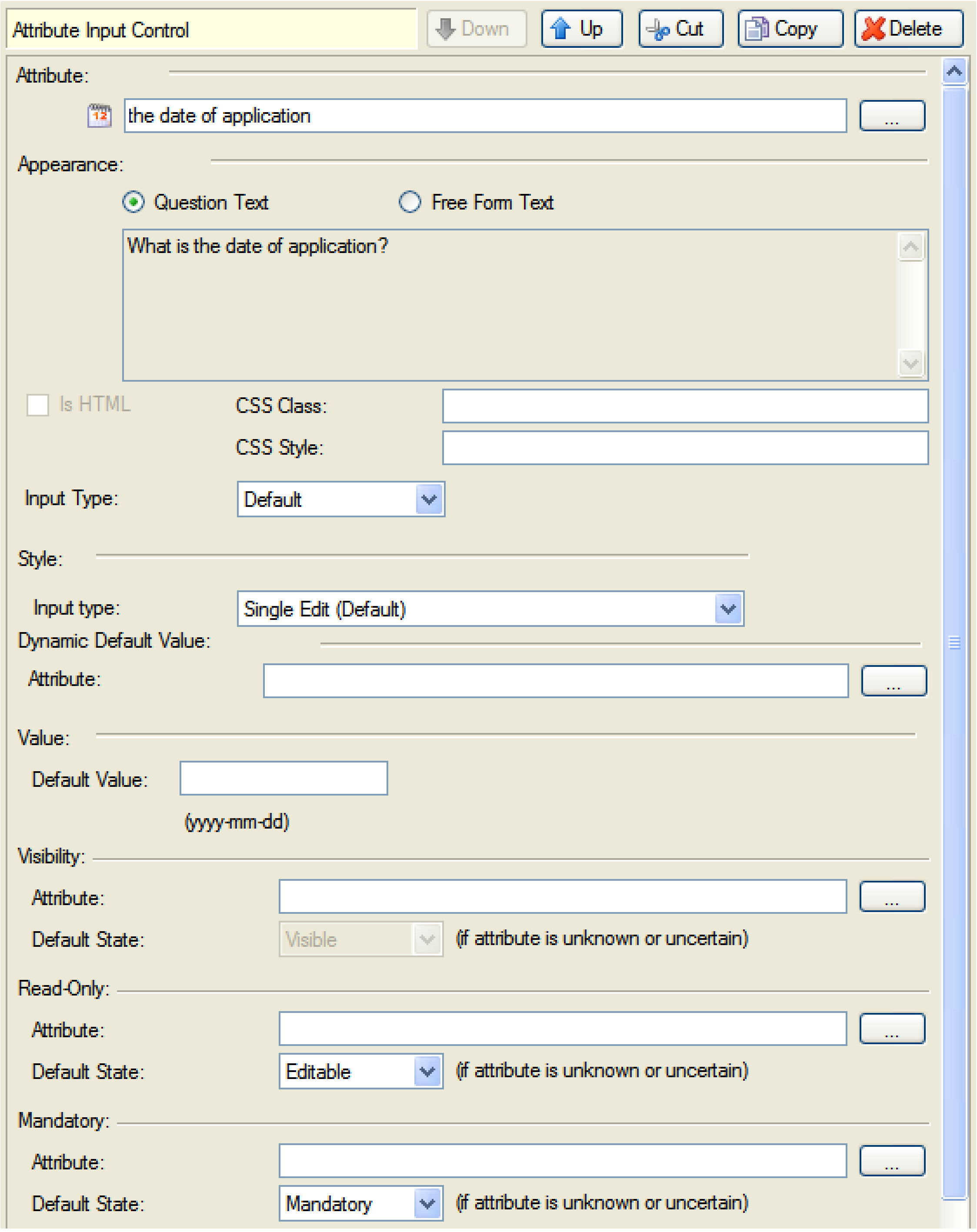
expenses: the business expenses

A Currency input control will be created.

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: the date of application

Appearance: ● Question Text ○ Free Form Text

What is the date of application?

☐ Is HTML

CSS Class:

CSS Style:

Input Type: Default

Style:

Input type: Single Edit (Default)

Dynamic Default Value:

Attribute:

Value:

Default Value:

(yyyy-mm-dd)

Visibility:

Attribute:

Default State: Visible (if attribute is unknown or uncertain)

Read-Only:

Attribute:

Default State: Editable (if attribute is unknown or uncertain)

Mandatory:

Attribute:

Default State: Mandatory (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.

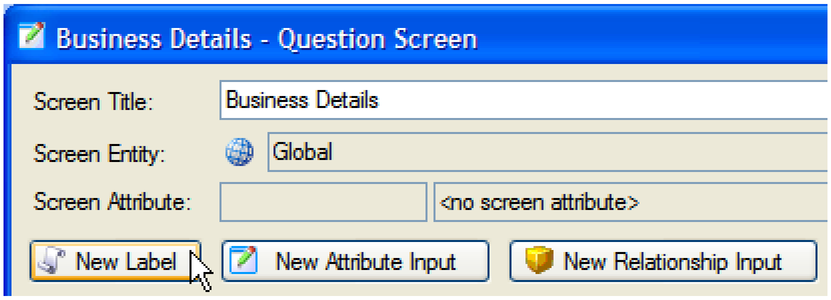
2. Click **OK** to apply any changes you have made to the screen. This will close the screen edit dialog. To save the changes permanently to your screen file, you will need to save it from the main menu in Oracle Policy Modeling.

Add labels to question screens

Labels provide a means for assisting the user to understand the context within which questions are being asked.

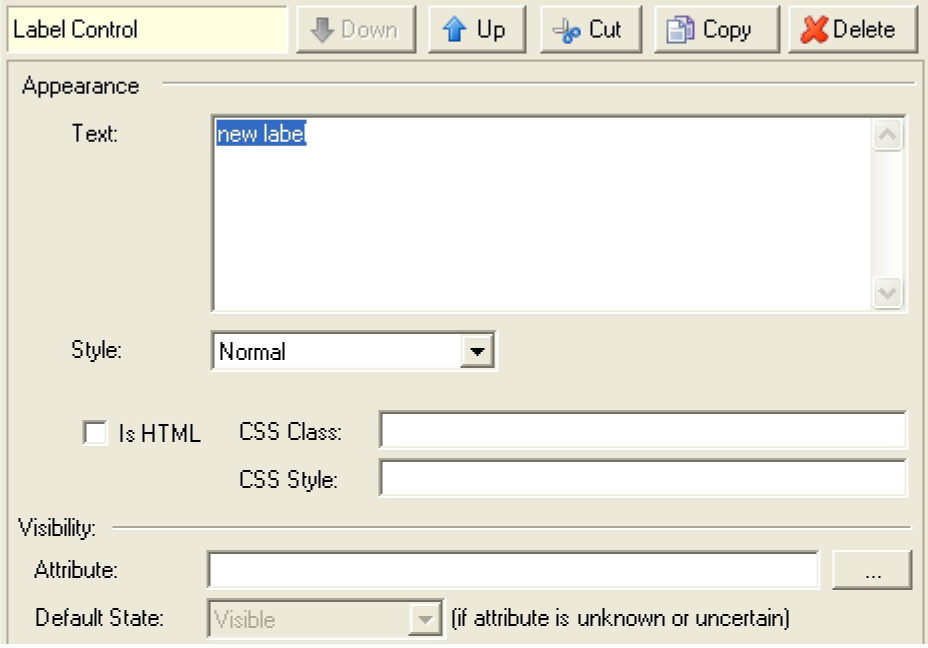
To add a new label to your screen:

1. Open your question screen in your screens file.
2. Click the **New Label** button in the question screen dialog.



Screen Title: Business Details
Screen Entity: Global
Screen Attribute: <no screen attribute>

The right hand side of the question screen dialog will display options for the label.



Text: new label
Style: Normal
☐ Is HTML
CSS Class:
CSS Style:
Attribute:
Default State: Visible (if attribute is unknown or uncertain)

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.

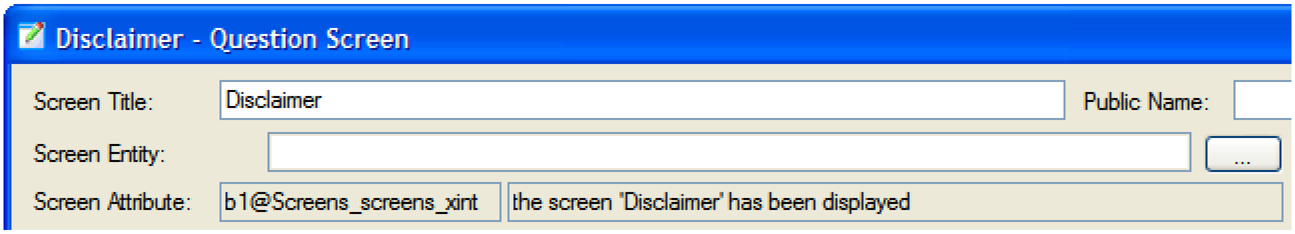
6. If required, enter a **CSS Class** and/or **CSS Style**. See Change the appearance of a control for more information on this setting.

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:



Disclaimer - Question Screen

Screen Title: Disclaimer

Public Name:

Screen Entity:

Screen Attribute: b1@Screens_screens_xint the screen 'Disclaimer' has been displayed

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

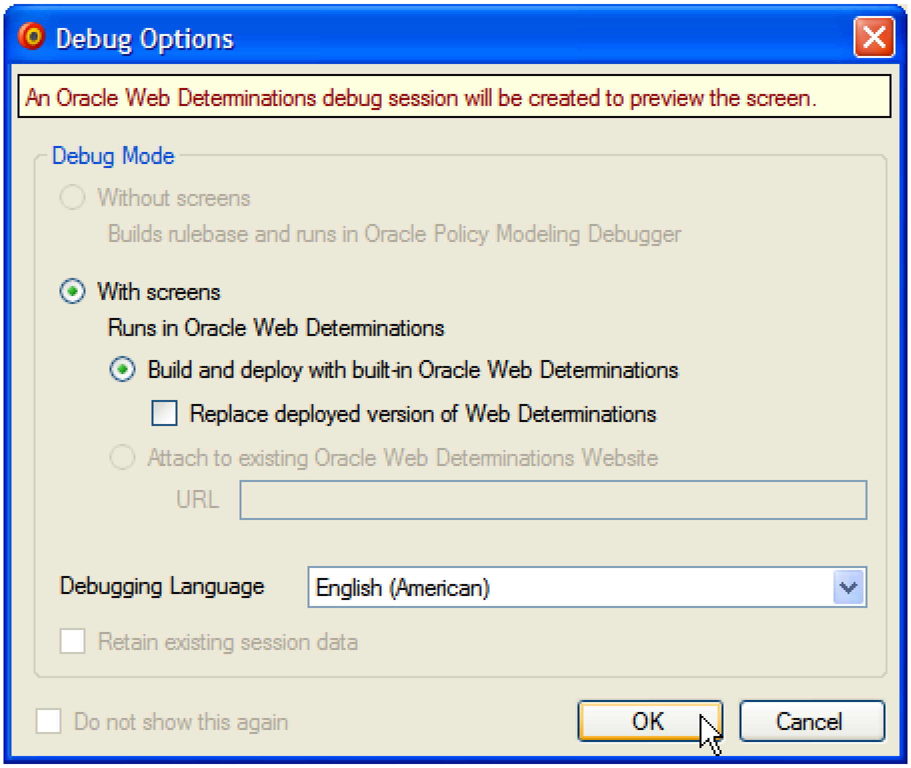
it is known whether or not the claimant is eligible for the benefit

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.**



An Oracle Web Determinations debug session will be created to preview the screen.

Debug Mode

○ Without screens
Builds rulebase and runs in Oracle Policy Modeling Debugger

● With screens
Runs in Oracle Web Determinations

● Build and deploy with built-in Oracle Web Determinations

☐ Replace deployed version of Web Determinations

○ Attach to existing Oracle Web Determinations Website

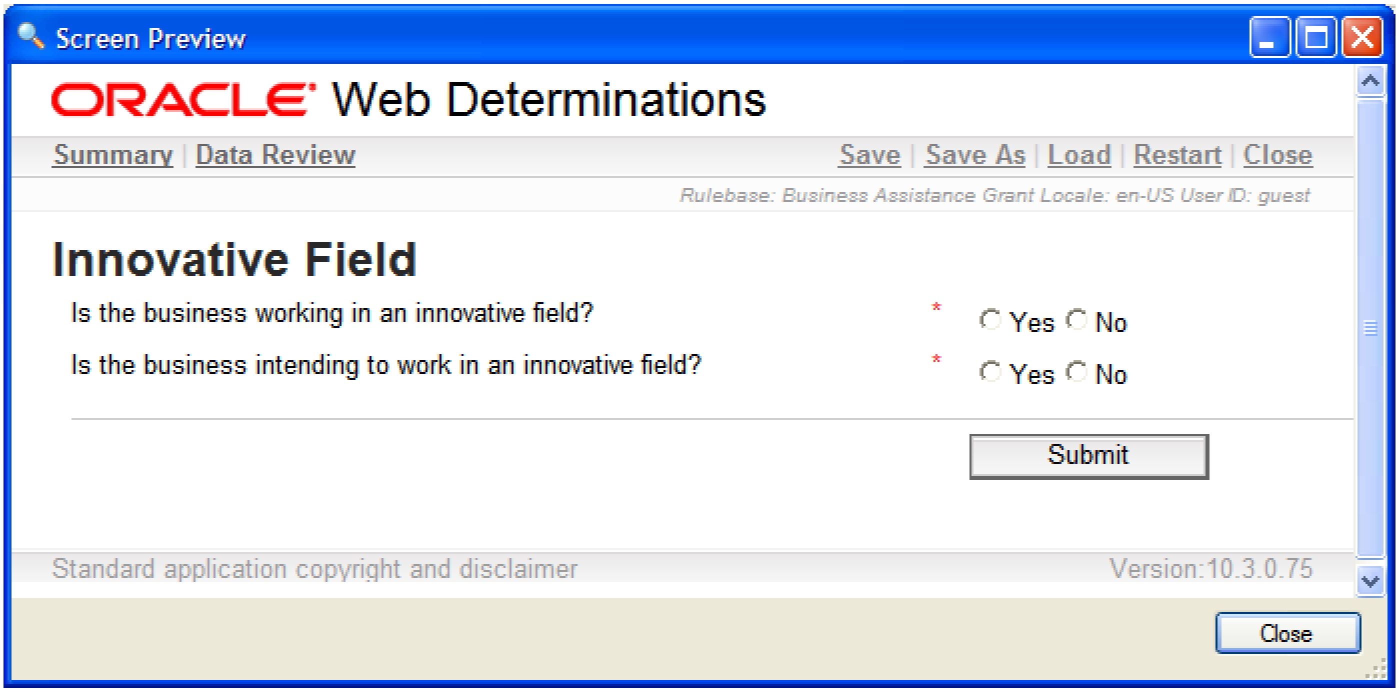
URL

Debugging Language: English (American)

☐ Retain existing session data

☐ Do not show this again

3. 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.



Rulebase: Business Assistance Grant Locale: en-US User ID: guest

Innovative Field

Is the business working in an innovative field? * ○ Yes ○ No

Is the business intending to work in an innovative field? * ○ Yes ○ No

Standard application copyright and disclaimer Version:10.3.0.75

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.

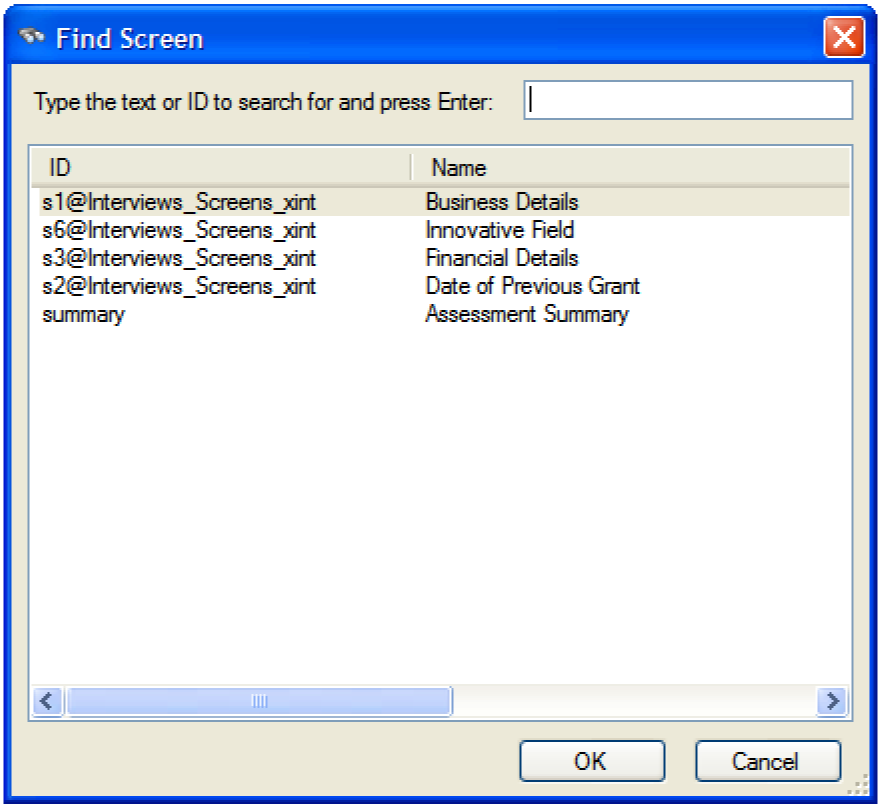
Modify a question screen

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.



ID	Name
s1@Interviews_Screens_xint	Business Details
s6@Interviews_Screens_xint	Innovative Field
s3@Interviews_Screens_xint	Financial Details
s2@Interviews_Screens_xint	Date of Previous Grant
summary	Assessment Summary

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

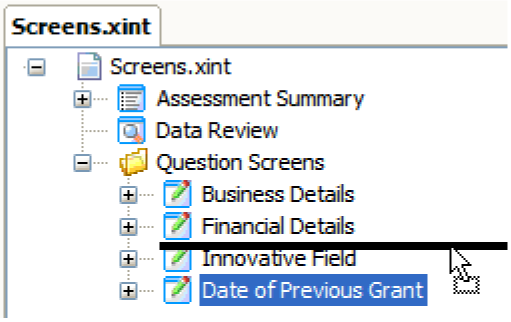
To delete a question screen, select the screen name in your screens file and press **Delete**, or right-click and select **Delete** from the pop-up menu.

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

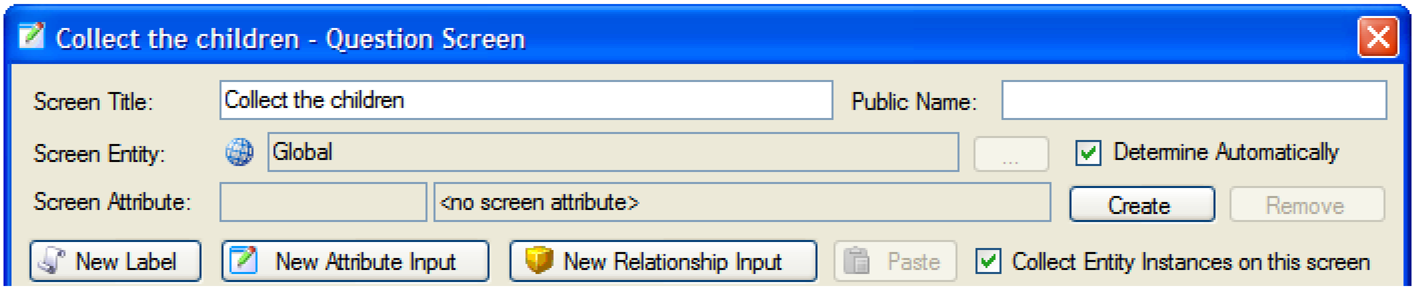
Use substitution on entity screens to identify the entity instance

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: Collect the children

Public Name:

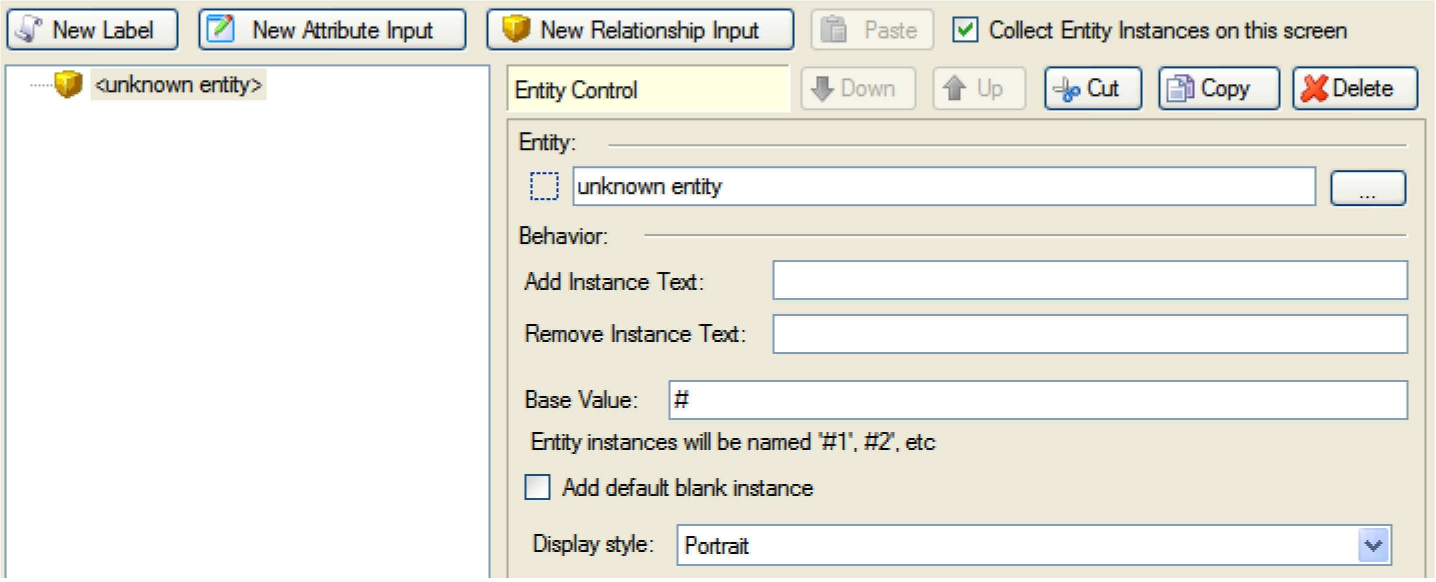
Screen Entity: Global

☑ Determine Automatically

Screen Attribute: <no 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: unknown entity

Behavior:

Add Instance Text:

Remove Instance Text:

Base Value: #

Entity instances will be named '#1', #2', etc

☐ Add default blank instance

Display style: Portrait

4. Click the browse button next to the **Entity** field to open the **Entity Selector** dialog. Select the appropriate entity.

Select an entity to be collected.

the school
the child

the child
Defined in document: Properties.xsrc

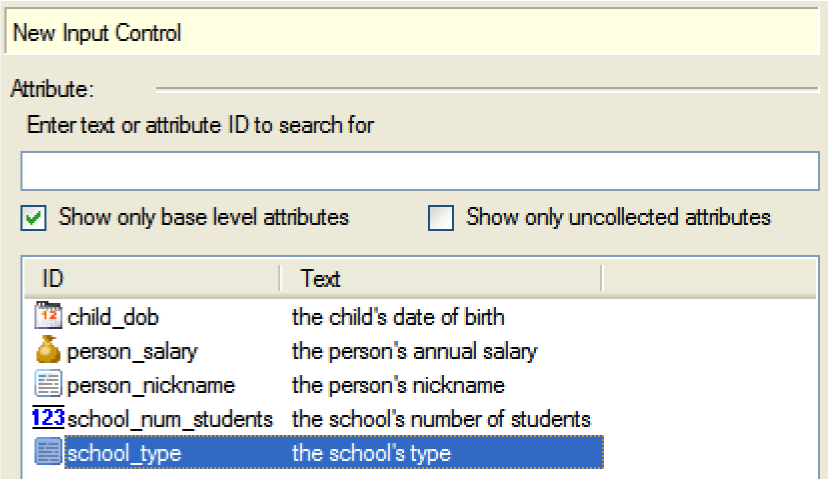
5. Click **OK**. The entity name will now appear in the **Entity** text box in the Entity Control window. The Screen Entity is also shown, based on the containment relationship defined for the entity you have selected. TIP: The entity control is displayed as a cube in the screen control list on the left 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
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.

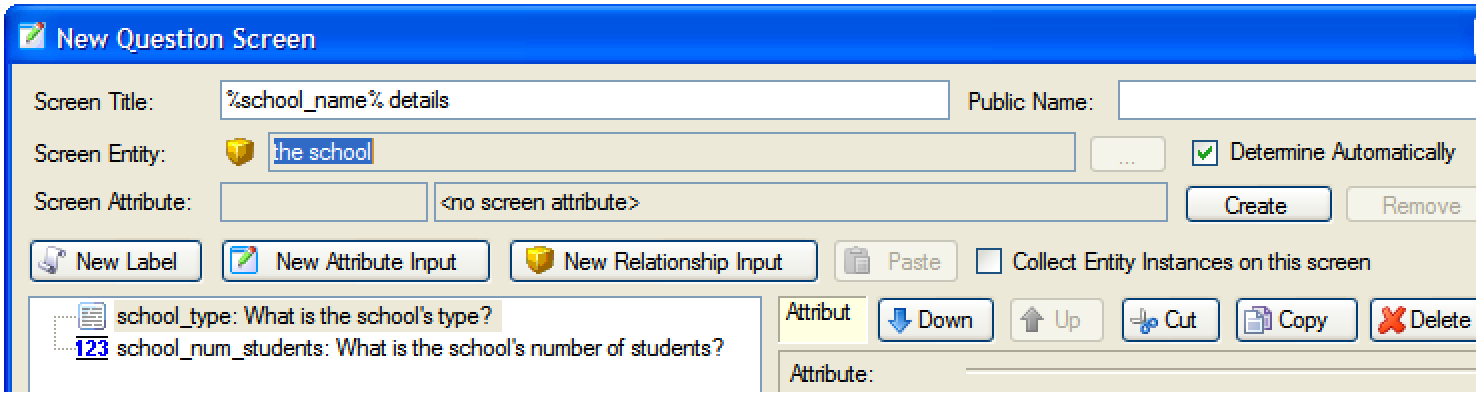
4. Click **Create**.
5. Select the attribute on the left hand side of the screen. The right side of the question screen edit dialog will be replaced with a set of edit controls for the question based on its data type. Here you can customize questions and customize user input options.
6. Click **OK**.

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.



Screen Title: %school_name% details

Public Name:

Screen Entity: the school

☑ Determine Automatically

Screen Attribute: <no screen attribute>

☐ Collect Entity Instances on this screen

school_type: What is the school's type?

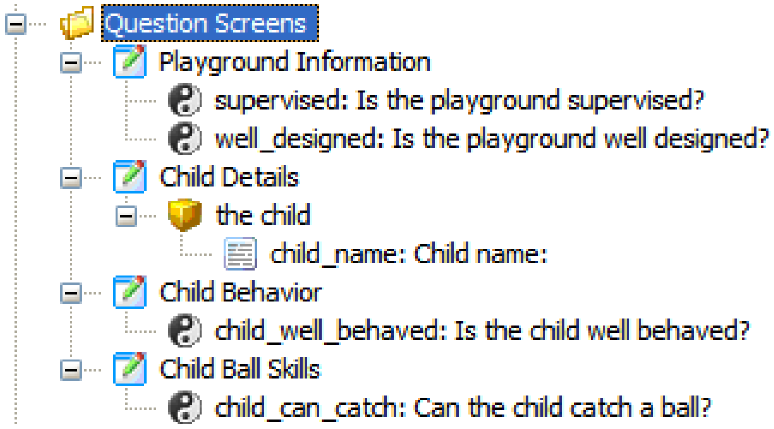
123 school_num_students: What is the school's number of students?

(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

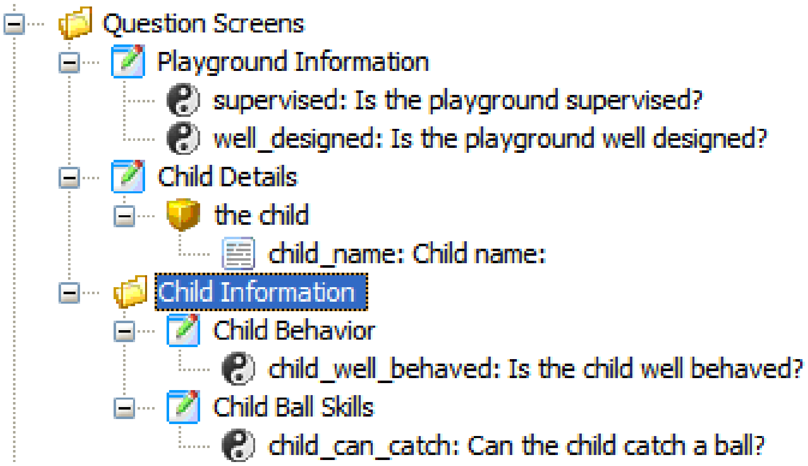
The default behavior of entity question screens is to show a screen for all instances of an entity before moving on to the next screen. For example, the following screen structure:



- Question Screens
  - Playground Information
    - supervised: Is the playground supervised?
    - well_designed: Is the playground well designed?
  - Child Details
    - the child
      - child_name: Child name:
  - Child Behavior
    - child_well_behaved: Is the child well behaved?
  - Child Ball Skills
    - child_can_catch: Can the child catch a ball?

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.

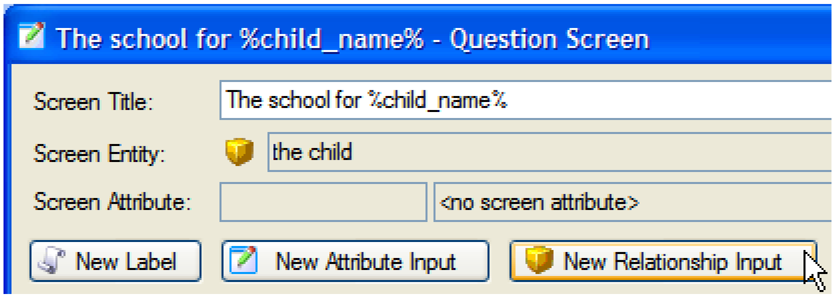
A variable should be substituted in all headings and controls where entity clarification is required. See Substitute an attribute value into the text on screens for how to do this.

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.

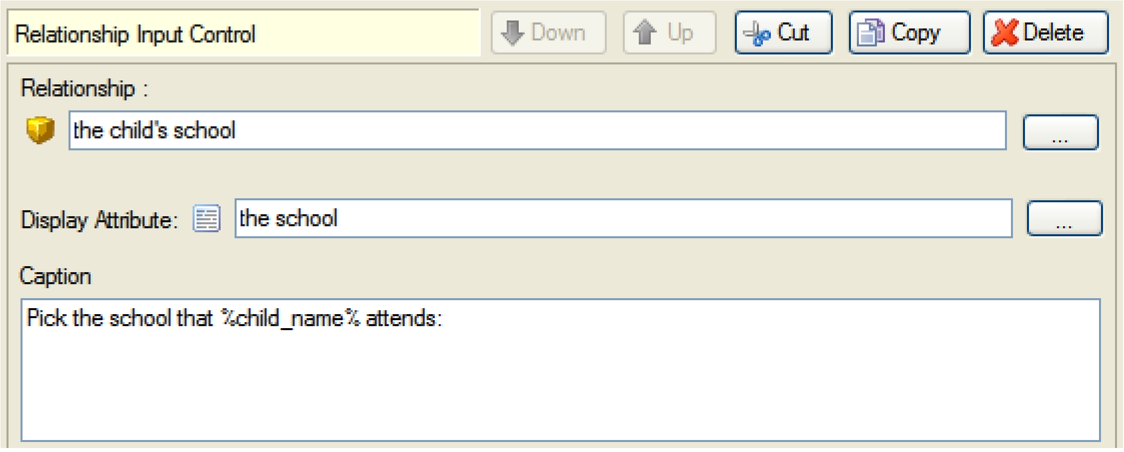

The school for %child_name% - Question Screen

Screen Title: The school for %child_name%

Screen Entity: the child

Screen Attribute: <no screen attribute>

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.


Relationship Input Control

Relationship : the child's school

Display Attribute: the school

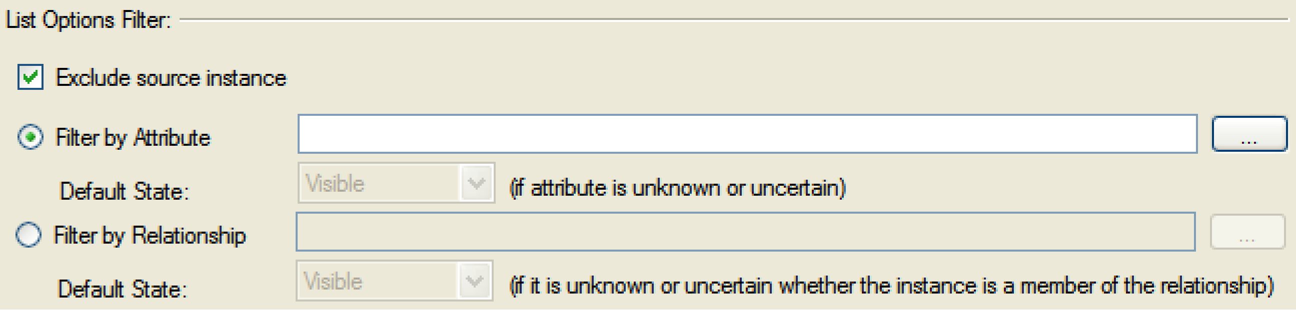
Caption

Pick the school that %child_name% attends:

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:



List Options Filter:

☑ Exclude source instance

● Filter by Attribute

Default State: Visible (if attribute is unknown or uncertain)

○ Filter by Relationship

Default State: Visible (if it is unknown or uncertain whether the instance is a member of the relationship)

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:

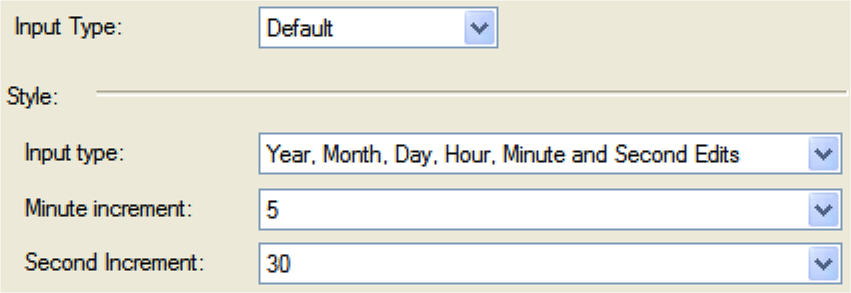
1. In your screens file, double click to open the screen containing the question.
2. Click on the question in the left hand pane to open it for editing in the right hand pane.
3. Select the **Input type** from the drop-down list.
4. Click **OK**.

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: Default

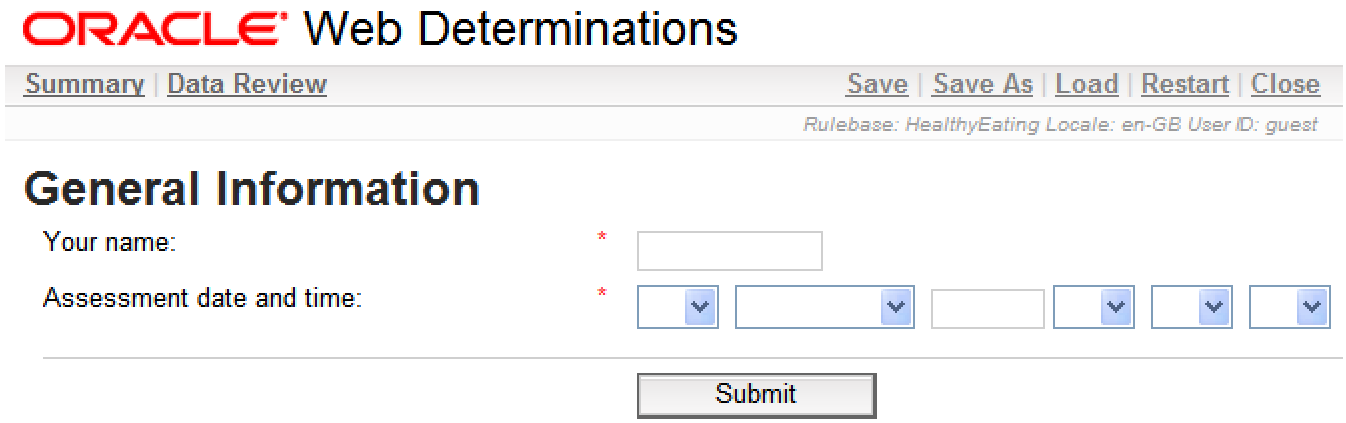
Style:

Input type: Year, Month, Day, Hour, Minute and Second Edits

Minute increment: 5

Second Increment: 30

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.

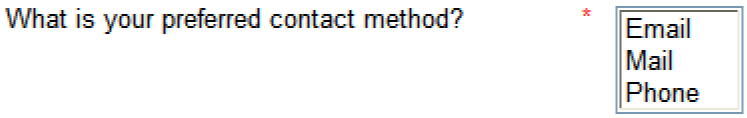


Note that if the **Single Edit** option is chosen for a date/time or time variable, the user can omit seconds and these will be set to 00 by default.

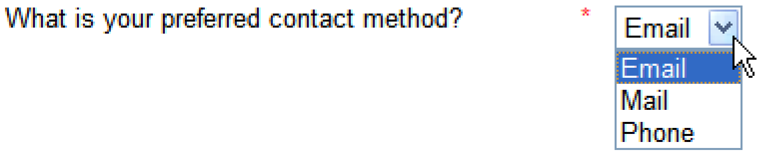
Specify the values for a restrictive input control

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,

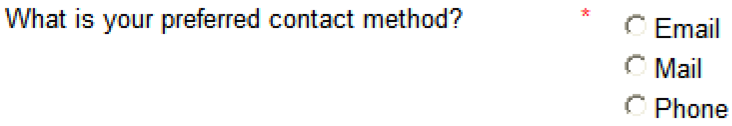
- List box:



- Drop down list:



- Radio buttons:

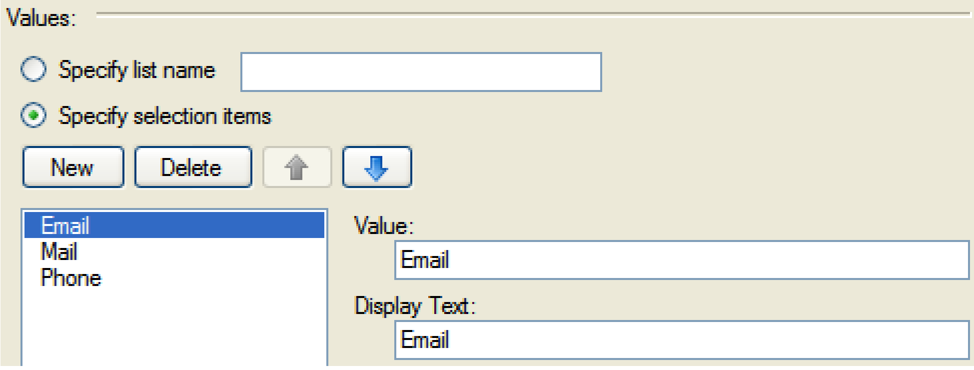


NOTE: By default drop down lists are searchable and configurable. For more information, see Change the appearance of a drop down list in Oracle Web Determinations.

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

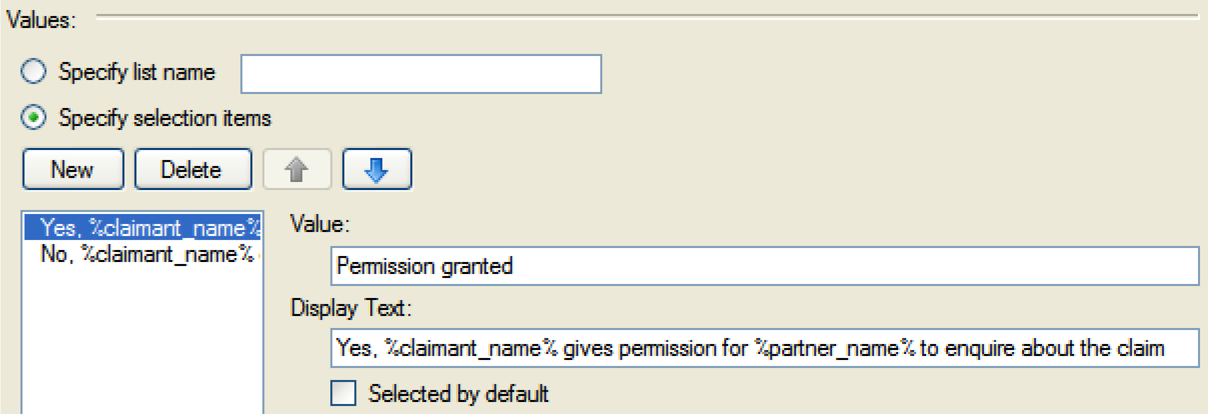
Email
Mail
Phone

Value: Email

Display Text: Email

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

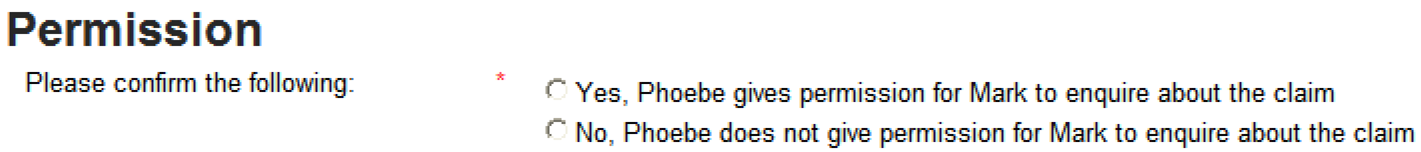
Yes, %claimant_name%
No, %claimant_name%

Value: Permission granted

Display Text: Yes, %claimant_name% gives permission for %partner_name% to enquire about the claim

☐ 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**.

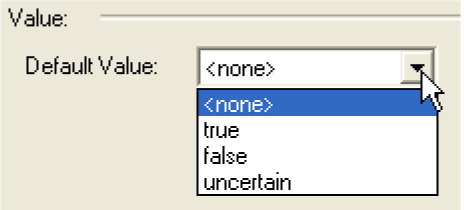
NOTE: A dynamic default attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect, it may be in the screen entity.

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.

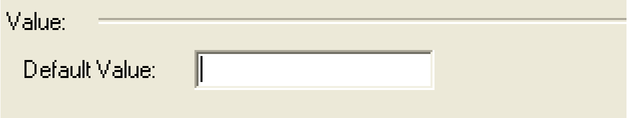
To specify a default value for an input:

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:

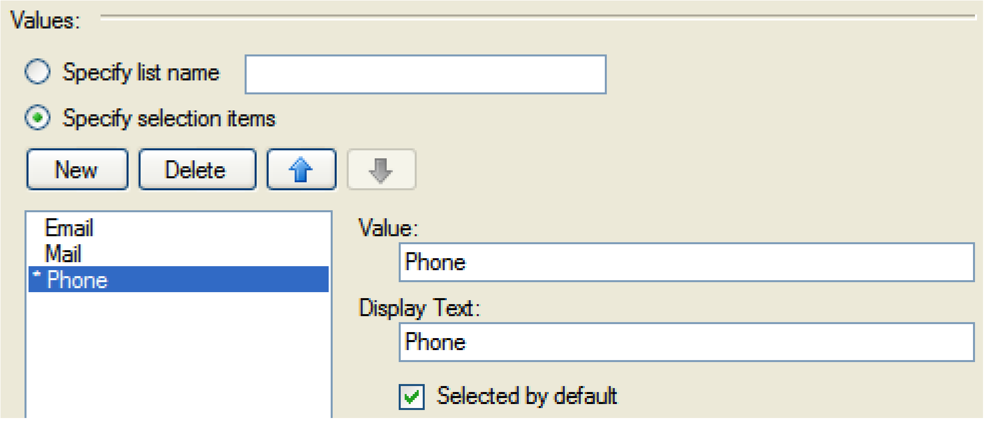
Default Value: <none>



Value:

Default Value:

4. 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: Phone

Display Text: Phone

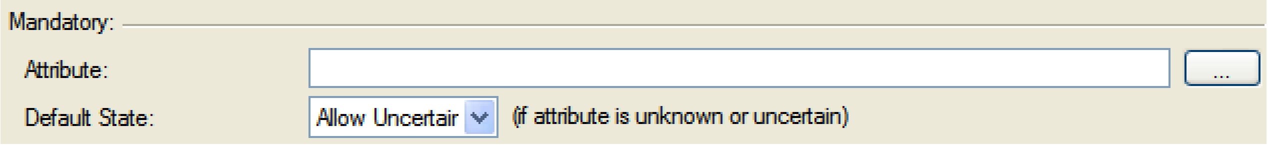
☑ Selected by default

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):

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 **Mandatory** section, select the **Allow Uncertain** option from the drop-down list for **Default State**.



Mandatory:

Attribute:

Default State: Allow Uncertair (if attribute is unknown or uncertain)

4. 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:
   1. Will be treated as the value **Uncertain** for all inputs except text inputs.
   2. 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

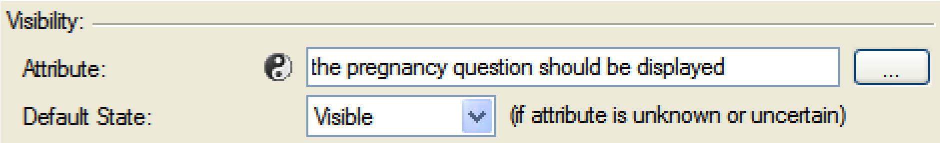
Control the visibility of summary screen elements

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.



Attribute: the pregnancy question should be displayed

Default State: Visible (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.

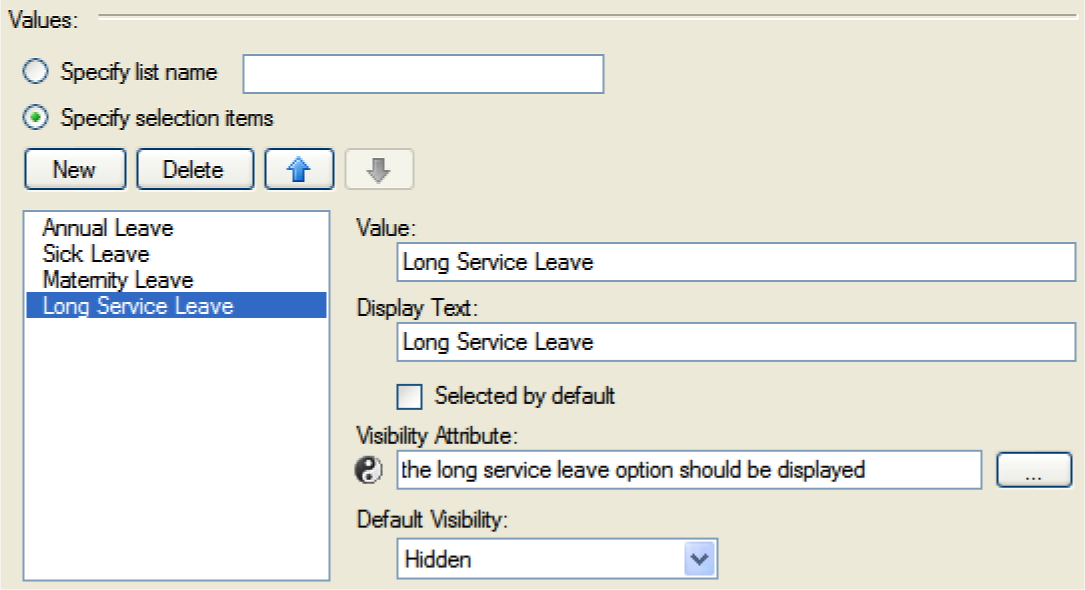
NOTES:

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**.



Values:

○ Specify list name

● Specify selection items

Annual Leave
Sick Leave
Maternity Leave
Long Service Leave

Value: Long Service Leave

Display Text: Long Service Leave

☐ Selected by default

Visibility Attribute: the long service leave option should be displayed

Default Visibility: Hidden

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.

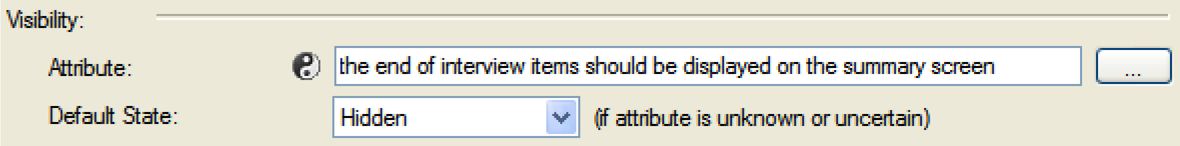
NOTE: A visibility attribute may be in the global entity, in the same entity as the control it is attached to, or if the control is for an entity collect it may be in the screen entity. The value of the visibility attribute can not be temporal.

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: the end of interview items should be displayed on the summary screen

Default State: Hidden (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 and 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:

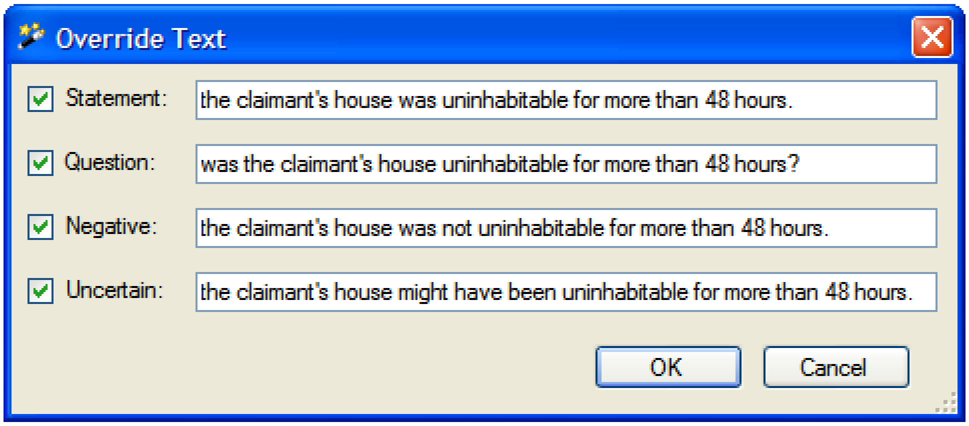
a. when you need to simplify a complex sentence, or
b. in a Rapid Language Support rulebase where the generic form defined in the RLS parser does not create the correct sentence text. For more information on using an RLS parser, and changing individual sentence forms in such a project, see the Help available 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 sentences 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.



☑ Statement: the claimant's house was uninhabitable for more than 48 hours.

☑ Question: was the claimant's house uninhabitable for more than 48 hours?

☑ Negative: the claimant's house was not uninhabitable for more than 48 hours.

☑ Uncertain: the claimant's house might have been uninhabitable for more than 48 hours.

5. Select **OK** in the Override Text dialog box and then **OK** in the Attribute Editor dialog box.

Customize question text

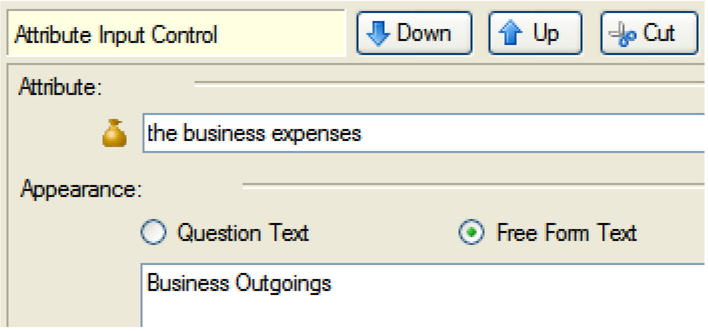
Not all question forms generated by Oracle Policy Modeling's automatic parsing of attributes are suitable for the required purpose. In many cases, rather than asking a question in question form, you might want to abbreviate this to a label for the text field.

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.



Attribute: the business expenses

Appearance: ○ Question Text ● Free Form Text

Business Outgoings

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 **the client company's** income in the relevant tax year. becomes Parker Incorporated expended more than thirty percent of **its** 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 Collect the gender of a person below.	The client expended more that thirty percent of **the client's income** in the relevant tax year. becomes He expended more that thirty percent of **his** 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 **the man's** knee. becomes Bob has had surgery to **his** knee.
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.	The client is in the second trimester of **the client's** pregnancy. becomes Gillian is in the second trimester of **her** 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

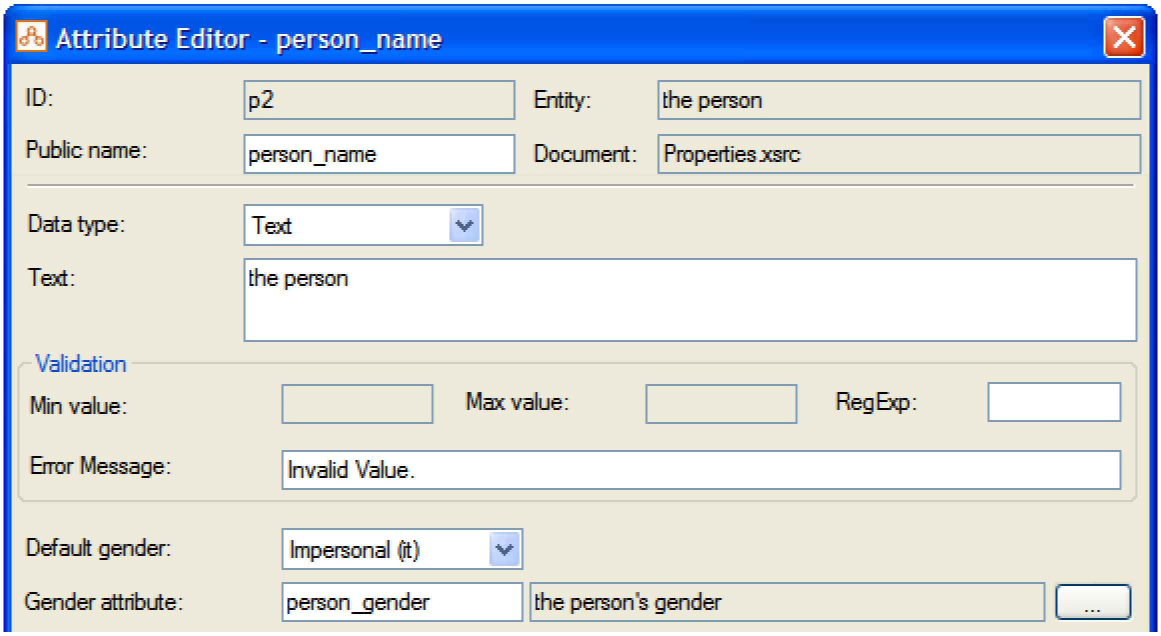
A gender collect control is used to provide a means for collecting whether or not a person is male or female. This is done by adding a gender attribute as a control on a screen. Essentially, the gender attribute is a text type variable, and the control will set it to a value of either "male" or "female".

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.



ID: p2
Entity: the person
Public name: person_name
Document: Properties.xsrc
Data type: Text
Text: the person
Min value:
Max value:
RegExp:
Error Message: Invalid Value.
Default gender: Impersonal (it)
Gender attribute: person_gender the person's gender

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").

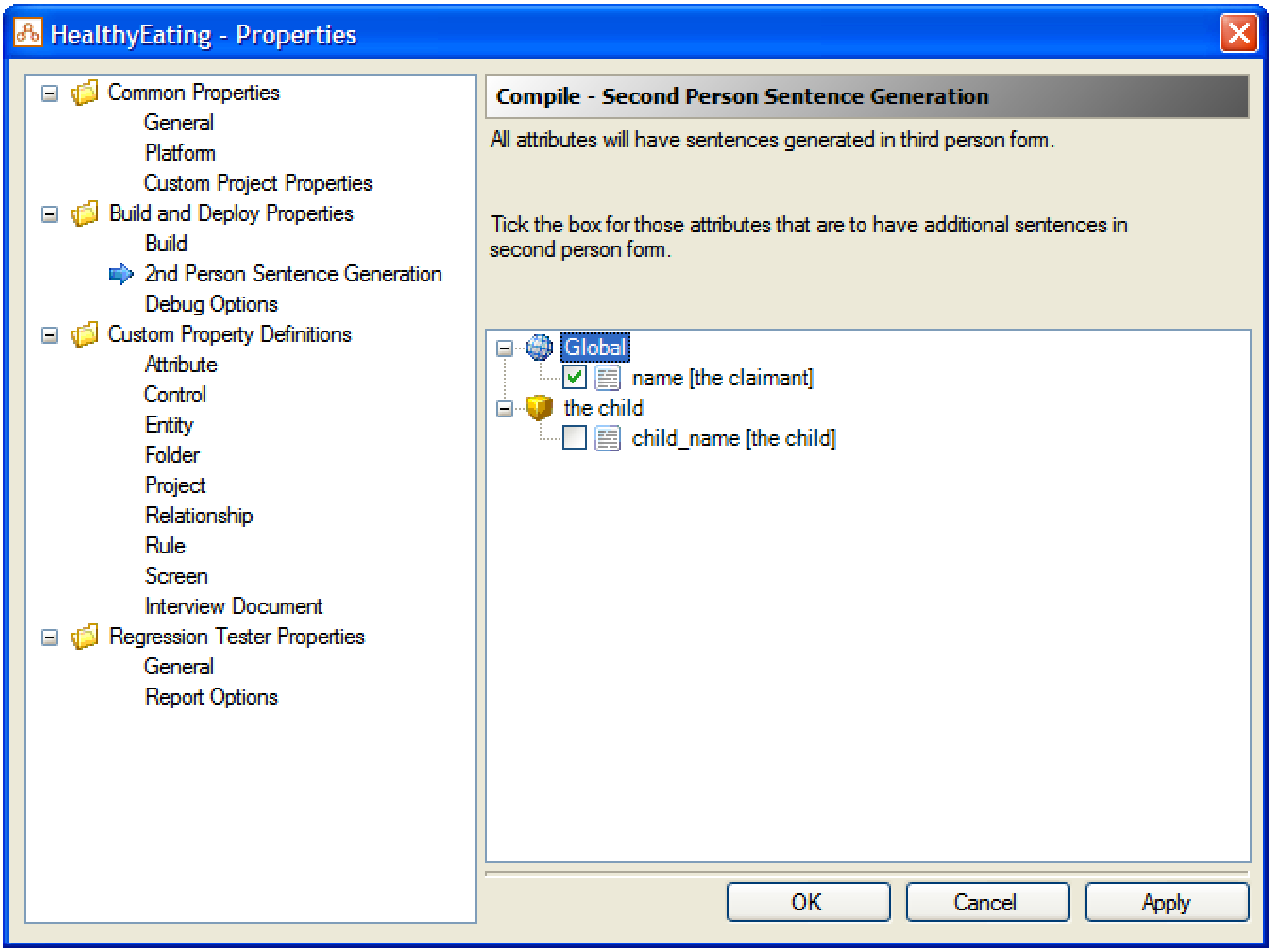
For example, you may wish to define your screen title to read "Jane and her assets" (for an interview involving a person Jane who is female). You would do this with a screen title "%person% and %person:his/her/its% assets".

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.



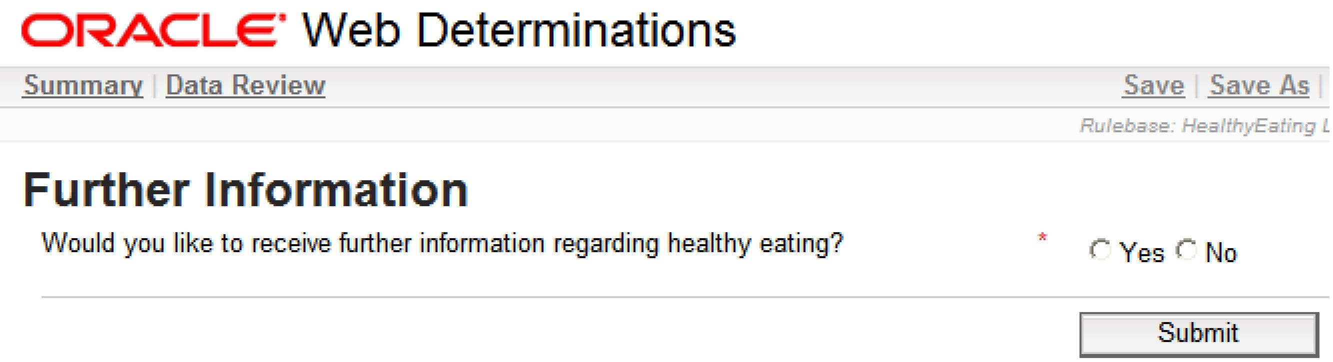
Compile - Second Person Sentence Generation

All attributes will have sentences generated in third person form.

Tick the box for those attributes that are to have additional sentences in second person form.

- Global
  - ☑ name [the claimant]
- the child
  - ☐ child_name [the child]

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.



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

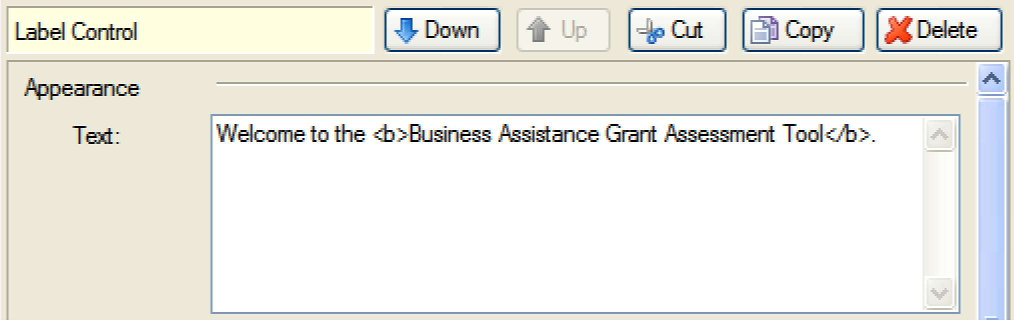
Change the appearance of text

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/.

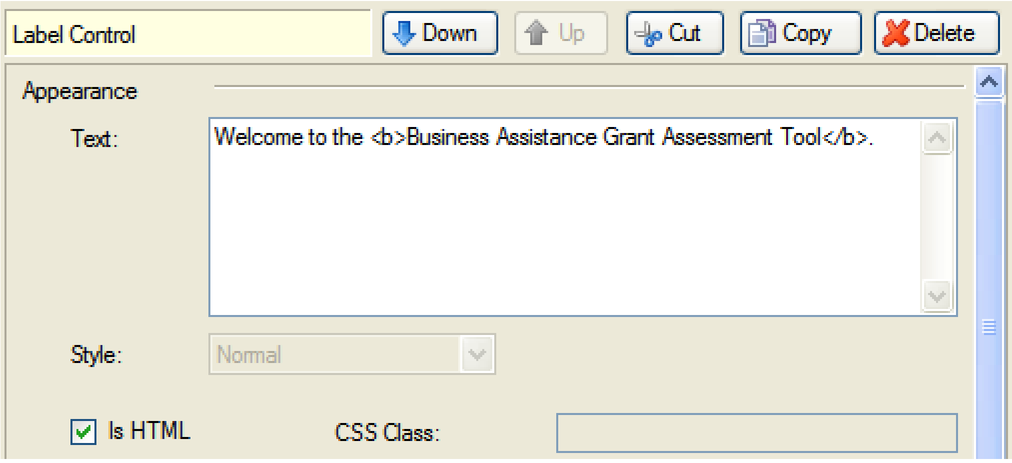
To change the appearance of text using HTML tags:

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.



Text: Welcome to the Business Assistance Grant Assessment Tool.

4. Select the **Is HTML** checkbox.



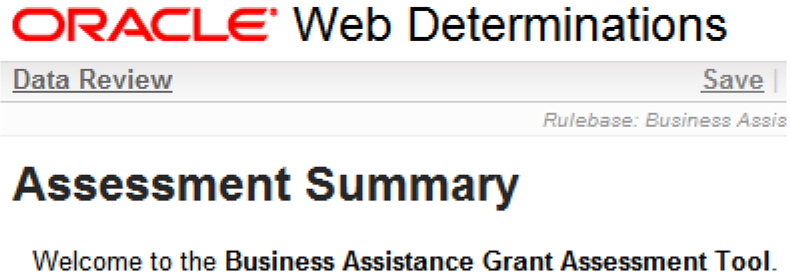
Text: Welcome to the Business Assistance Grant Assessment Tool.

Style: Normal

☑ Is HTML

CSS Class:

5. Click **OK.** In Web Determinations this would be rendered as:



Welcome to the **Business Assistance Grant Assessment Tool**.

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
```

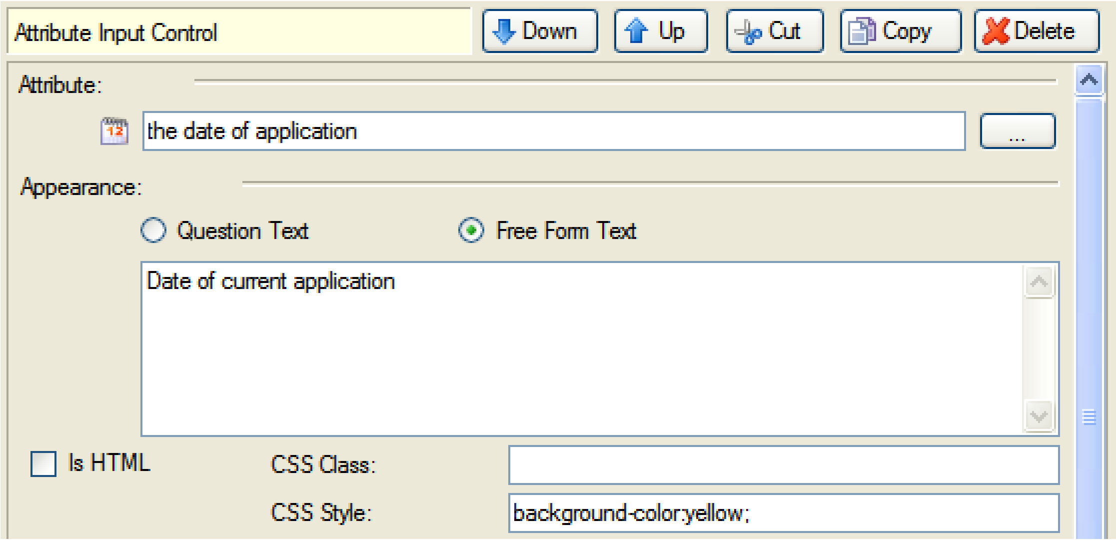
If a tag is used in a screens file that is not in this list, the rulebase will not load in Web Determinations. If necessary, this HTML validation can be turned off by setting the **screens.validate.html** setting in the file above to "false".

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,



Attribute Input Control

Attribute: the date of application

Appearance:

○ Question Text ● Free Form Text

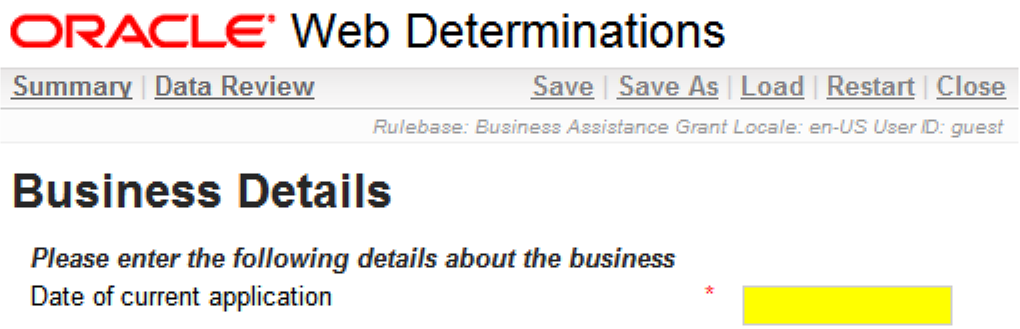
Date of current application

☐ Is HTML

CSS Class:

CSS Style: background-color:yellow;

4. Click **OK.** In Web Determinations this would be rendered as:



Rulebase: Business Assistance Grant Locale: en-US User ID: guest

Business Details

Please enter the following details about the business

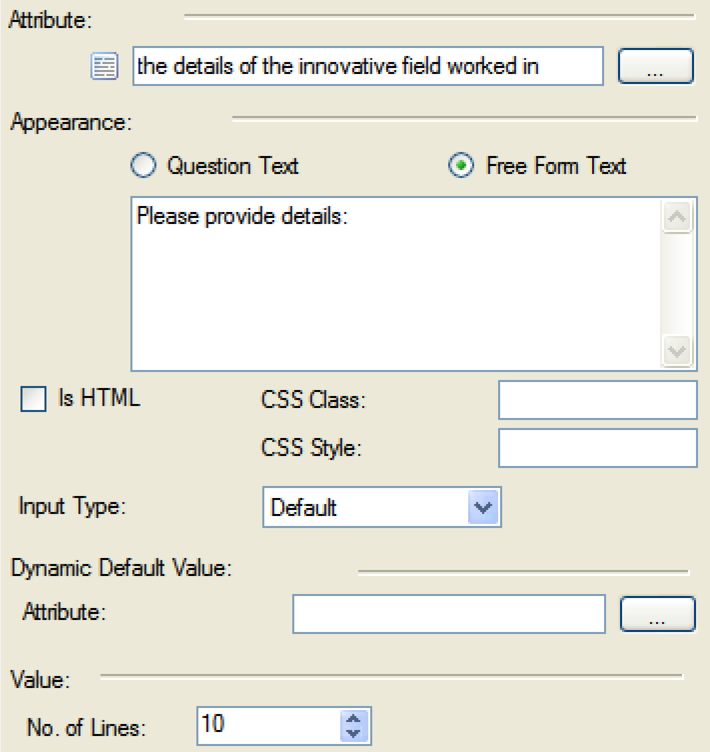
Date of current application *

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.



Attribute: the details of the innovative field worked in

Appearance: ○ Question Text ● Free Form Text

Please provide details:

☐ Is HTML

CSS Class:

CSS Style:

Input Type: Default

Dynamic Default Value:

Attribute:

Value:

No. of Lines: 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. <img src="../../../images/<file>"> (the format for adding images to the summary screen)
    For example, <img src="../../../images/button.jpg">

    ii. <img src="../../../../images/<file>"> (the format for adding images to question screens)
    For example, <img src="../../../../images/button.jpg">

5. Select the **Is HTML** checkbox.
6. Click **OK**.
7. Copy the image file to \Release\web-determinations\WEB-INF\classes\images.

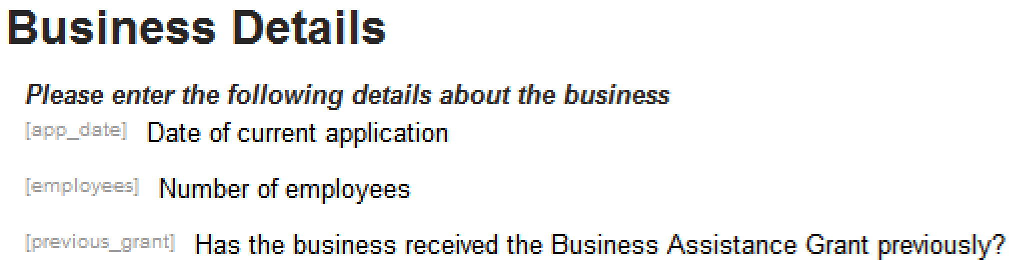
Show/hide features used for debugging

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:

Rulebase: Web Determinations Locale: en-US User ID: guest Case ID: Test Case 1

- 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":

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:

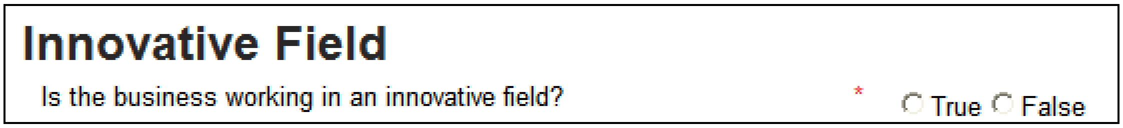
1. Open the messages.<locale>.properties file using Notepad.
2. Edit the 'application-name' configuration line, eg
   ```
   application-name = Income Assistance
   ```
3. Save the file.
4. Build and run the rulebase to view the new text in the Web Determinations banner.

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:



Innovative Field

Is the business working in an innovative field? * ○ True ○ False

Here the out-of-the-box label text for the boolean answers "Yes" and "No", has been replaced with "True" and "False" respectively.

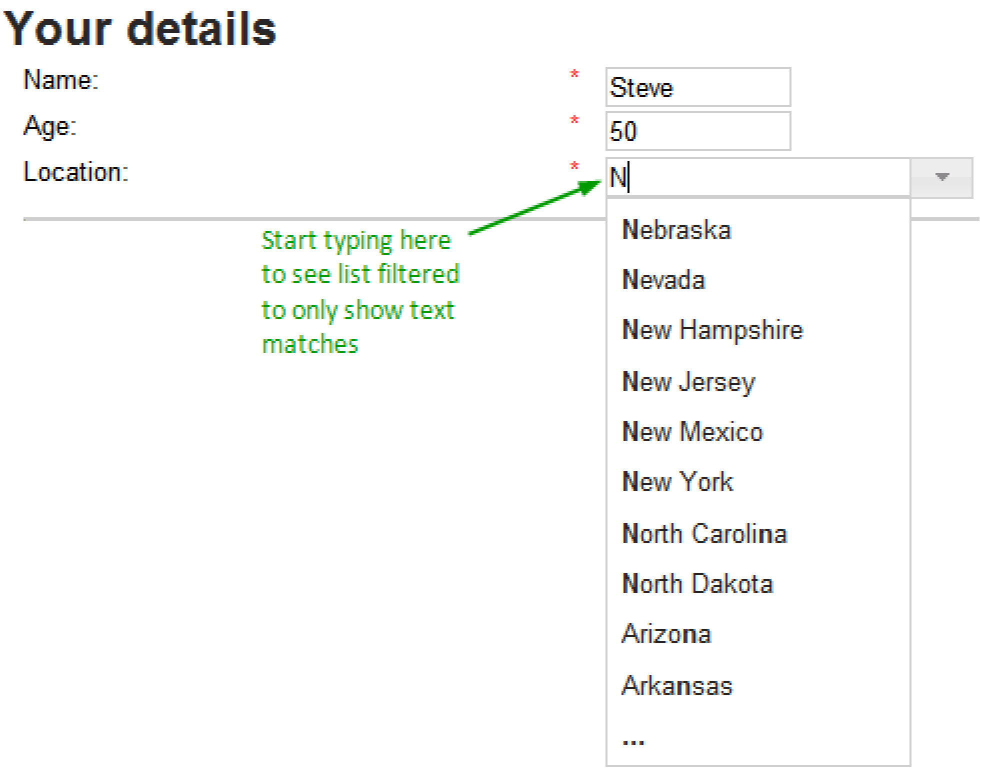
To change the text of a label:

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: * Steve

Age: * 50

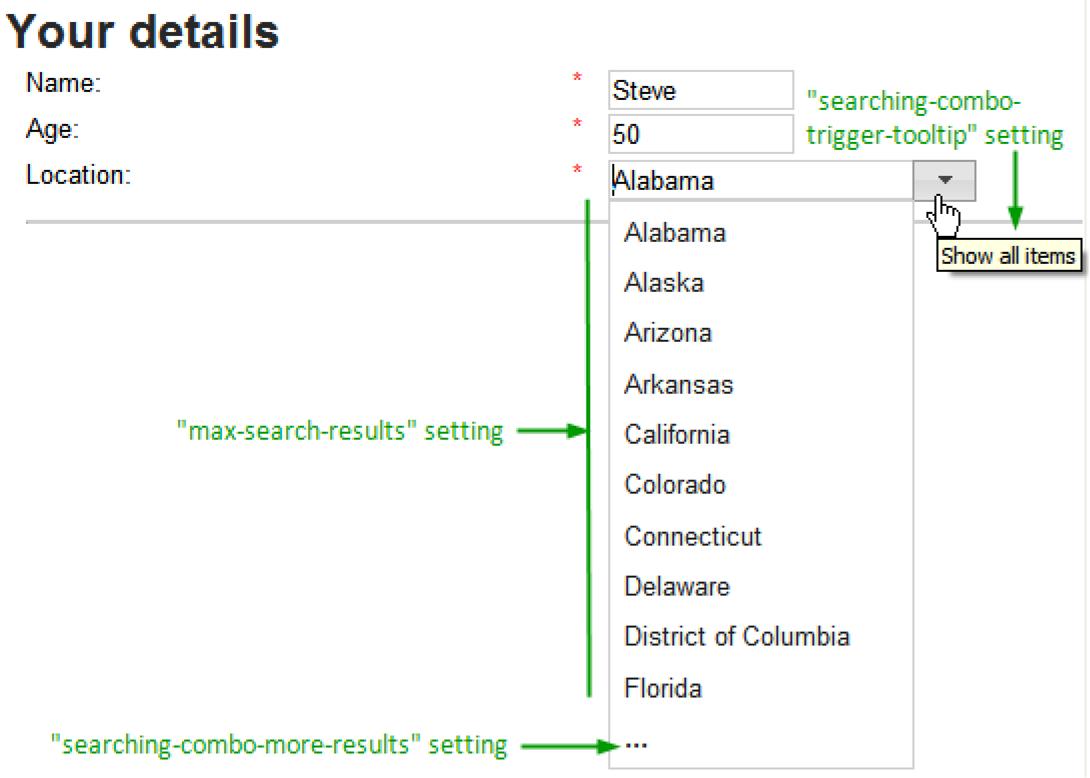
Location: * N

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.



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

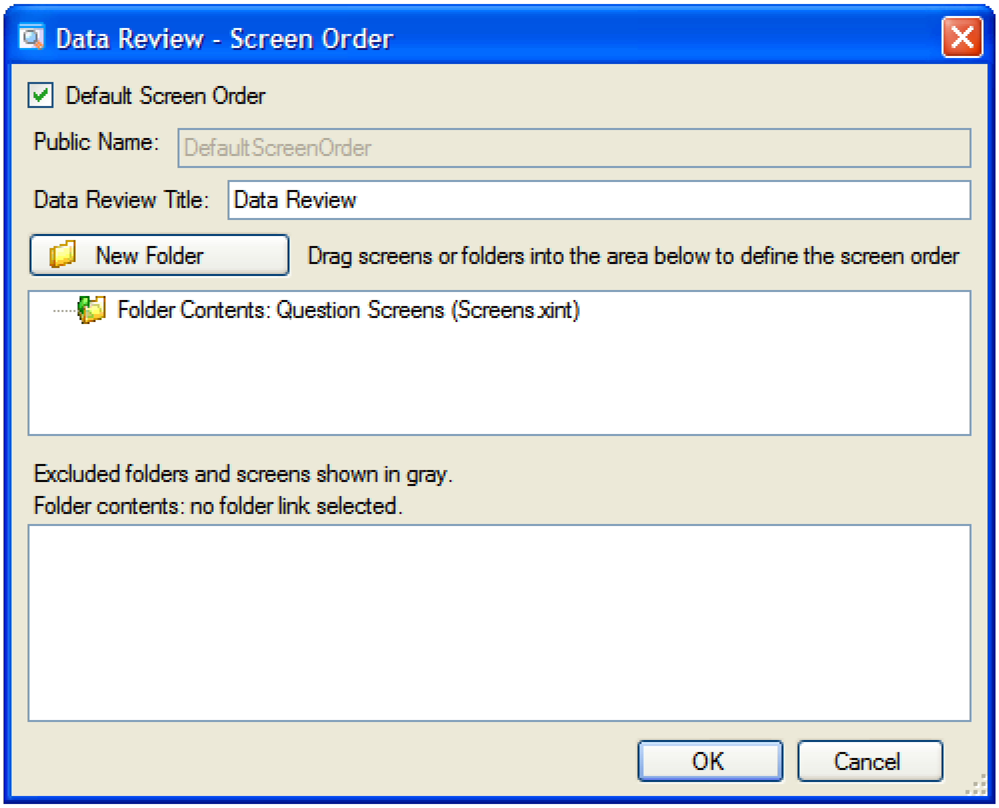
Create a new screen order

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.



☑ Default Screen Order

Public Name: DefaultScreenOrder

Data Review Title: Data Review

Drag screens or folders into the area below to define the screen order

Folder Contents: Question Screens (Screens.xint)

Excluded folders and screens shown in gray.
Folder contents: no folder link selected.

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.

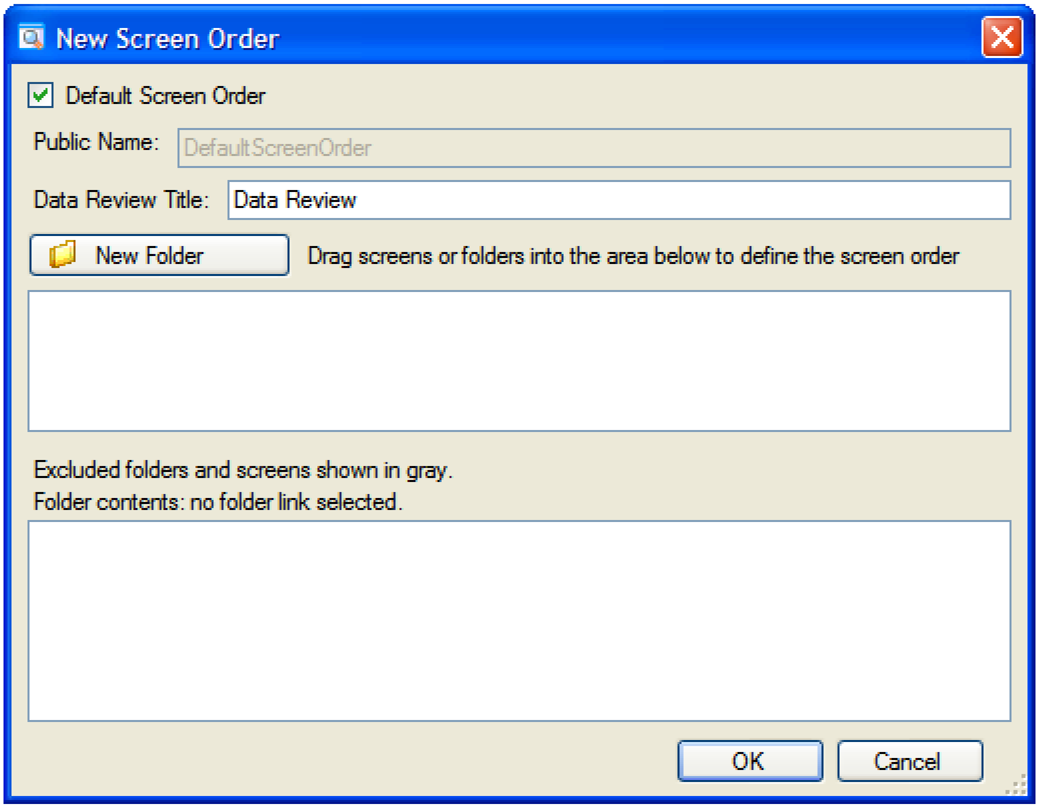
Note that this default behavior only applies to the first screens file added to a project. Any subsequent screens files added to the project will need to have a screen order added manually (see below).

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

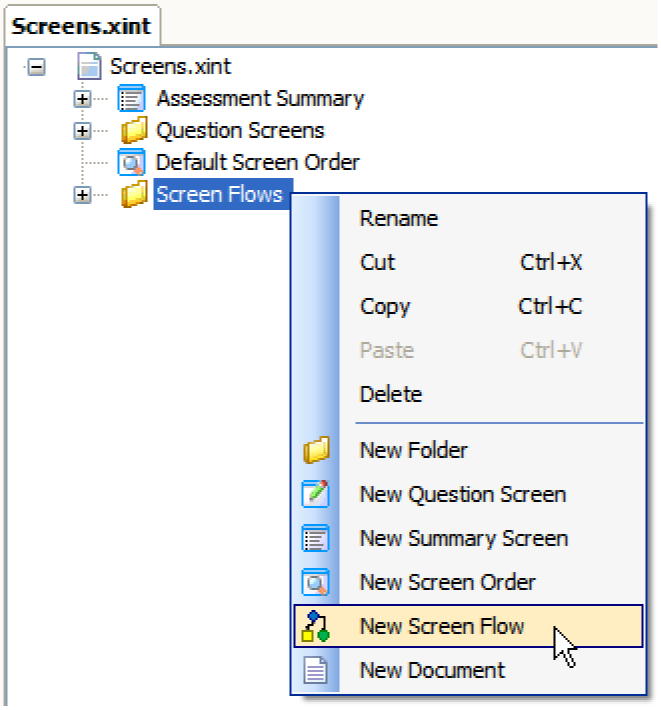
Create a new screen flow

Before you create a screen flow, first create a folder in which to store the flow:

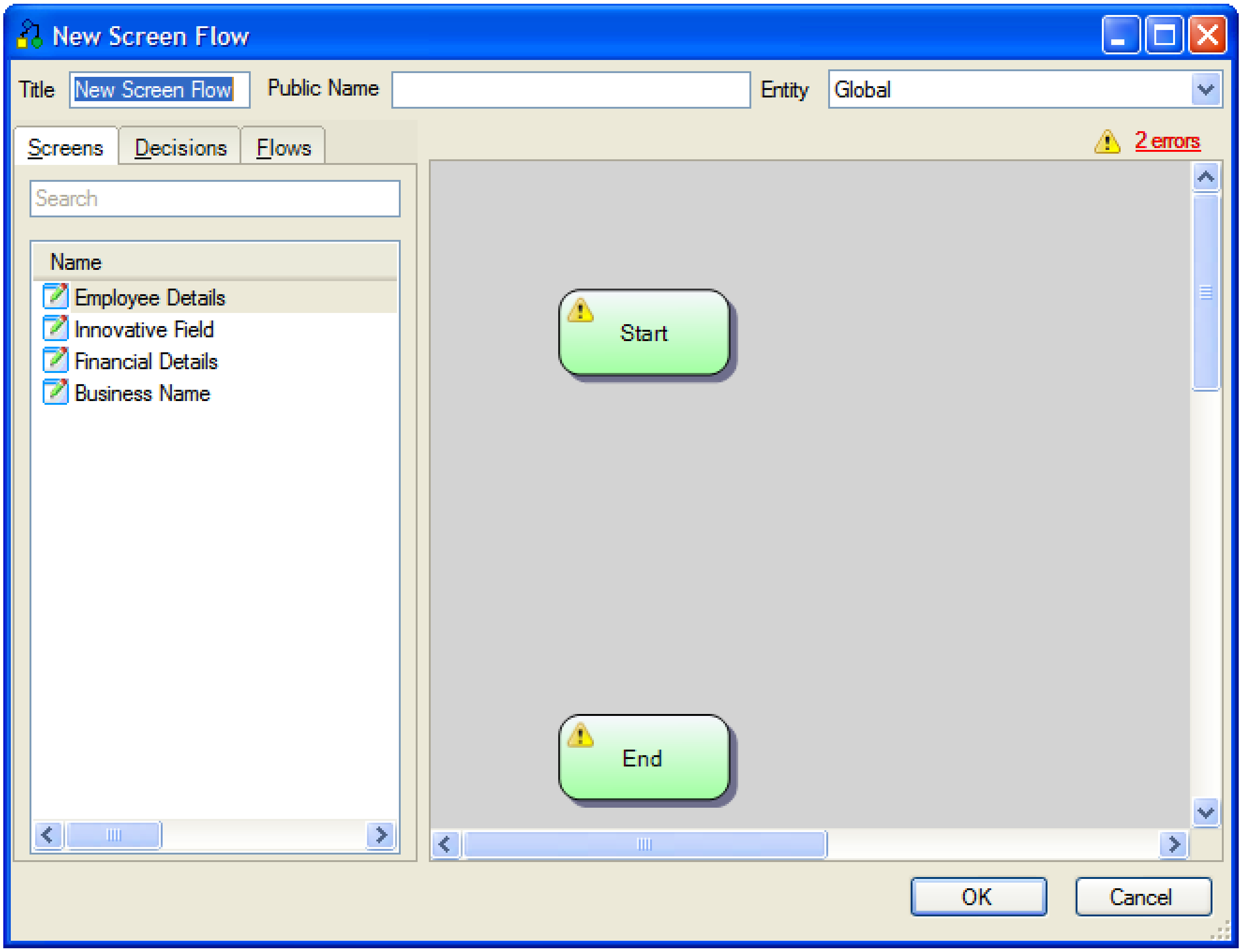
1. In your screens file, right-click the *.xint filename in the screens view.
2. Select **New Folder** from the pop-up menu.
3. Enter an appropriate name for your folder (eg "Screen Flows").

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.


Title: Business Assistance Grant

Public Name:

Entity: Global

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).

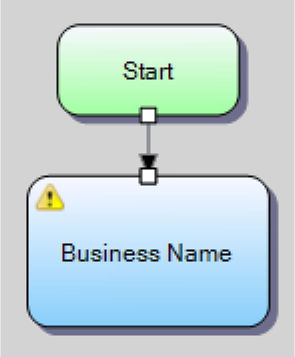

Title: Business Assistance Grant

Public Name:

Entity: Global

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.

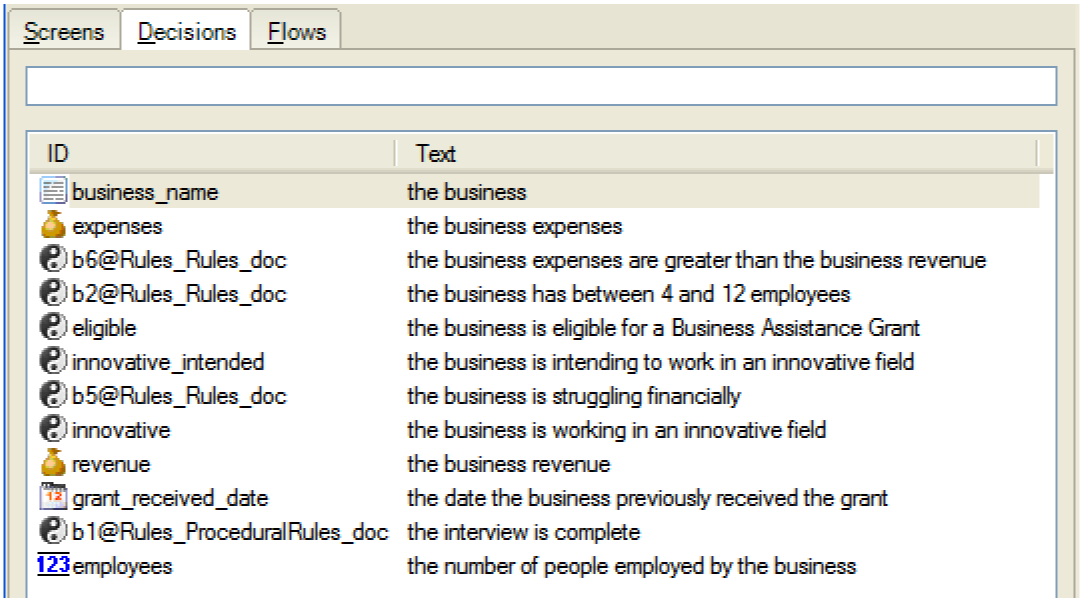


Add decision points to the screen flow

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.

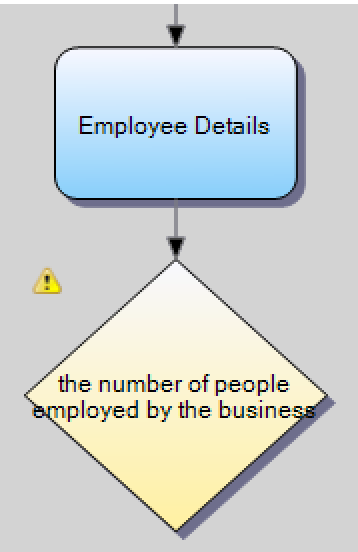
To add a decision point to your flow:

1. Click the **Decisions** tab in the **Screen Flow** dialog box.

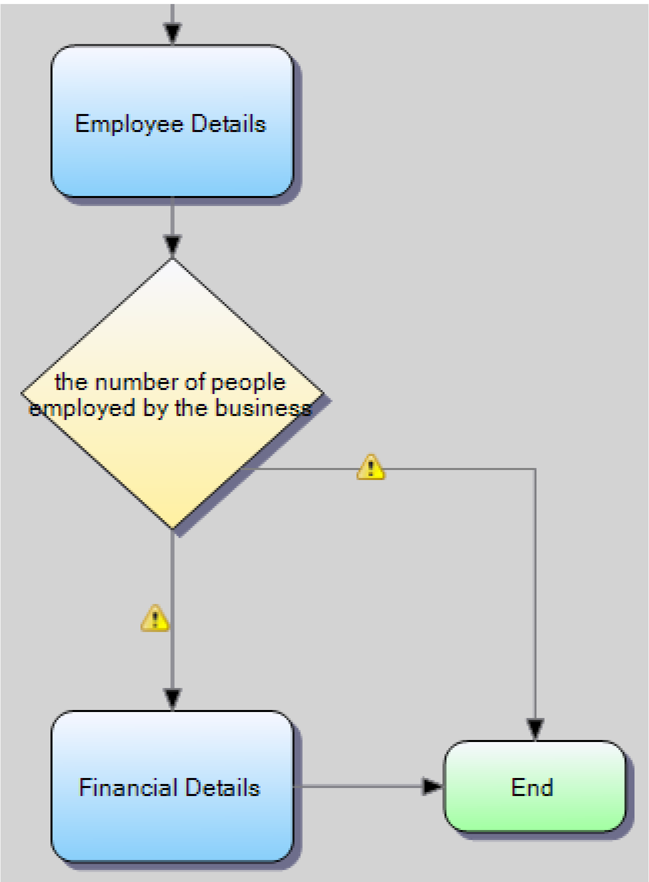


ID	Text
business_name	the business
expenses	the business expenses
b6@Rules_Rules_doc	the business expenses are greater than the business revenue
b2@Rules_Rules_doc	the business has between 4 and 12 employees
eligible	the business is eligible for a Business Assistance Grant
innovative_intended	the business 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
123 employees	the number of people employed by the business

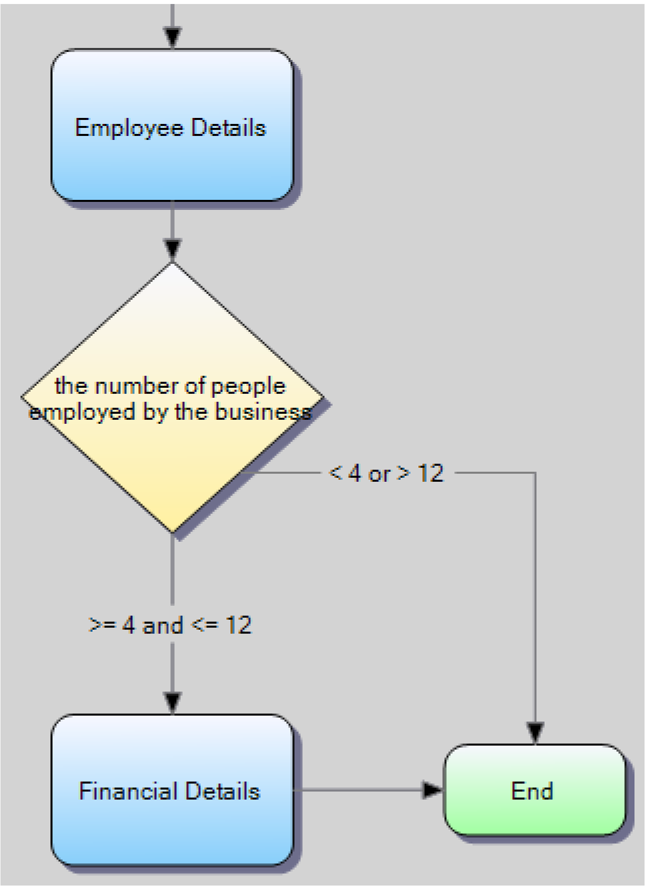
2. Select the attribute you want to add to your flow. You can search or filter the list by entering part of the attribute text or ID in the Search text box.
3. Drag the attribute to just under the shape that it should flow from in the diagram.



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:

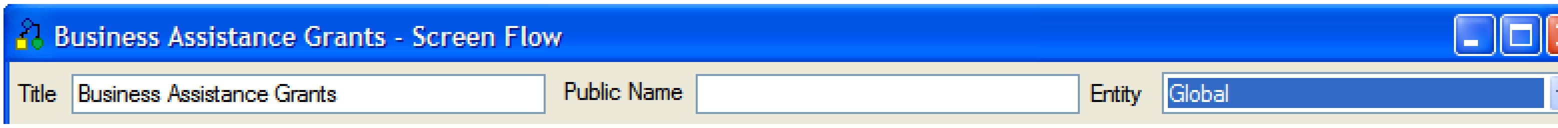
a. When a screen flow is executed, if it reaches a decision shape where the attribute is unknown, it will trigger an investigation of that goal using the default screen order. If an attribute is required that doesn't appear in the screen order, then it will use the question search to investigate the 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.

b. 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.

c. 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.


Business Assistance Grants - Screen Flow

Title Business Assistance Grants

Public Name

Entity Global

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.

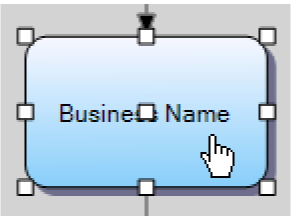
Edit a screen flow

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.

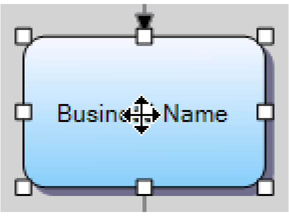
Move a shape

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

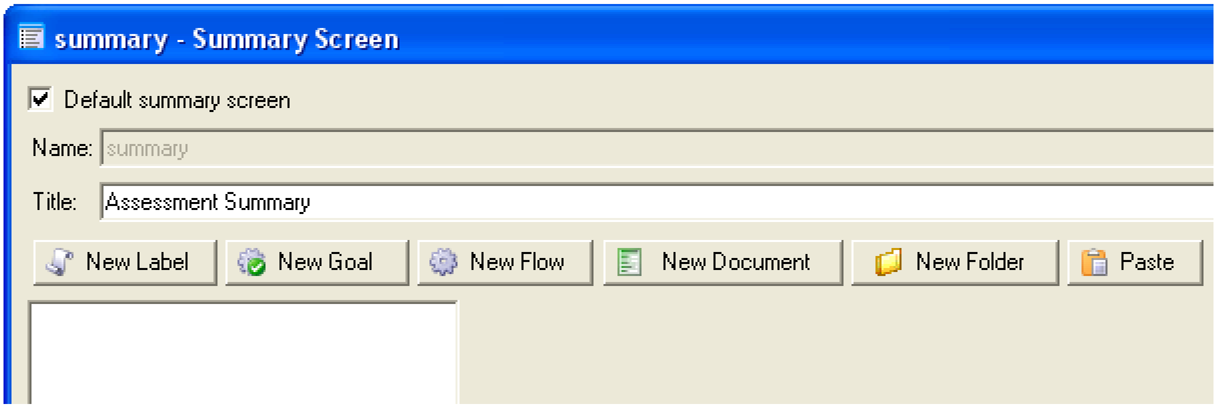
At any point in the interview, returning to the summary screen will typically allow the user to:

- 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:

   ☑ Default summary screen

   Name: summary

   Title: Assessment Summary

   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.

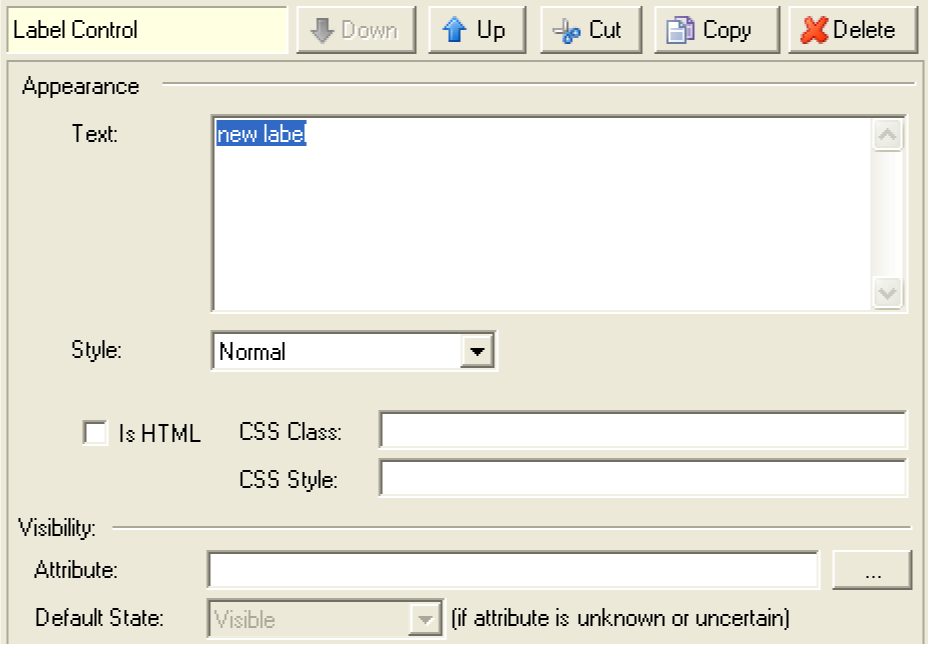
You can add folders to your summary screen in Oracle Policy Modeling to sort your summary screen items into manageable units, and to allow summary screen items to be added for entities. You can also add visibility attributes to summary screen folders, to control the display of all elements within the 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.



Label Control

Appearance

Text: new label

Style: Normal

☐ Is HTML

CSS Class:

CSS Style:

Visibility:

Attribute:

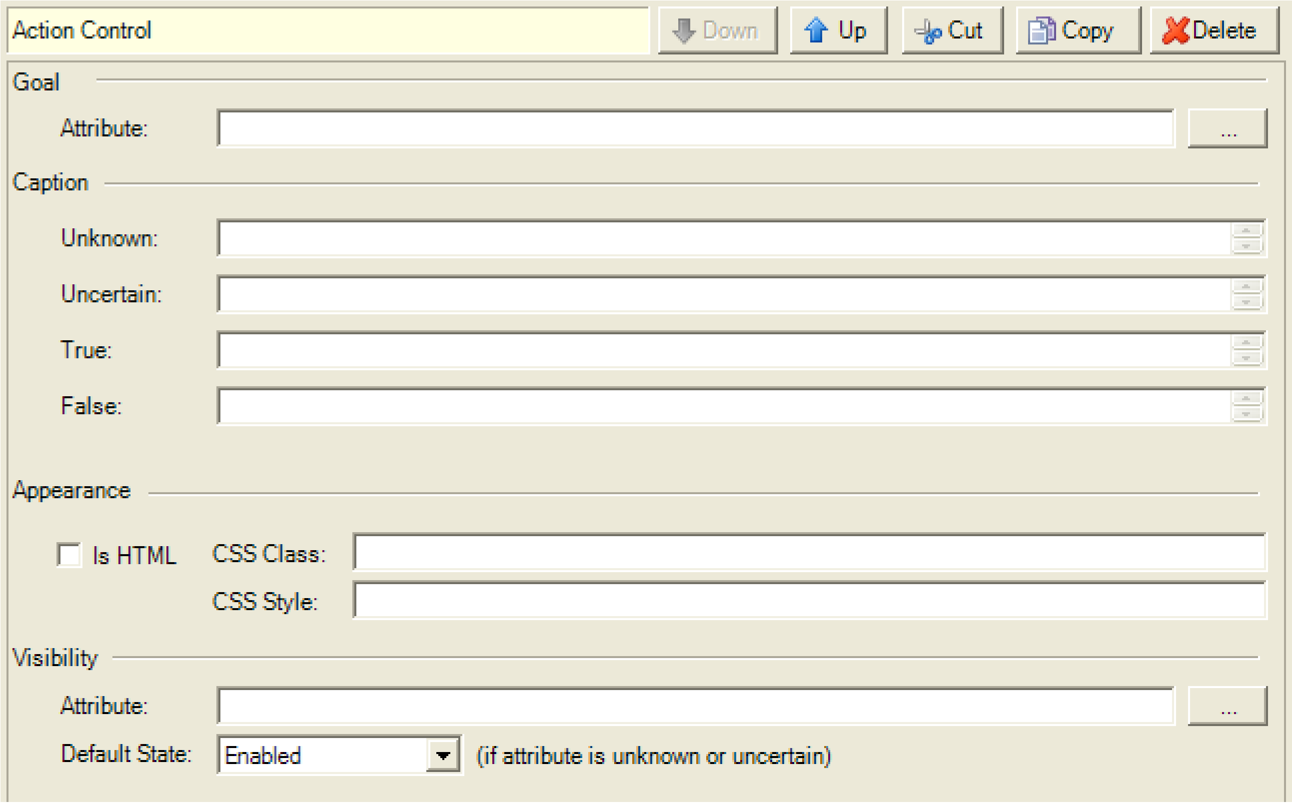
Default State: Visible (if attribute is unknown or uncertain)

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:

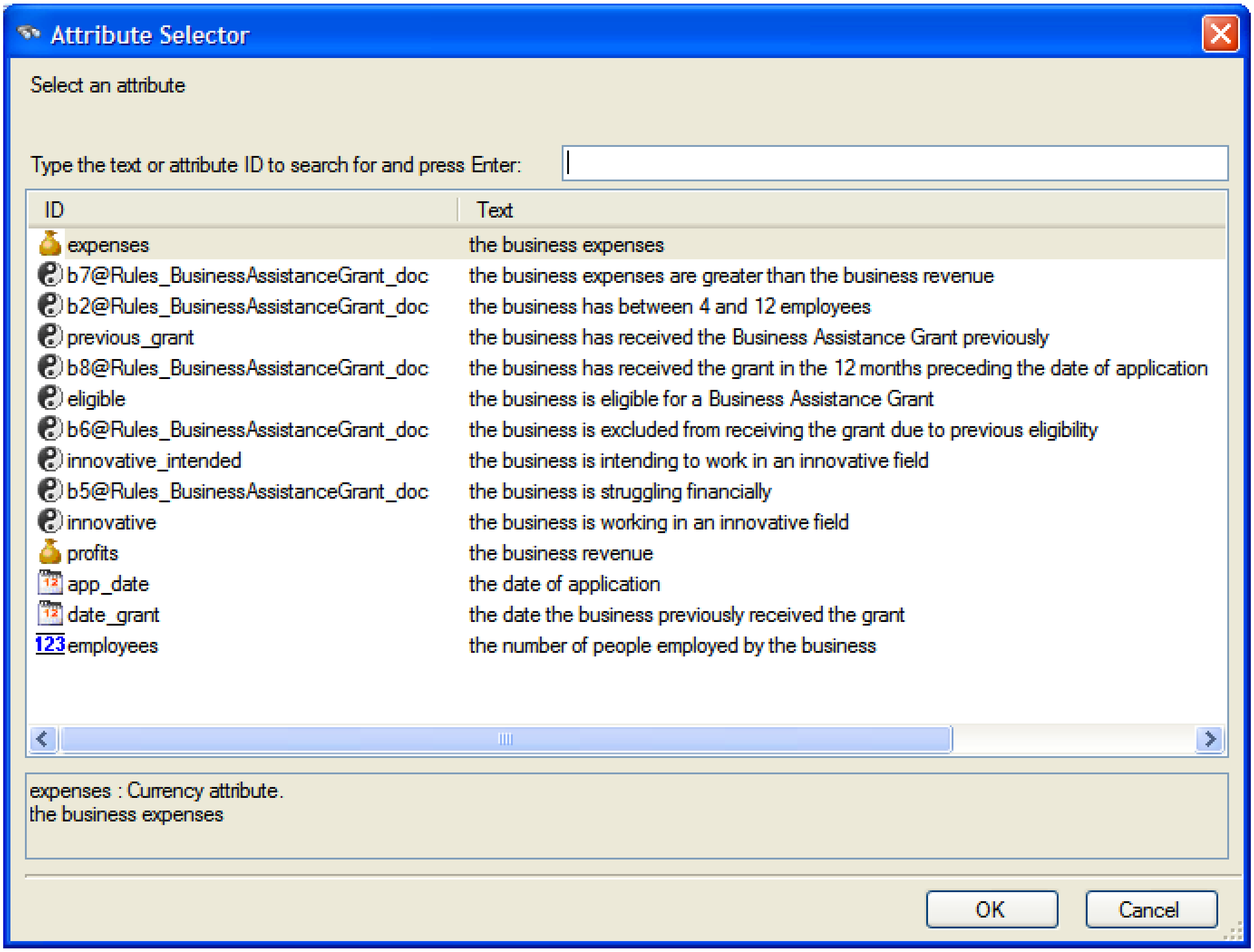
1. Open the summary screen editor dialog by double-clicking on the summary screen entry in the main list of screens for your screens file.
2. Click the **New Goal** button at the top of the summary screen dialog. You can then edit the action control by selecting it in the list of summary screen items in the left hand pane. The details for the action control will appear on the right side of the screen edit dialog.



☐ Is HTML

Default State: Enabled (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.



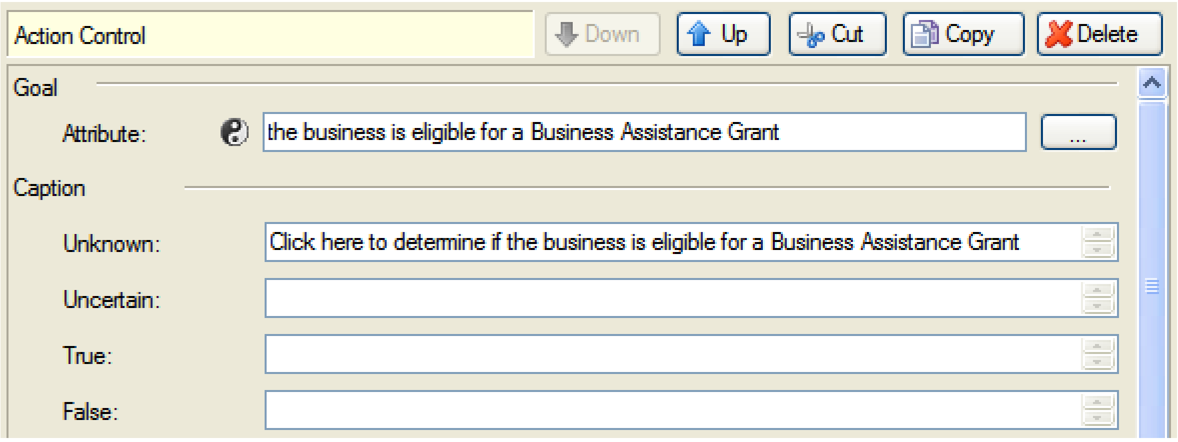
Select an attribute

Type the text or attribute ID to search for and press Enter:

ID	Text
expenses	the business expenses
b7@Rules_BusinessAssistanceGrant_doc	the business expenses are greater than the business revenue
b2@Rules_BusinessAssistanceGrant_doc	the business has between 4 and 12 employees
previous_grant	the business has received the Business Assistance Grant previously
b8@Rules_BusinessAssistanceGrant_doc	the business has received the grant in the 12 months preceding the date of application
eligible	the business is eligible for a Business Assistance Grant
b6@Rules_BusinessAssistanceGrant_doc	the business is excluded from receiving the grant due to previous eligibility
innovative_intended	the business is intending to work in an innovative field
b5@Rules_BusinessAssistanceGrant_doc	the business is struggling financially
innovative	the business is working in an innovative field
profits	the business revenue
app_date	the date of application
date_grant	the date the business previously received the grant
employees	the number of people employed by the business

expenses : Currency attribute.
the business expenses

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.



Goal

Attribute: the business is eligible for a Business Assistance Grant

Caption

Unknown: Click here to determine if the business is eligible for a Business Assistance Grant

Uncertain:

True:

False:

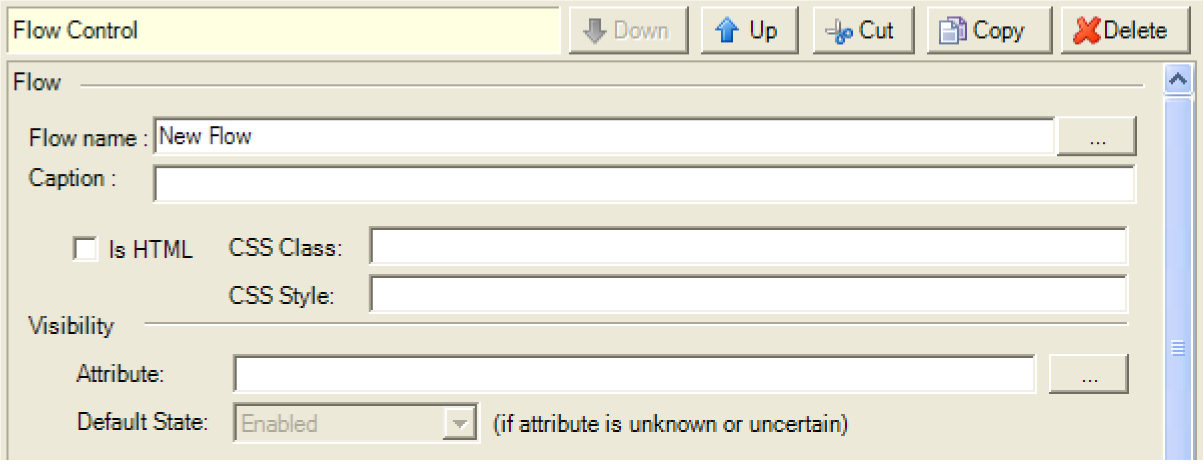
5. Enter **Captions** to define the text that will be displayed when the goal attribute has uncertain, unknown, true and false values. (The **Unknown** caption text is defaulted for you based on the attribute name.)
6. Select the **Is HTML** checkbox if your label/control is HTML. See Change the appearance 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.



Flow Control

Flow

Flow name : New Flow

Caption :

☐ Is HTML

CSS Class:

CSS Style:

Visibility

Attribute:

Default State: Enabled (if attribute is unknown or uncertain)

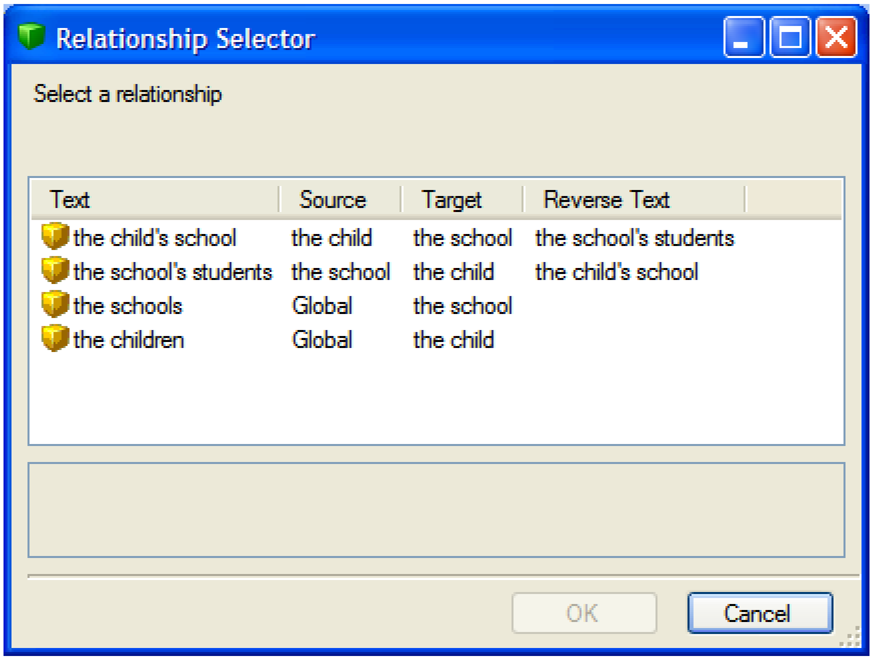
2. Select the **Browse** button next to the **Flow name** text field and select your flow from the list of pre-existing flows in the **Flow Browser** dialog box. Click **OK**.
   In the **Flow Control** pane, the attribute you selected now appears in the **Flow name** field.
3. Modify the **Caption** 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.



Text	Source	Target	Reverse Text
the child's school	the child	the school	the school's students
the school's students	the school	the child	the child's school
the schools	Global	the school	
the children	Global	the child	

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).

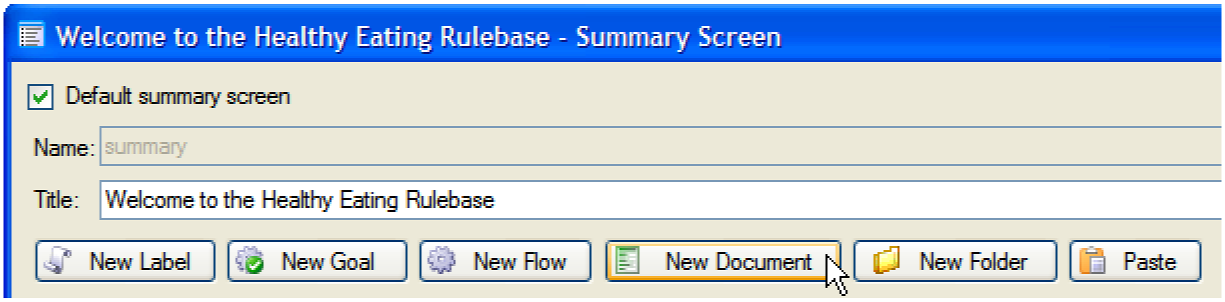
Note that summary screen folders with relationships may be nested, if this structure is reflected in the relevant entity relationships.

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.



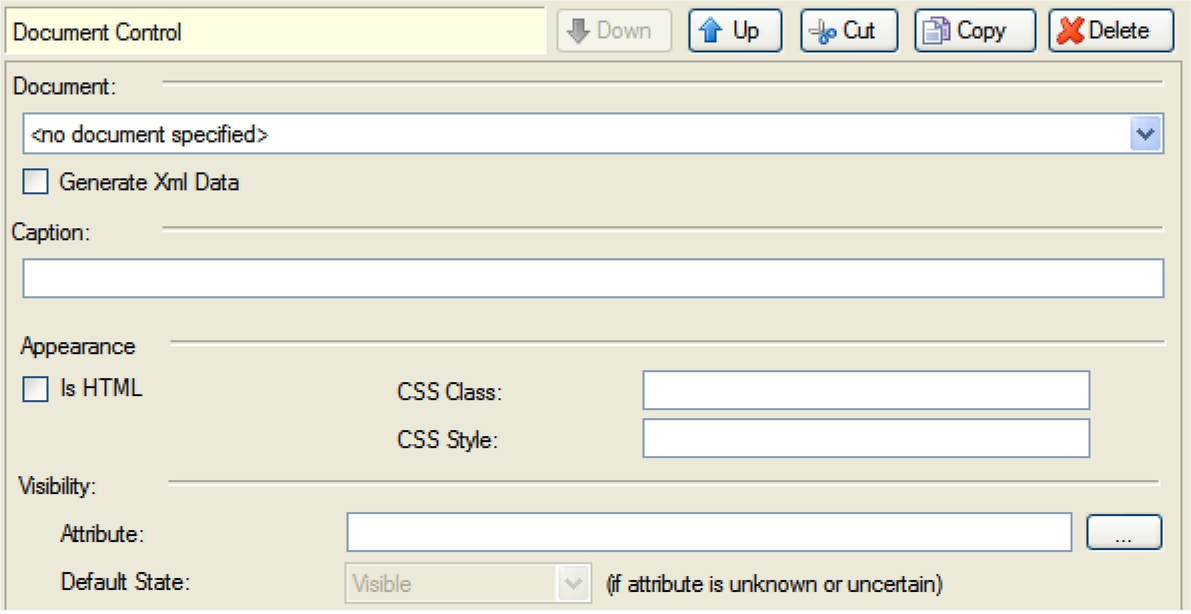
Welcome to the Healthy Eating Rulebase - Summary Screen

☑ Default summary screen

Name: summary

Title: Welcome to the Healthy Eating Rulebase

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.



Document Control

Document: <no document specified>

☐ Generate Xml Data

Caption:

Appearance

☐ Is HTML

CSS Class:

CSS Style:

Visibility:

Attribute:

Default State: Visible (if attribute is unknown or uncertain)

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 sample data.
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 Change the appearance of text 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 Change the appearance of a control 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 Control the visibility of summary screen elements 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.

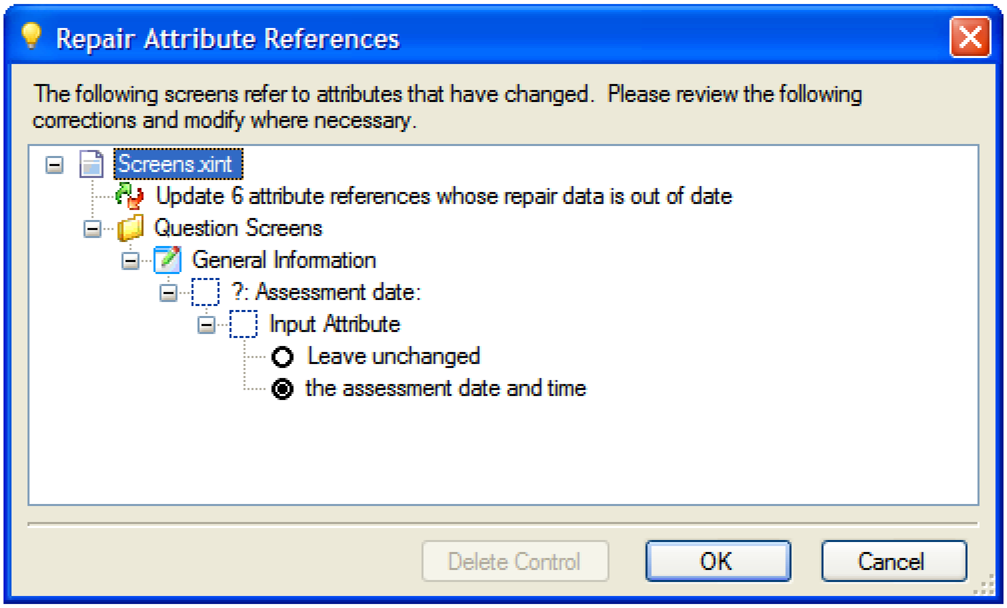
To run an inferred screen attributes report, in Oracle Policy Modeling select **Reports | Inferred Screen Attributes**. When generating this report you have the option to include base level attributes proven by shortcut rules.

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.



The following screens refer to attributes that have changed. Please review the following corrections and modify where necessary.

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

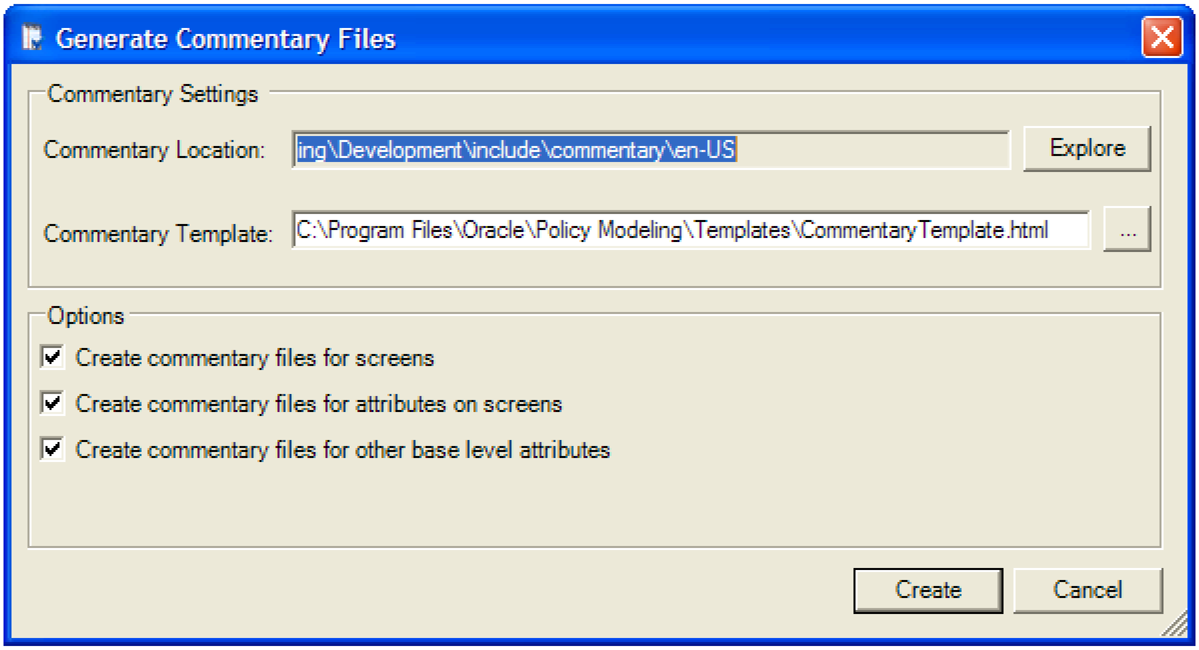
Delete a commentary file

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.



Commentary Location: ing\Development\include\commentary\en-US

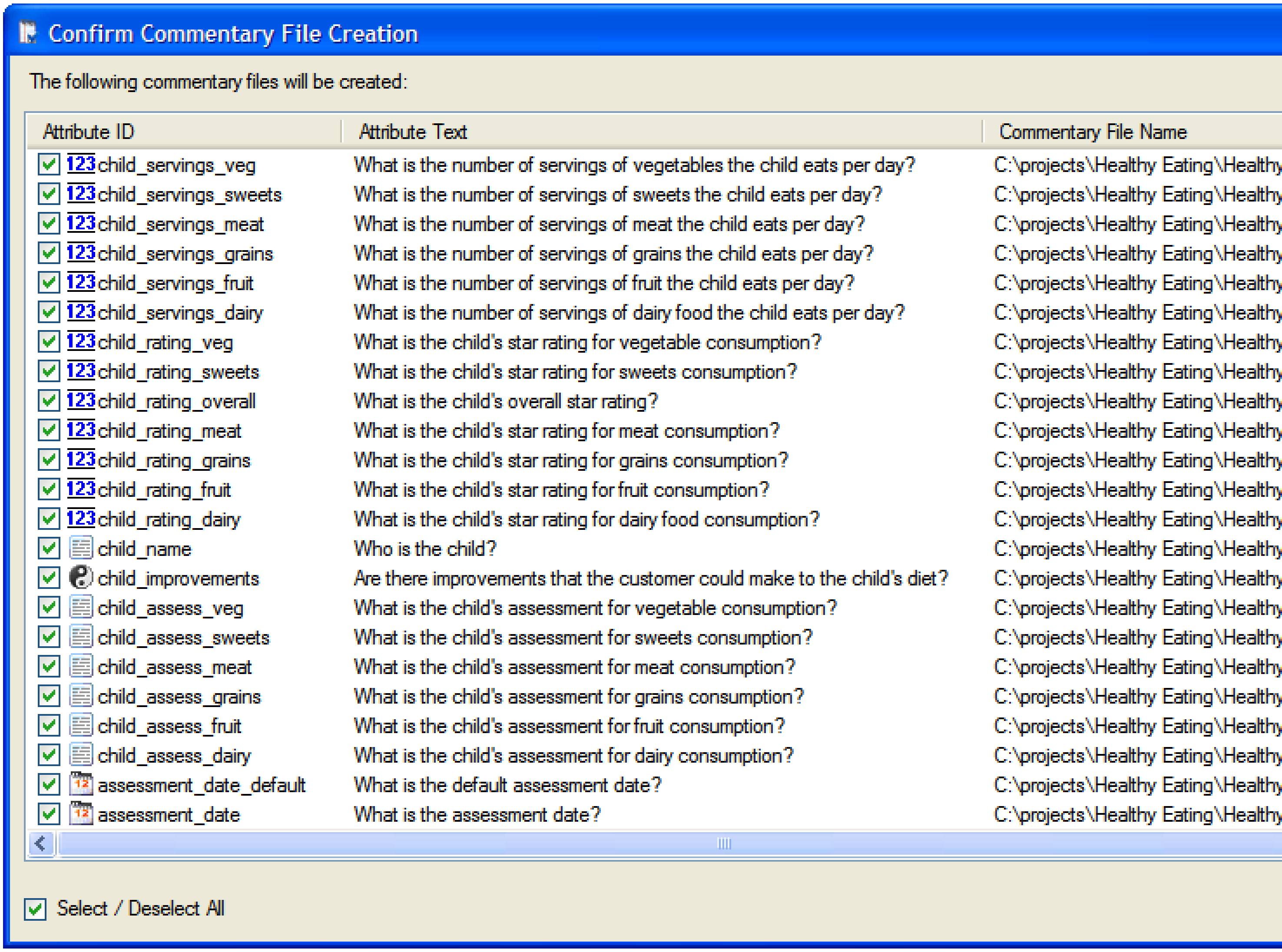
Commentary Template: C:\Program Files\Oracle\Policy Modeling\Templates\CommentaryTemplate.html

☑ Create commentary files for screens
☑ Create commentary files for attributes on screens
☑ Create commentary files for other base level attributes

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.)



The following commentary files will be created:

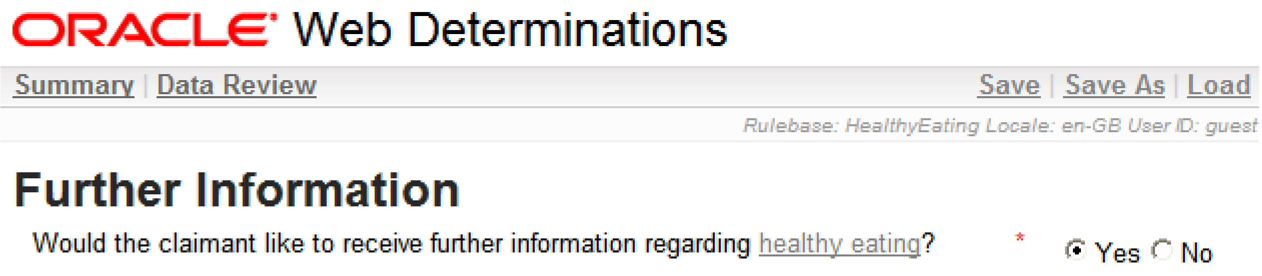
	Attribute ID	Attribute Text	Commentary File Name
☑	child_servings_veg	What is the number of servings of vegetables the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_servings_sweets	What is the number of servings of sweets the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_servings_meat	What is the number of servings of meat the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_servings_grains	What is the number of servings of grains the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_servings_fruit	What is the number of servings of fruit the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_servings_dairy	What is the number of servings of dairy food the child eats per day?	C:\projects\Healthy Eating\Healthy
☑	child_rating_veg	What is the child's star rating for vegetable consumption?	C:\projects\Healthy Eating\Healthy
☑	child_rating_sweets	What is the child's star rating for sweets consumption?	C:\projects\Healthy Eating\Healthy
☑	child_rating_overall	What is the child's overall star rating?	C:\projects\Healthy Eating\Healthy
☑	child_rating_meat	What is the child's star rating for meat consumption?	C:\projects\Healthy Eating\Healthy
☑	child_rating_grains	What is the child's star rating for grains consumption?	C:\projects\Healthy Eating\Healthy
☑	child_rating_fruit	What is the child's star rating for fruit consumption?	C:\projects\Healthy Eating\Healthy
☑	child_rating_dairy	What is the child's star rating for dairy food consumption?	C:\projects\Healthy Eating\Healthy
☑	child_name	Who is the child?	C:\projects\Healthy Eating\Healthy
☑	child_improvements	Are there improvements that the customer could make to the child's diet?	C:\projects\Healthy Eating\Healthy
☑	child_assess_veg	What is the child's assessment for vegetable consumption?	C:\projects\Healthy Eating\Healthy
☑	child_assess_sweets	What is the child's assessment for sweets consumption?	C:\projects\Healthy Eating\Healthy
☑	child_assess_meat	What is the child's assessment for meat consumption?	C:\projects\Healthy Eating\Healthy
☑	child_assess_grains	What is the child's assessment for grains consumption?	C:\projects\Healthy Eating\Healthy
☑	child_assess_fruit	What is the child's assessment for fruit consumption?	C:\projects\Healthy Eating\Healthy
☑	child_assess_dairy	What is the child's assessment for dairy consumption?	C:\projects\Healthy Eating\Healthy
☑	assessment_date_default	What is the default assessment date?	C:\projects\Healthy Eating\Healthy
☑	assessment_date	What is the assessment date?	C:\projects\Healthy Eating\Healthy

☑ Select / Deselect All

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>
   For example, "What is the claimant's weekly net pay?".
   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, pay.
   (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 pay?".
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

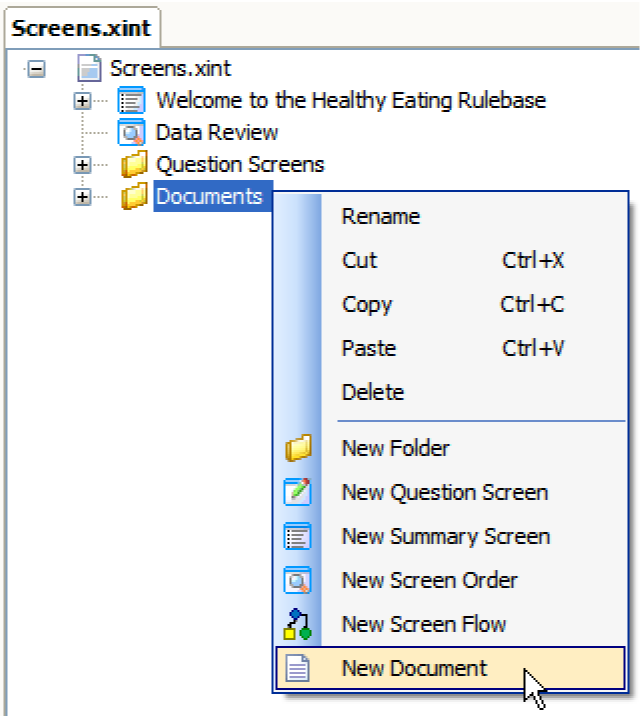
By default, the first screens file that is added to a project will contain a Documents folder. To add additional folders to your screens file:

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.

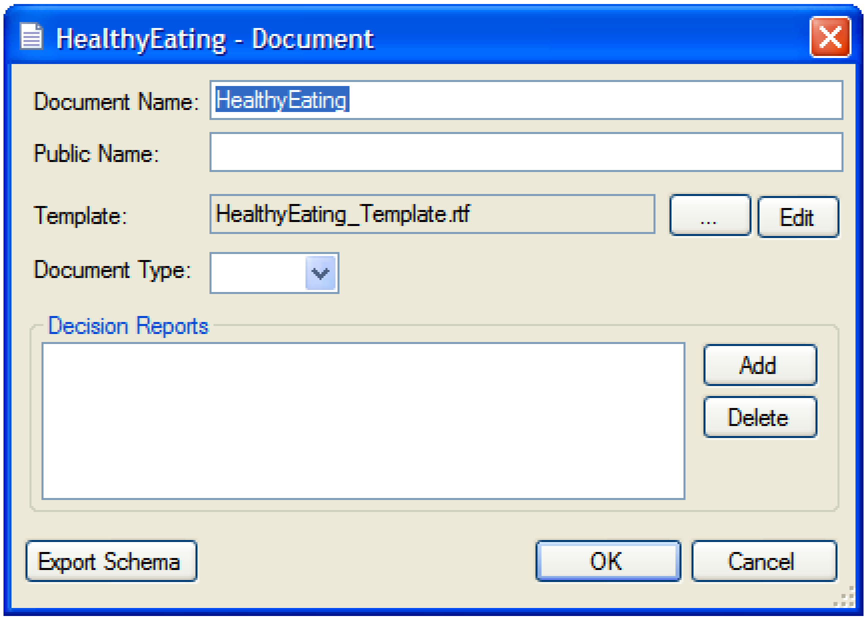
Create a new interview document

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:



Document Name: HealthyEating

Public Name:

Template: HealthyEating_Template.rtf

Document Type:

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

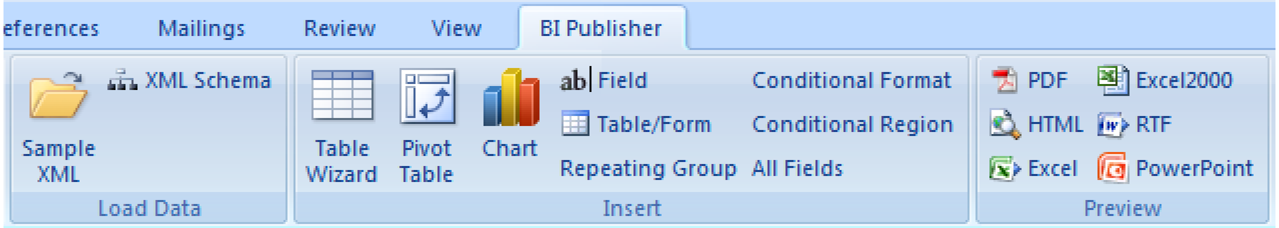
To modify an existing interview document, double-click on the document name in your screens file, or right-click and select **Open** from the pop-up menu. Make the necessary changes in the Document editor and then click **OK**.

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.



Note: If you cannot see the BI Publisher Ribbon in your Microsoft Word document after you open it from Policy Modeling, you need to install BI Publisher Desktop. For more information, see Download BI Publisher.

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.

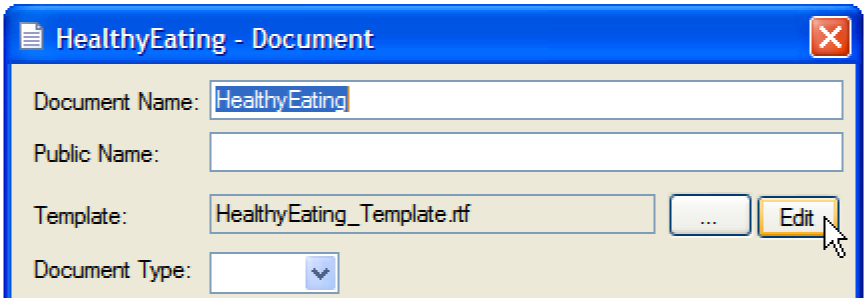
Create a template file

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:



Document Name: HealthyEating
Public Name:
Template: HealthyEating_Template.rtf
Document Type:

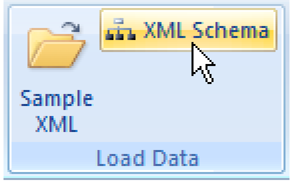
Alternatively, you can open the template file from its location in the project under \include\templates\<locale> using Microsoft Word or Windows Explorer.

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:

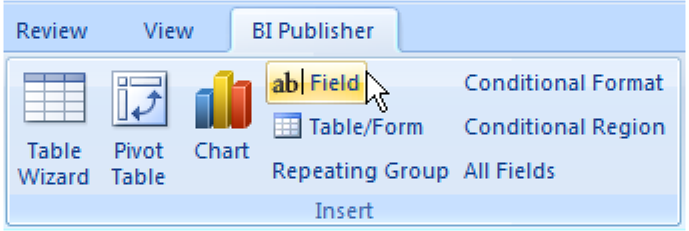
- fields
- tables
- charts
- pictures

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.

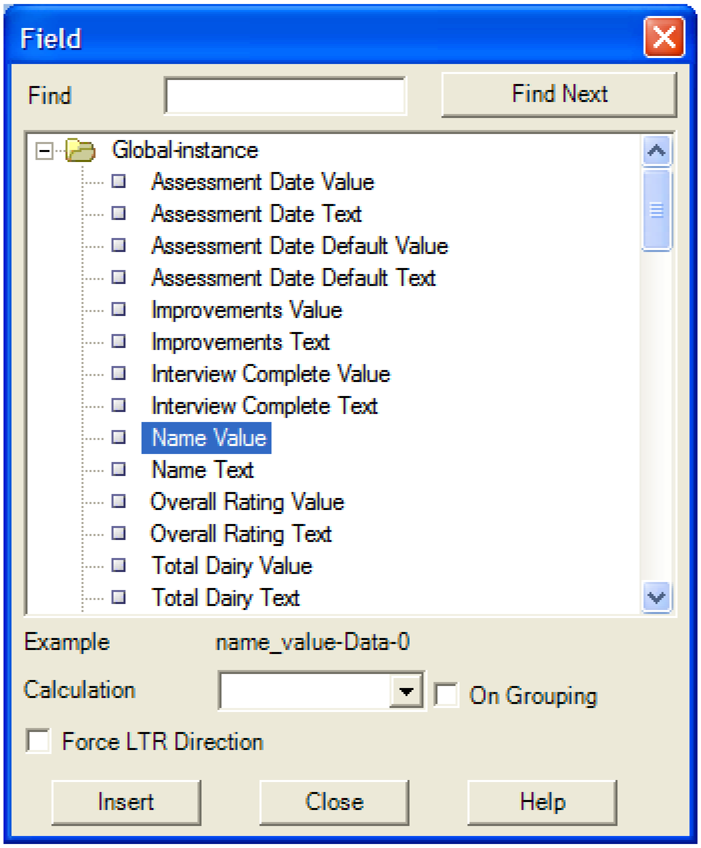
To insert a field into a template:

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:



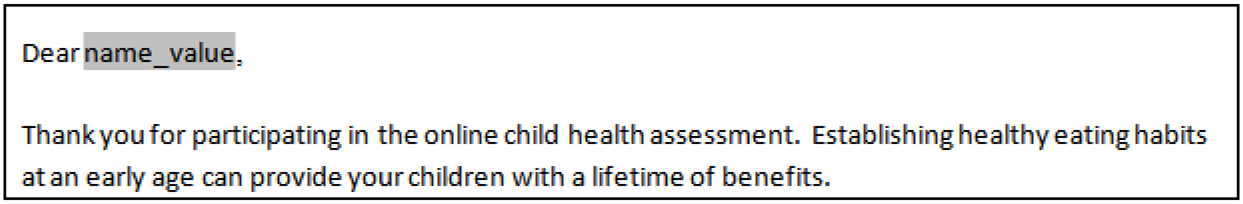
(If you have not already loaded a data source, you will be prompted to do so now.)

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):



Example name_value-Data-0

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:



Dear name_value,

Thank you for participating in the online child health assessment. Establishing healthy eating habits at an early age can provide your children with a lifetime of benefits.

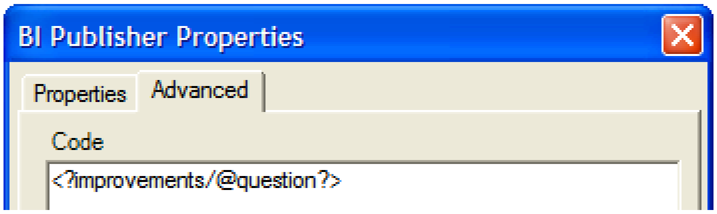
IMPORTANT NOTE: For attribute values (formatted for the project region) and attribute text this is all you need to do. For other field types (eg unformatted attribute values, attribute question text, attribute type, decision reports, conditional text etc) you need to customize a BI Publisher field (see 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.

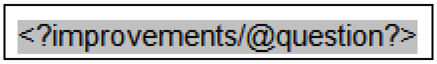
1. Follow the steps above for inserting a field into your template. Use either the Value or Text field for the relevant attribute (it does not matter which as you will be customizing the field anyway).
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: 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?>":



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:



<?improvements/@question?>

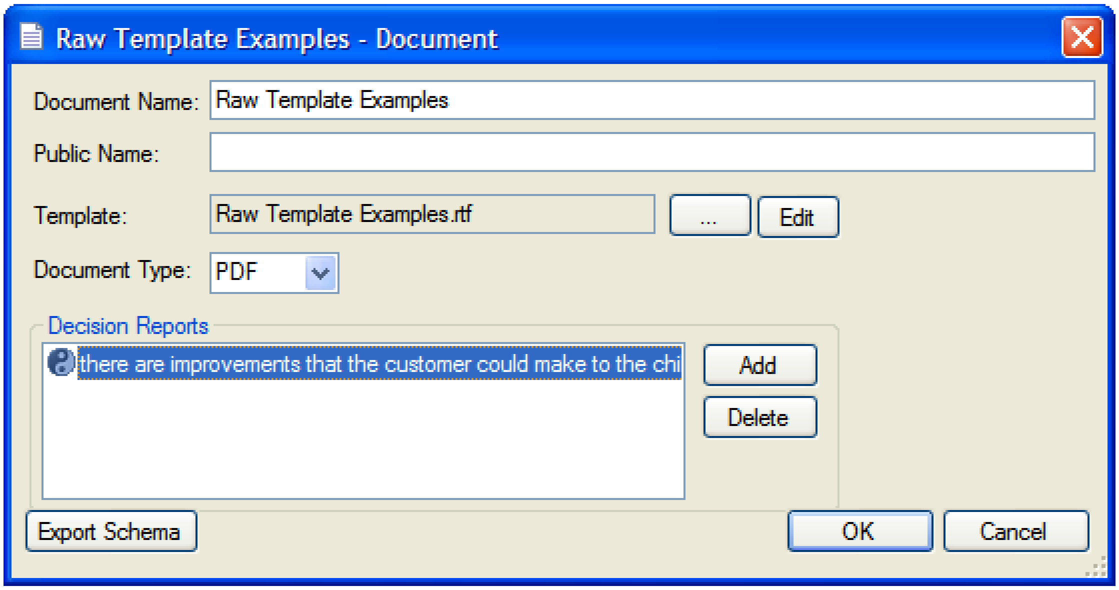
For the format required for other fields, see BI Publisher code for Oracle Policy Modeling.

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.



Document Name: Raw Template Examples

Public Name:

Template: Raw Template Examples.rtf

Document Type: PDF

Decision Reports: there are improvements that the customer could make to the chi

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:

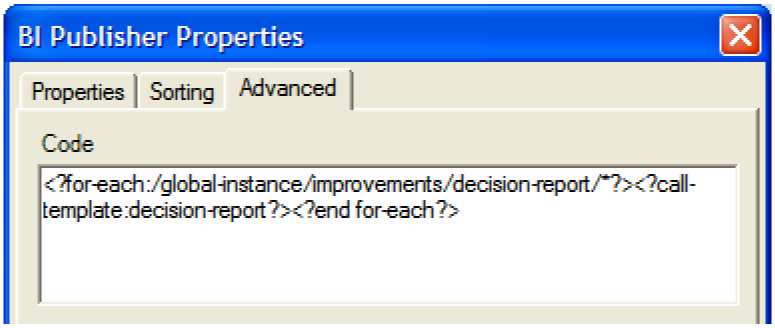
```
<?template@inlines:decision-report?>
<?if@inlines:"attribute-node"?>
<fo:list-block start-indent="{count(ancestor::attribute-node) * 7}mm">
<fo:list-item>
<fo:list-item-label>
<fo:block>*</fo:block>
</fo:list-item-label>
<fo:list-item-body>
<fo:block><xsl:value-of select="@text"/></fo:block>
</fo:list-item-body>
</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:

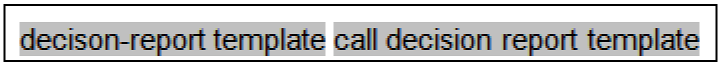


Code

```
<?for-each:/global-instance/improvements/decision-report/*?><?call-
template:decision-report?><?end for-each?>
```

   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

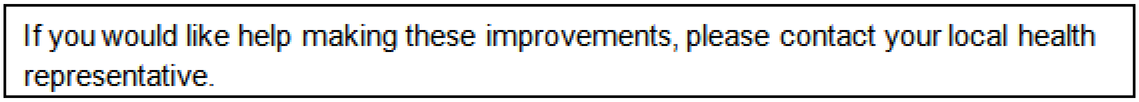
You can specify that certain text is only shown when a particular condition is met. Both simple conditions and multiple conditions are supported in BI Publisher. Conditional text can be achieved by:

1. Using the BI Publisher Conditional Region dialog, or
2. Manually defining the conditional fields.

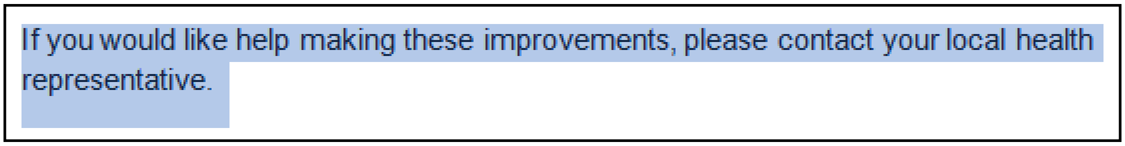
The Conditional Region dialog can be used when the condition is that an attribute EQUALS a value. It can also be used for greater/less than comparisons for number variables provided you select 'Number' from the drop-down list (see below).

To insert conditional text using the Conditional Region dialog:

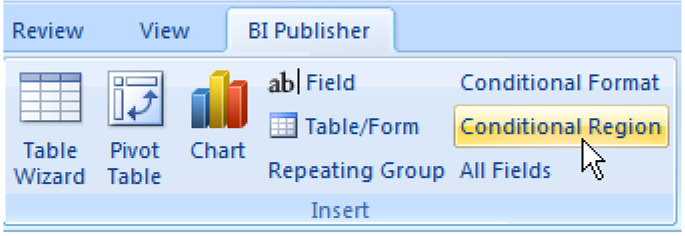
1. In your template write the text that you would like displayed under certain conditions.


If you would like help making these improvements, please contact your local health representative.

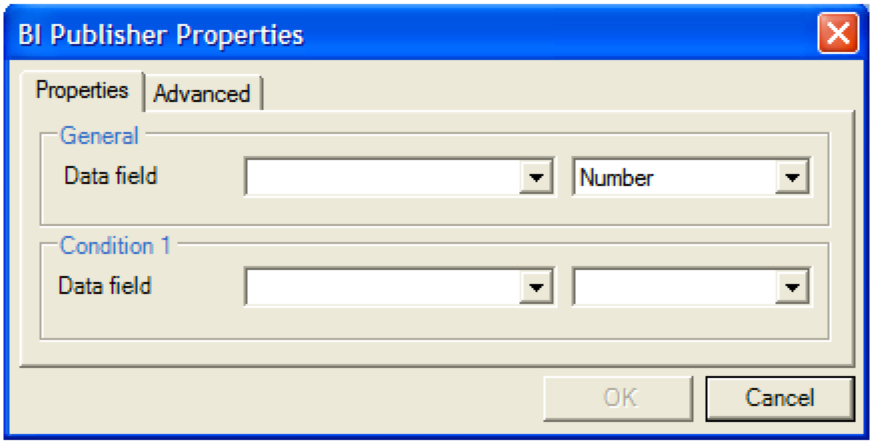
2. Select the text.


If you would like help making these improvements, please contact your local health representative.

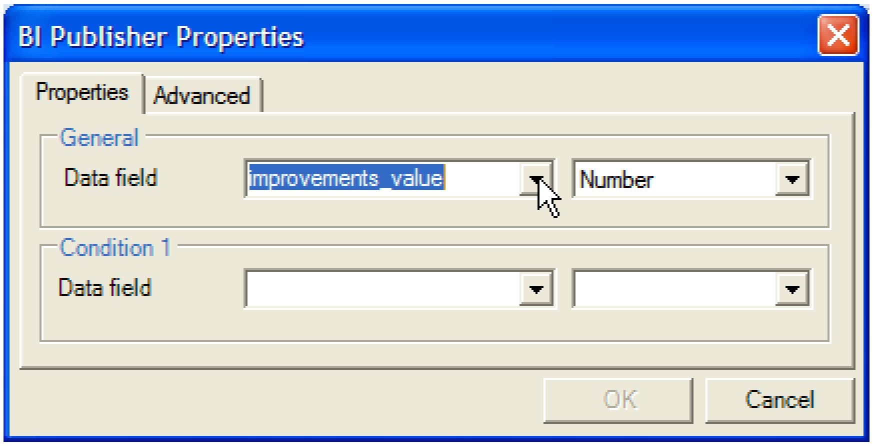
3. Click on the **Conditional Region** button on the BI Publisher menu:



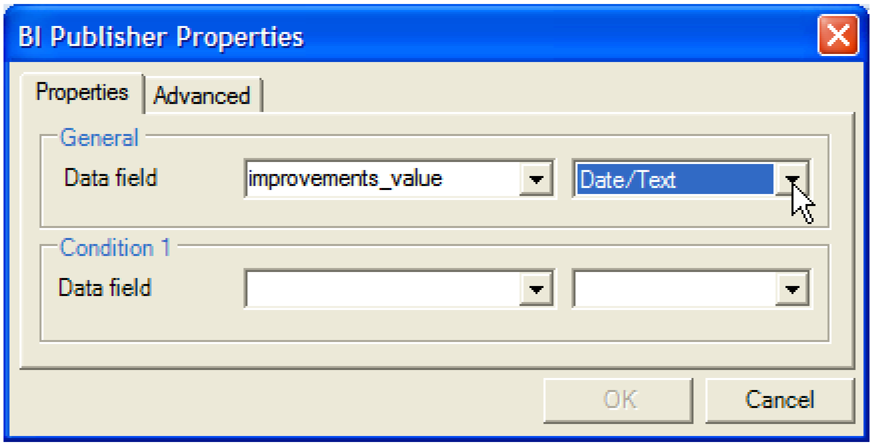
The following BI Publisher dialog will be displayed:



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.



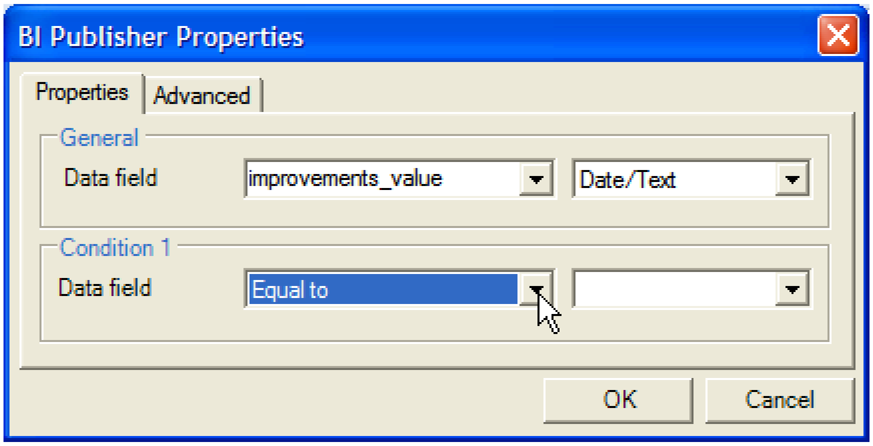
General

Data field improvements_value Date/Text

Condition 1

Data field

6. In the **Condition 1** box, select the condition that applies to the attribute from the **Data field** drop-down list.



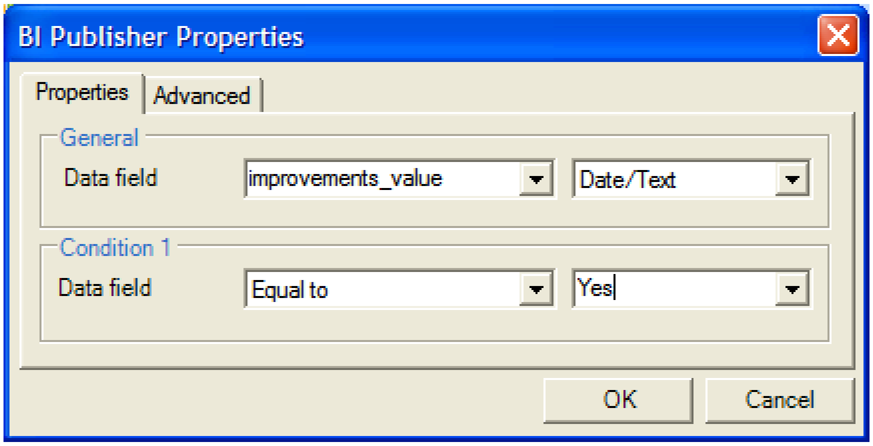
General

Data field improvements_value Date/Text

Condition 1

Data field Equal to

7. Also in the Condition 1 box, type in the condition value (formatted).



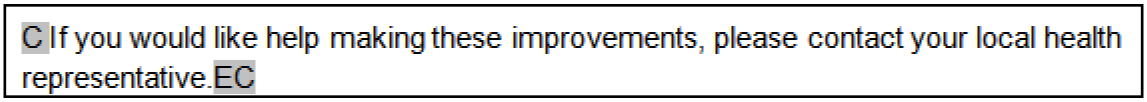
General

Data field improvements_value Date/Text

Condition 1

Data field Equal to Yes

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?>

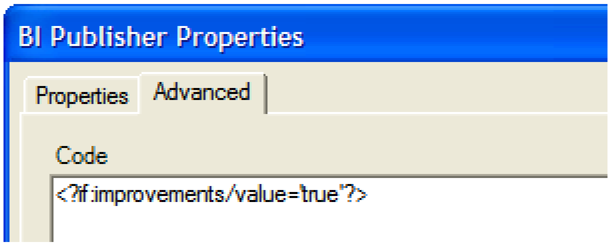
Multiple conditions take this format:

- <?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:

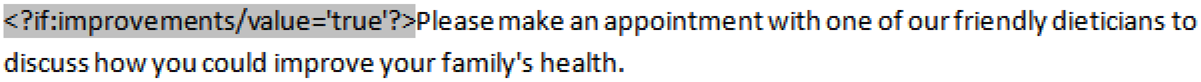
1. Insert a field for the relevant attribute value into your template.
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: If double-clicking the field does not open the BI Publisher Properties dialog, re-load your XML.)
3. Click on the **Advanced** tab. Enter the code for the start of the condition (eg <?if:**attribute_id**/value='**value**'?>):



Code

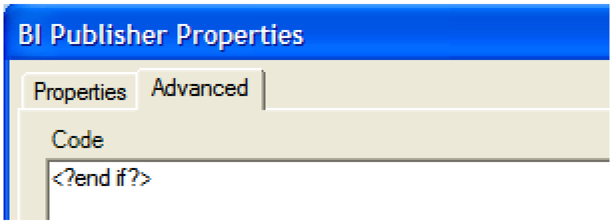
<?if:improvements/value='true'?>

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.



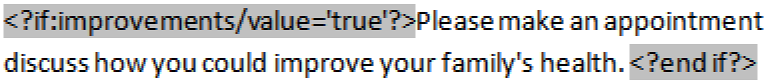
<?if:improvements/value='true'?>Please make an appointment with one of our friendly dieticians to discuss how you could improve your family's health.

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?>:



<?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:

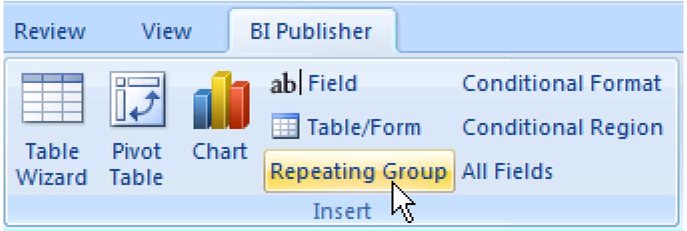


<?if:improvements/value='true'?>Please make an appointment with one of our friendly dieticians to discuss how you could improve your family's health. <?end if?>

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:



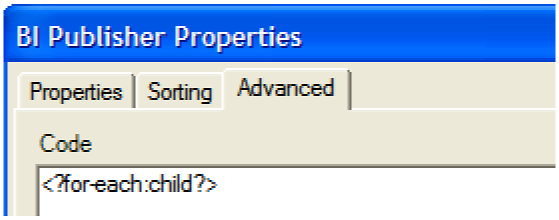
See the Template Builder for Microsoft Word help file for more information on how to set up your entity attributes in this way.

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:

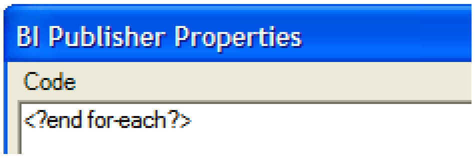
1. Insert a field for the relevant entity-level attribute value into your template.
2. Double-click on the field in the template to open the **BI Publisher Properties** dialog box. (TIP: If double-clicking the field does not open the BI Publisher Properties dialog, re-load your XML.)
3. Click on the **Advanced** tab. Enter the code that defines which entity the attribute belongs to, ie <?for-each:**entity_id**?>:



Code

```
<?for-each:child?>
```

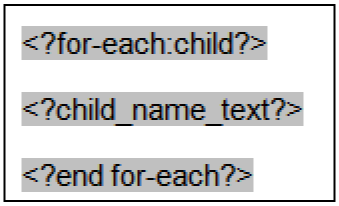
4. Select the code text and copy it (Ctrl+C), then paste it (Ctrl+V) into the **Text to display** field. Click **OK**.
5. In your template document, on a new line after the "for-each" field, insert another field for the relevant entity-level attribute into your template (just as you did in step 1). If 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?>:



Code

```
<?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:



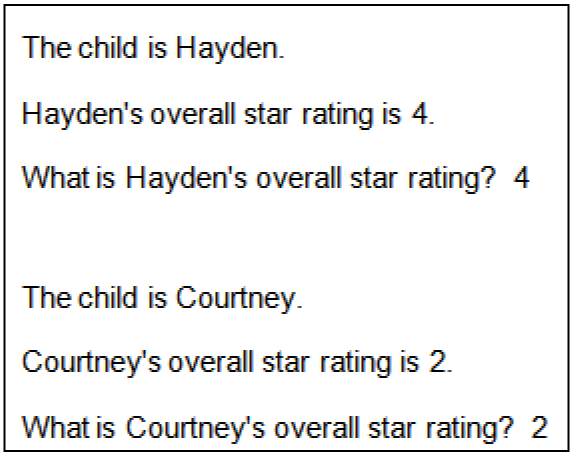
```
<?for-each:child?>

<?child_name_text?>

<?end for-each?>
```

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.

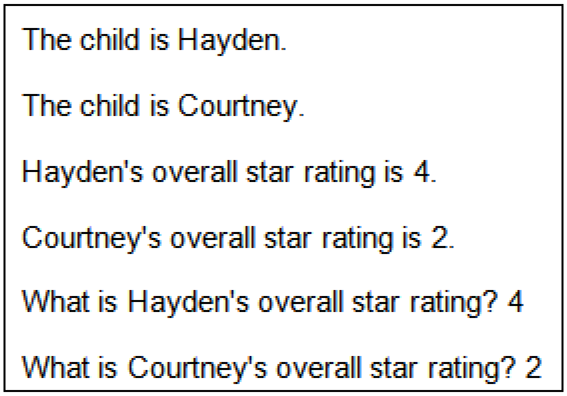
Courtney's overall star rating is 2.

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.

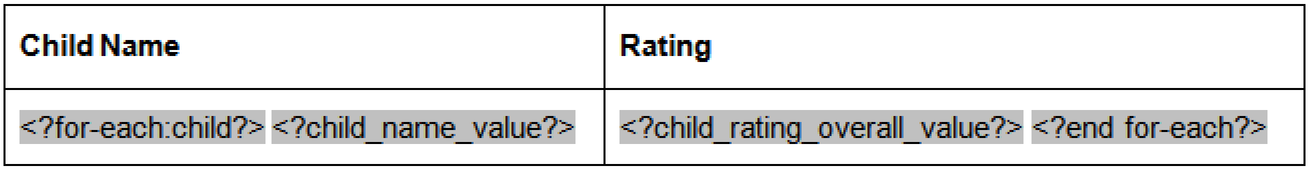
Hayden's overall star rating is 4.

Courtney's overall star rating is 2.

What is Hayden's overall star rating? 4

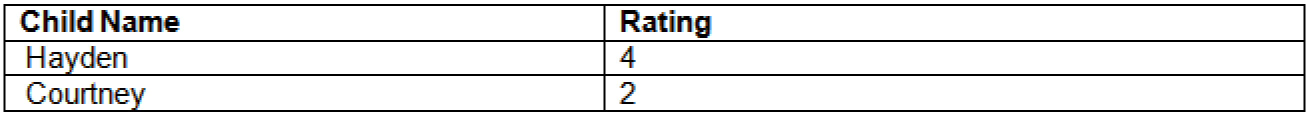
What is Courtney's overall star rating? 2

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:

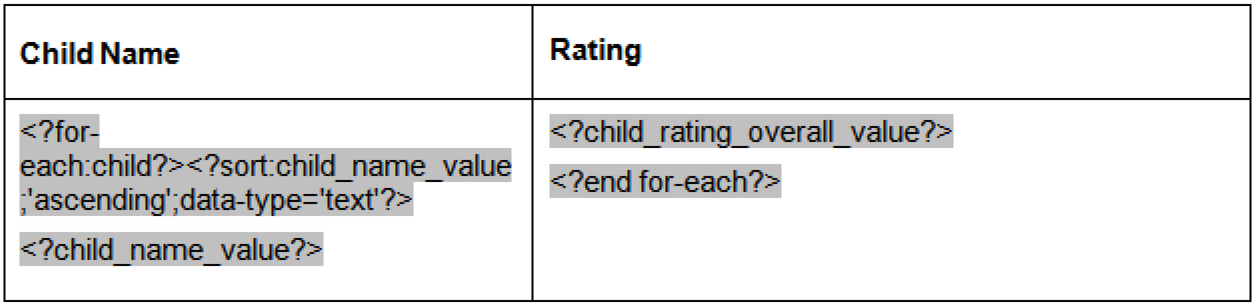


Child Name	Rating
Hayden	4
Courtney	2

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:

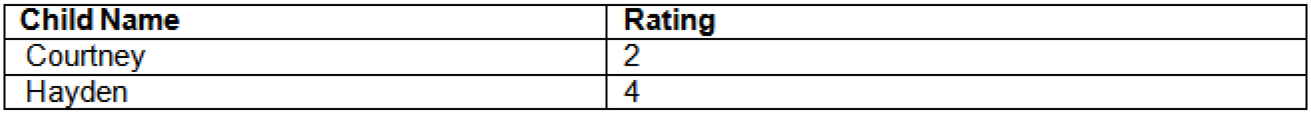
```
<?sort:entity_name_id_value;'ascending';data-type='text'?>
```

For example,



Child Name	Rating
<?for-each:child?><?sort:child_name_value;'ascending';data-type='text'?> <?child_name_value?>	<?child_rating_overall_value?> <?end for-each?>

This would display as:



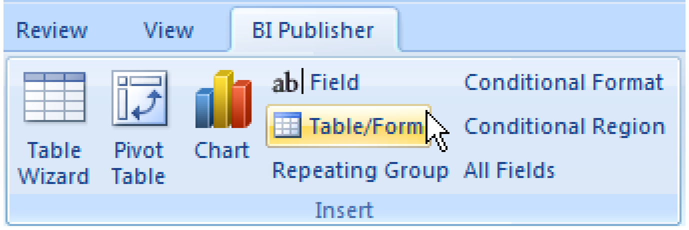
Child Name	Rating
Courtney	2
Hayden	4

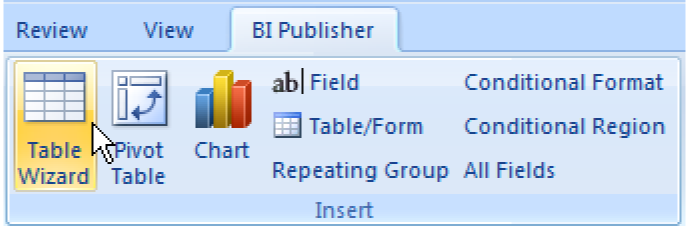
TIP: Sometimes having the full display text for the "for each" fields at the start and end of the rows can upset the formatting of your table. If so, replace the full display text with an abbreviation (eg "F" for the <?for-each:**entity_id**?> field, and "E" for the <?end for-each?> field). See the Combined 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).

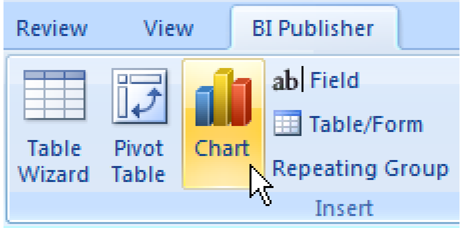




For more information on using the BI Publisher table formats, see the Template Builder for Microsoft Word help file.

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:

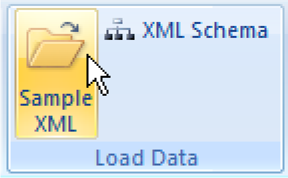
1. Open the document control on the summary screen.
2. Select the **Generate Xml Data** checkbox, then click **OK**.
3. Select **Build | Build and Debug**. In the **Debug Options** dialog box, select the option to debug **With screens**, and select the option **Build and deploy with built-in Oracle Web 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**:

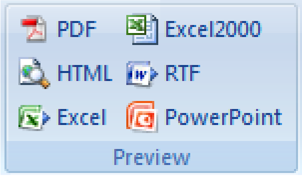


2. In the dialog box, select the XML file that contains your sample data (see above). You will be told when this data has been loaded successfully.

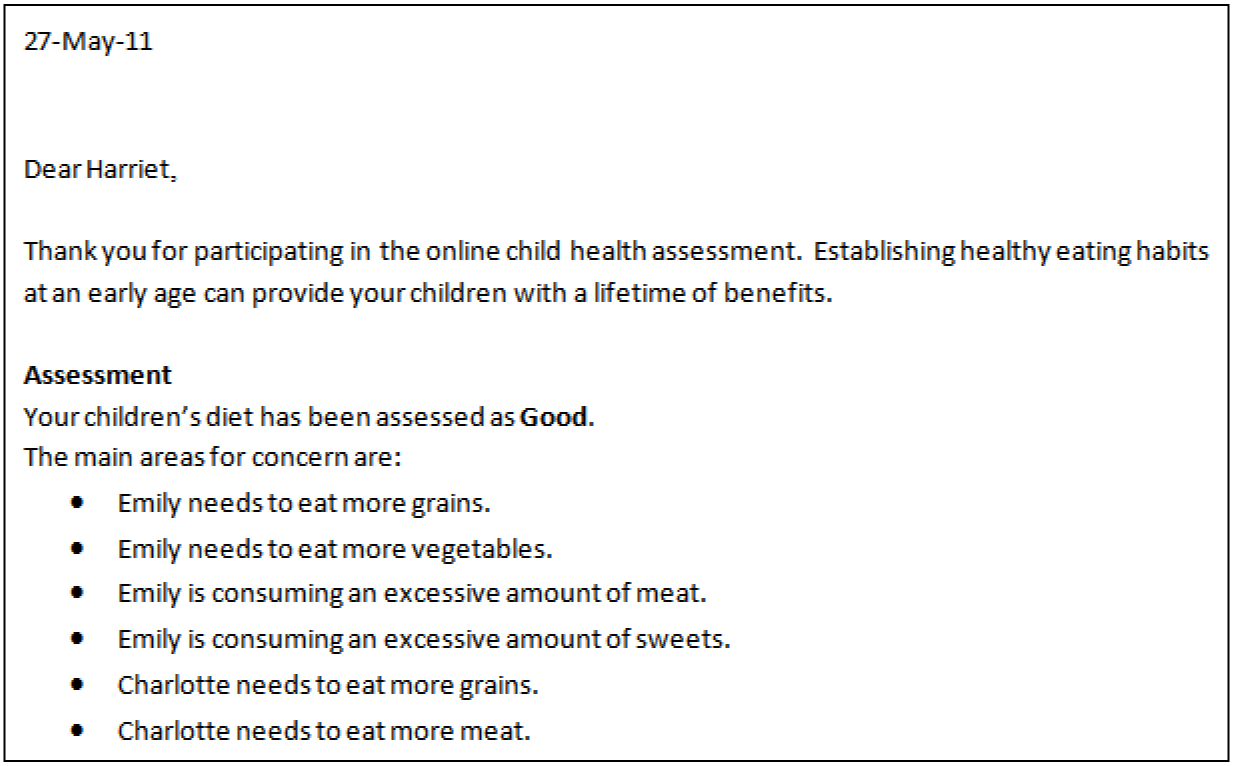
Preview the output

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:



27-May-11

Dear Harriet,

Thank you for participating in the online child health assessment. Establishing healthy eating habits at an early age can provide your children with a lifetime of benefits.

Assessment

Your children's diet has been assessed as **Good.**

The main areas for concern are:

- Emily needs to eat more grains.
- Emily needs to eat more vegetables.
- Emily is consuming an excessive amount of meat.
- Emily is consuming an excessive amount of sweets.
- Charlotte needs to eat more grains.
- Charlotte needs to eat more meat.

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

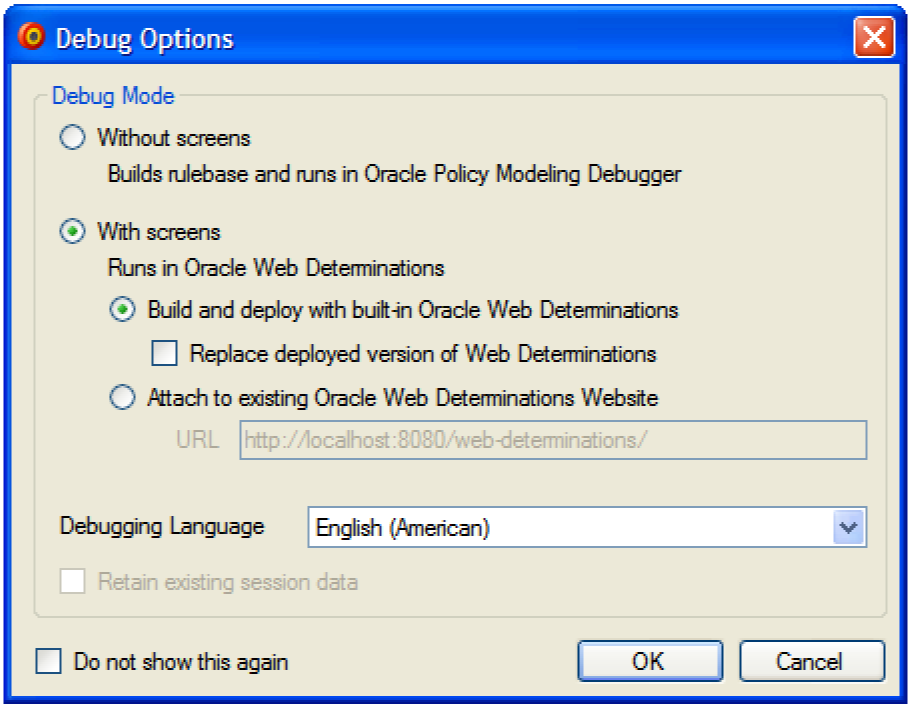
Save an interview in Web Determinations

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

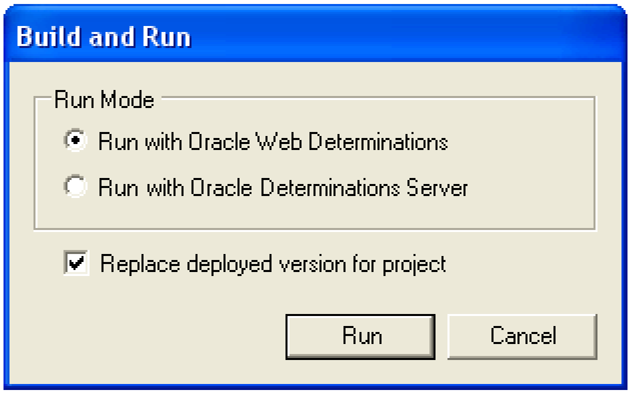
By default, a deployed instance of Oracle Web Determinations doesn't support debugging. To enable debugging for either Java or .NET, the following change needs to be made to the configuration file:

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.



Run Mode
● Run with Oracle Web Determinations
○ Run with Oracle Determinations Server

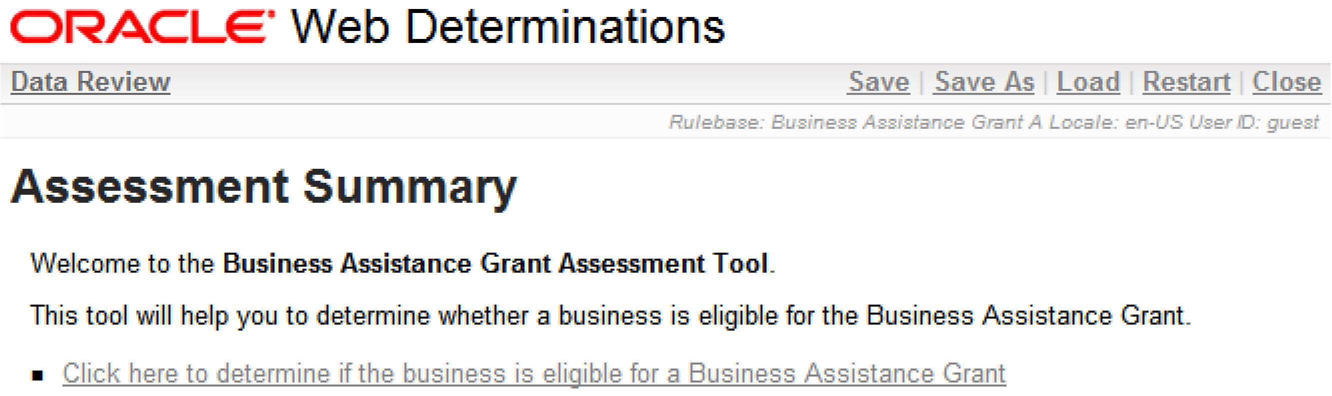
☑ Replace deployed version for project

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.



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

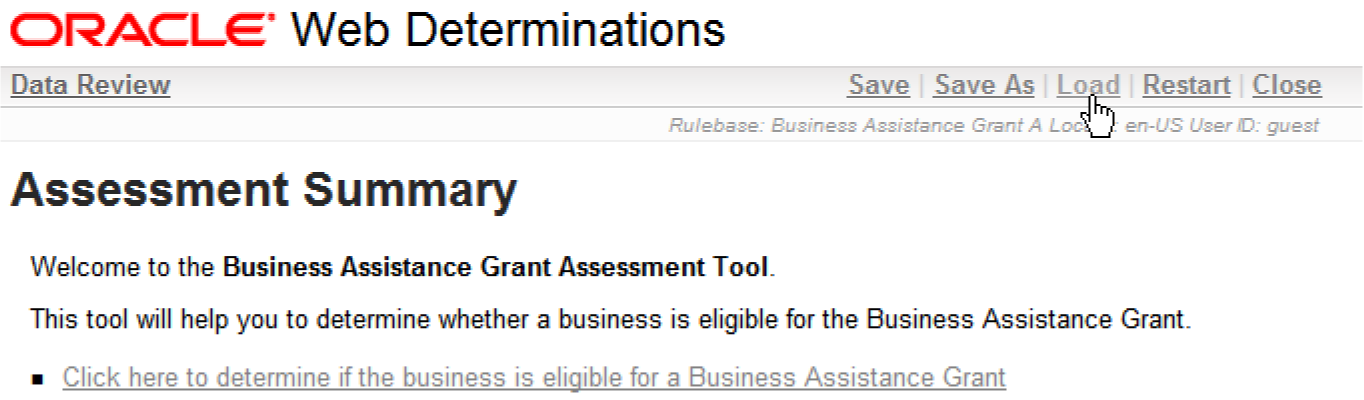
(If no summary screen has been defined in Oracle Policy Modeling, the summary screen in Web Determinations will be blank.)

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.



Rulebase: Business Assistance Grant A Loc : en-US User ID: guest

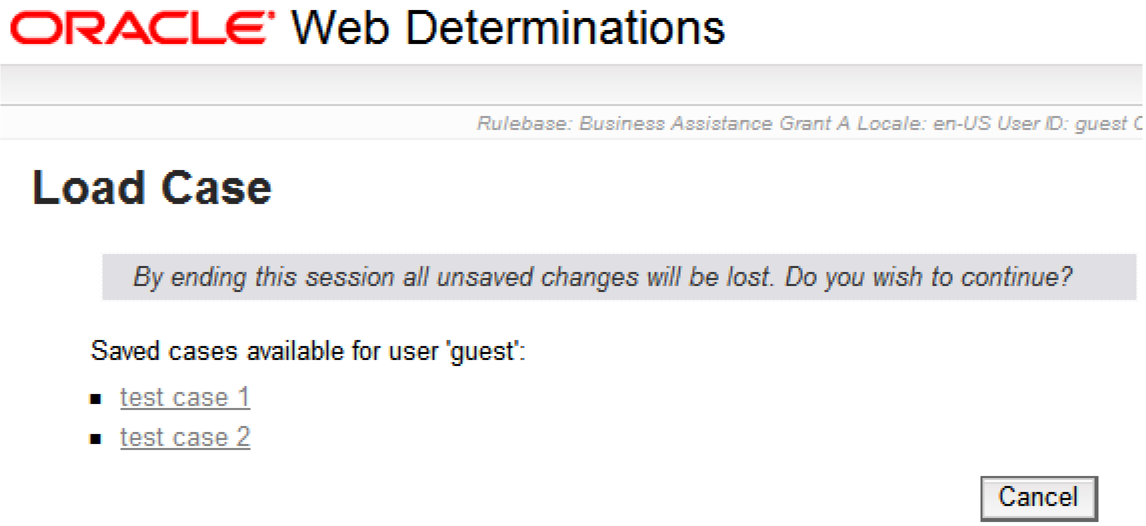
Assessment Summary

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.



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?

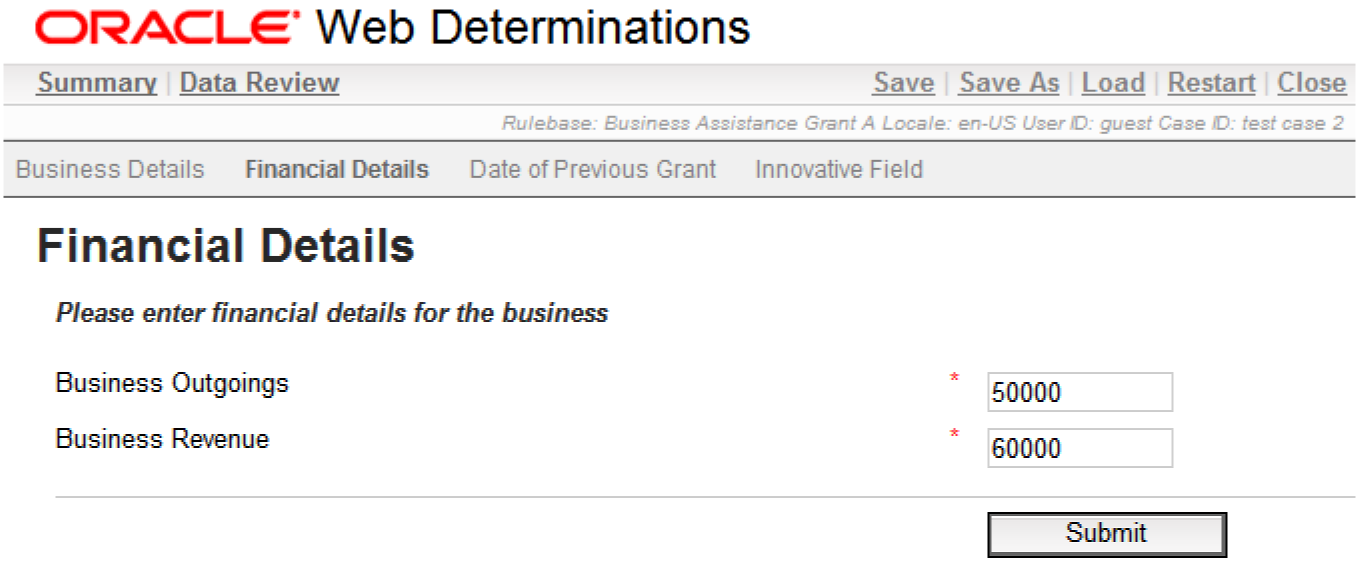
Saved cases available for user 'guest':

- test case 1
- test case 2

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:



Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 2

Financial Details

Please enter financial details for the business

Business Outgoings	*	50000
Business Revenue	*	60000

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.

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.

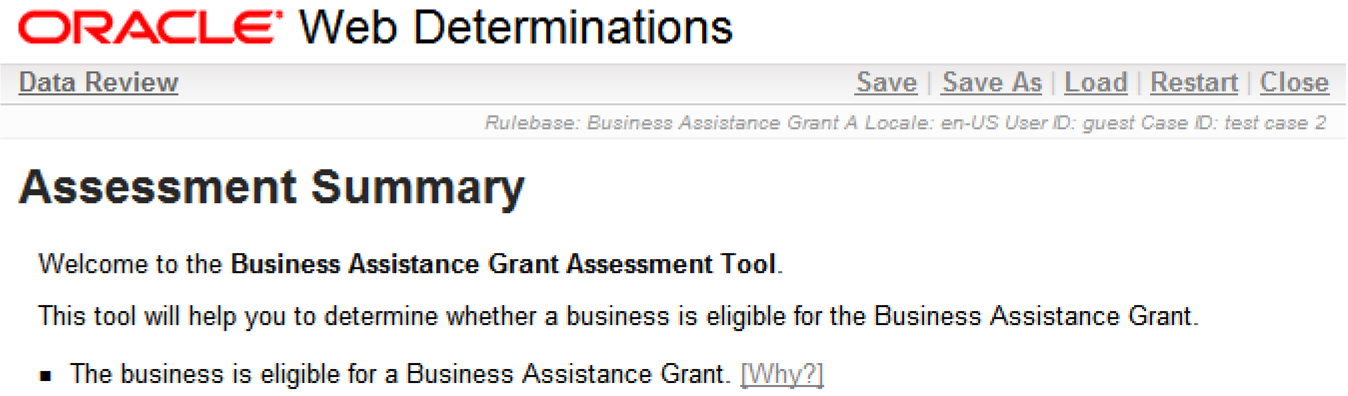
Help text

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.



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.

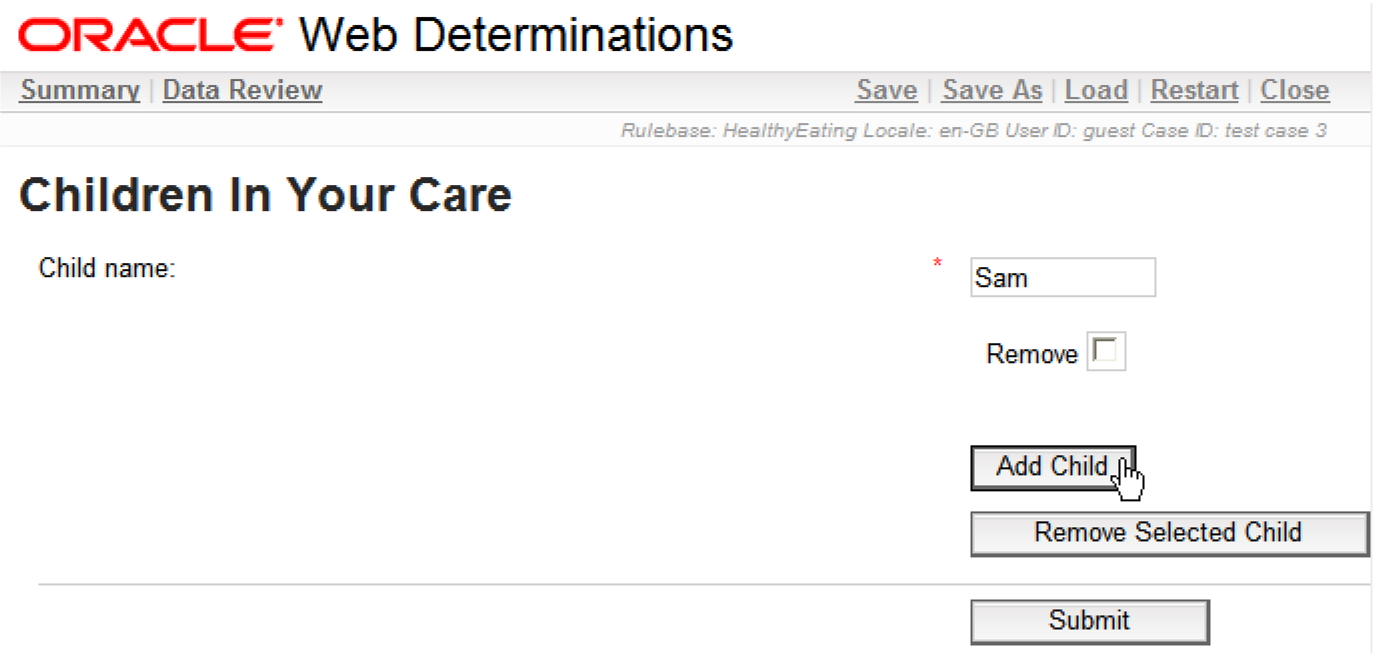
- The business is eligible for a Business Assistance Grant. [Why?]

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.)



Rulebase: HealthyEating Locale: en-GB User ID: guest Case ID: test case 3

Children In Your Care

Child name: * Sam

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.

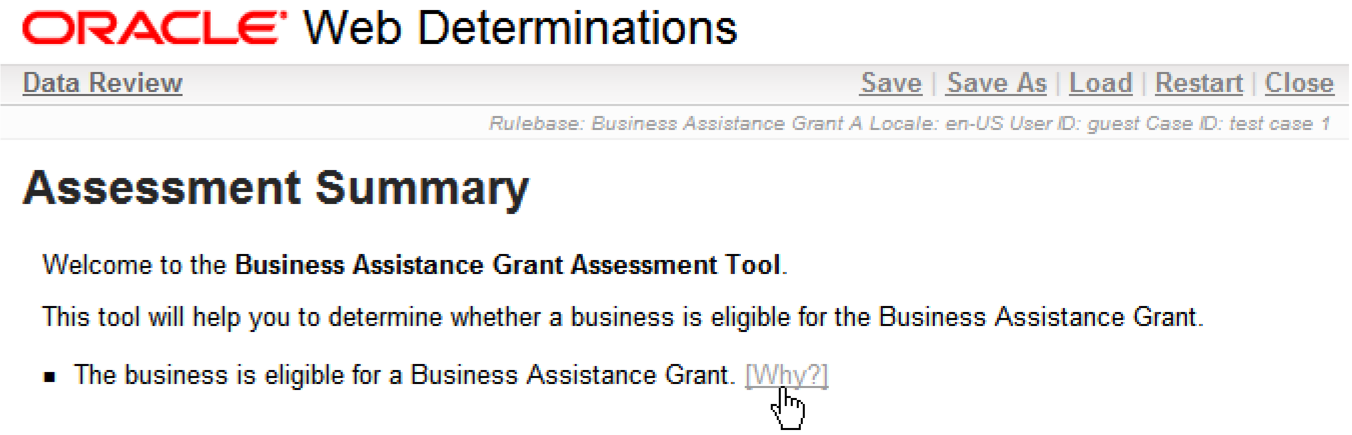
The screen that collects instances of 'the pet' uses 'the child's pets' as the relationship. Since this belongs to 'the child', the screen can appear once for every child that the user has entered. If the user enters three children then the screen for collecting pets must be displayed for all three children 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.



Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

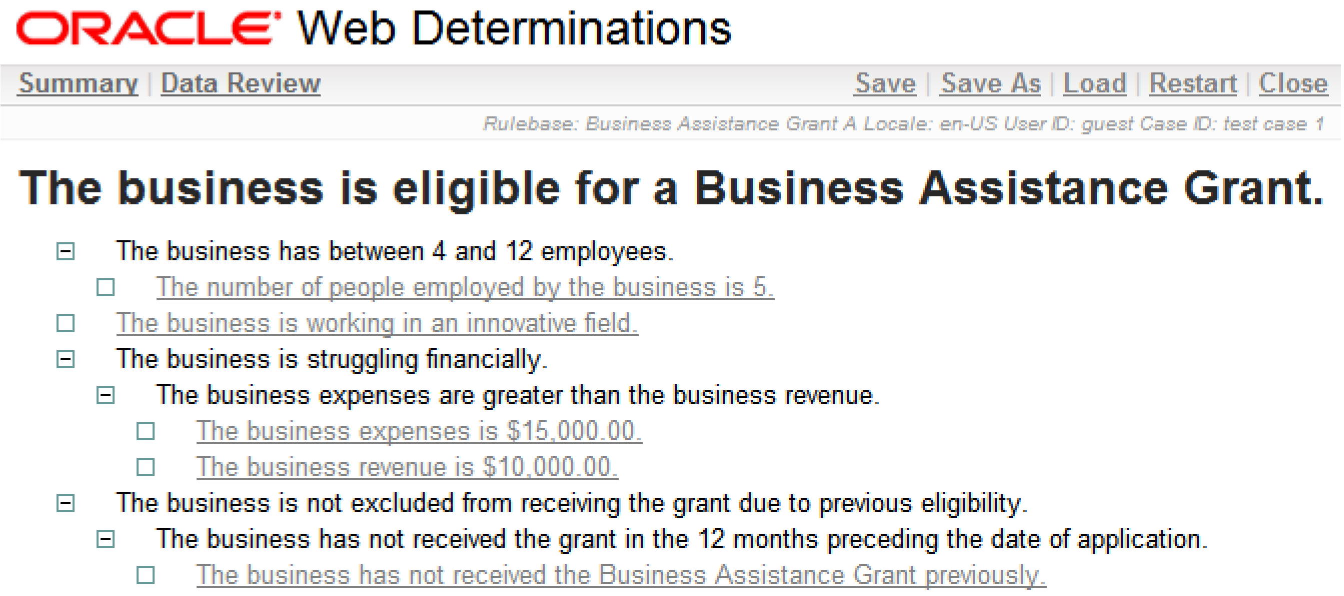
Assessment Summary

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:



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.

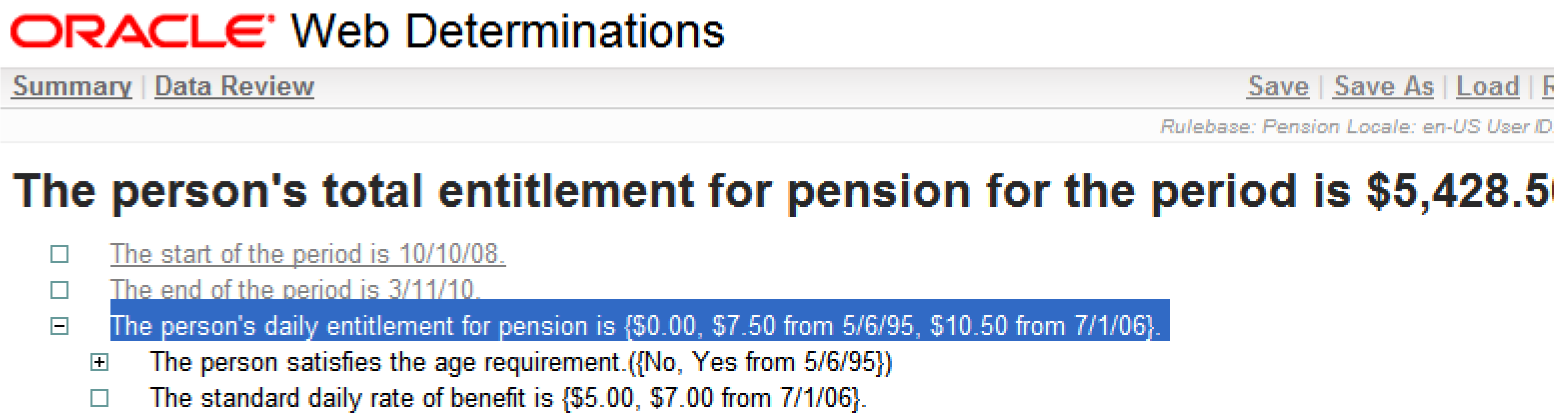
- ⊟ The business has between 4 and 12 employees.
  - ☐ The number of people employed by the business is 5.
- ☐ The business is working in an innovative field.
- ⊟ The business is struggling financially.
  - ⊟ The business expenses are greater than the business revenue.
    - ☐ The business expenses is $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:



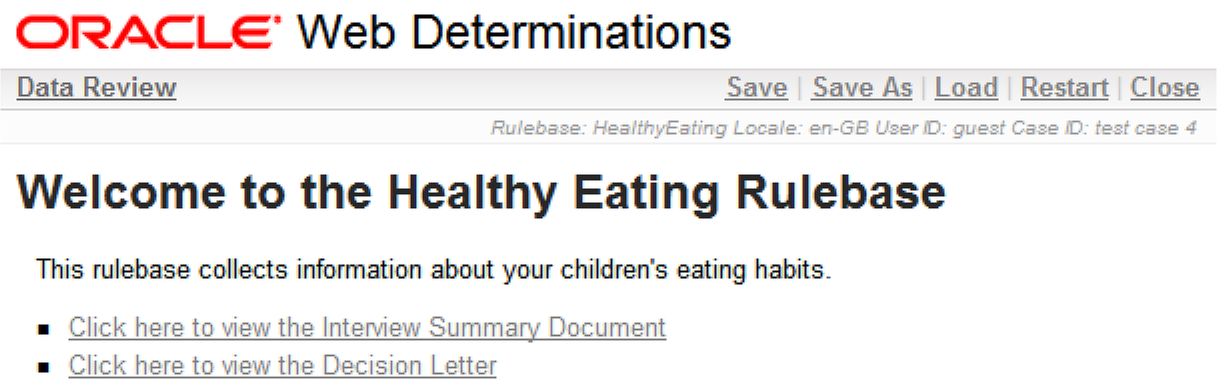
Rulebase: Pension Locale: en-US User ID

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.



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 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 be 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:

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.

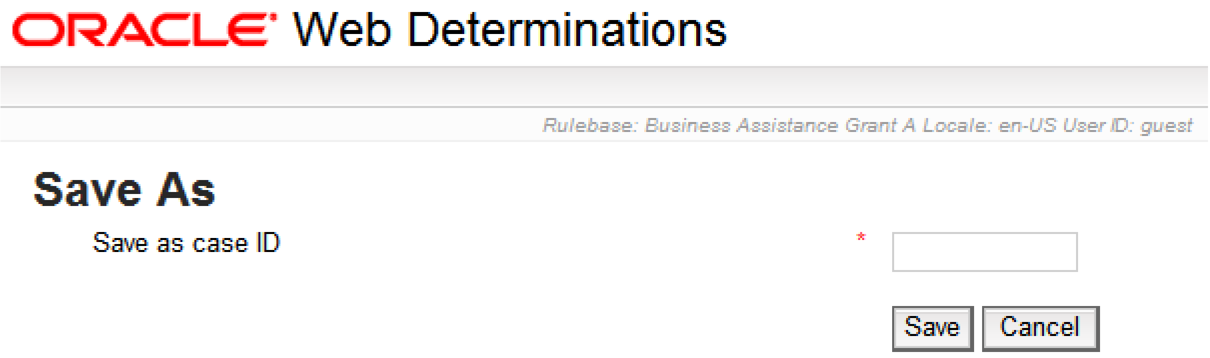
Save an interview in Web Determinations

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.

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.



Rulebase: Business Assistance Grant A Locale: en-US User ID: guest

Save as case ID *

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

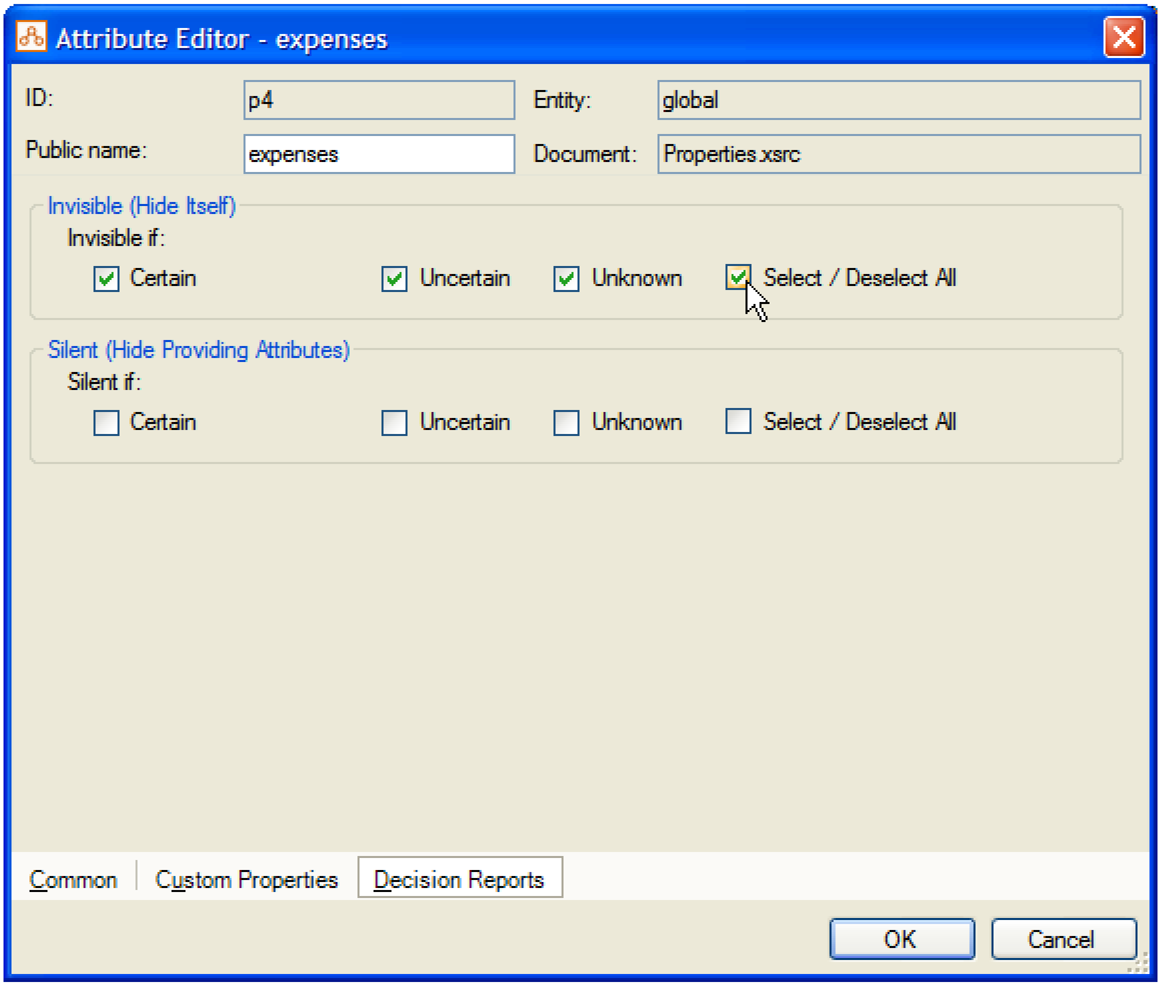
Add intermediate rules

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.



ID: p4

Entity: global

Public name: expenses

Document: Properties.xsrc

Invisible (Hide Itself)

Invisible if:

☑ Certain ☑ Uncertain ☑ Unknown ☑ Select / Deselect All

Silent (Hide Providing Attributes)

Silent if:

☐ Certain ☐ Uncertain ☐ Unknown ☐ Select / Deselect All

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

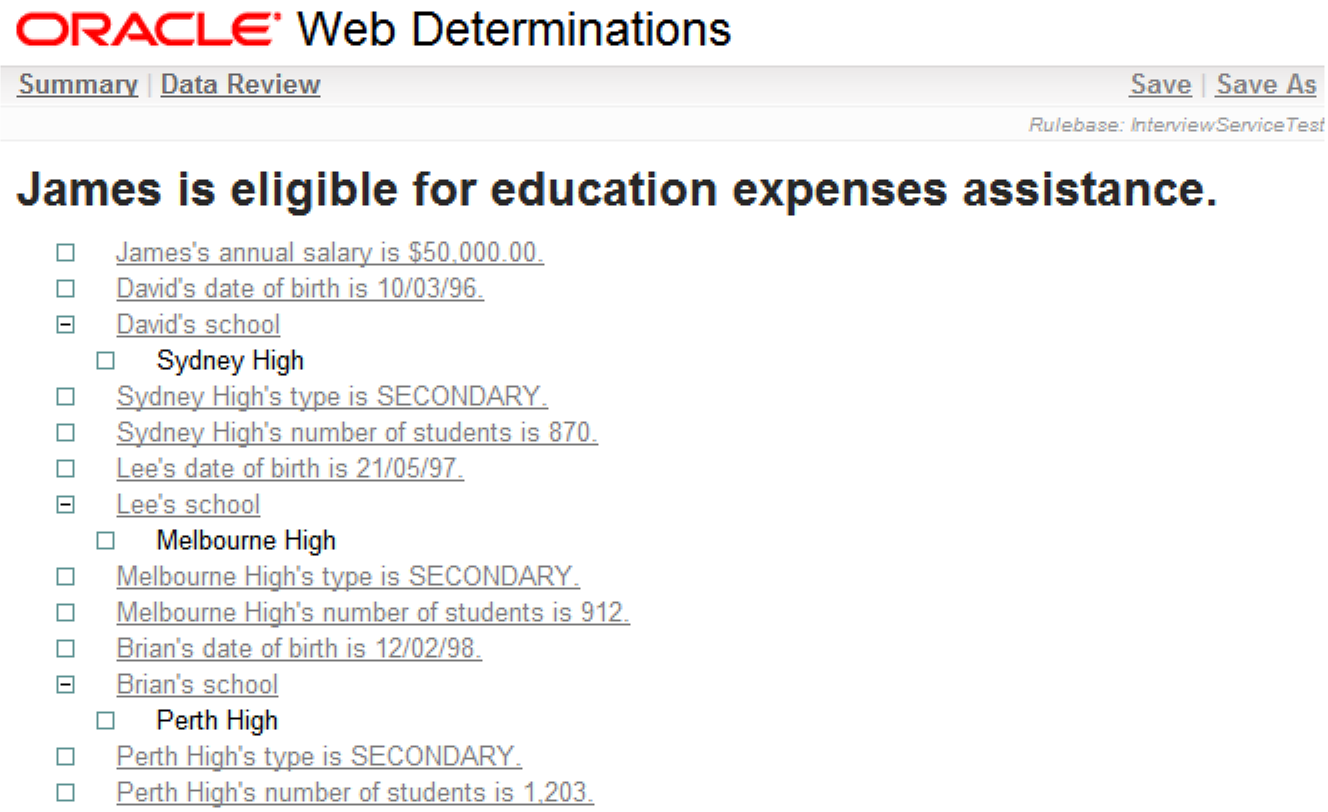
where "the claimant's sibling", "the claimant's sibling's country" and "the claimant" are all substituting variables.

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.)



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.

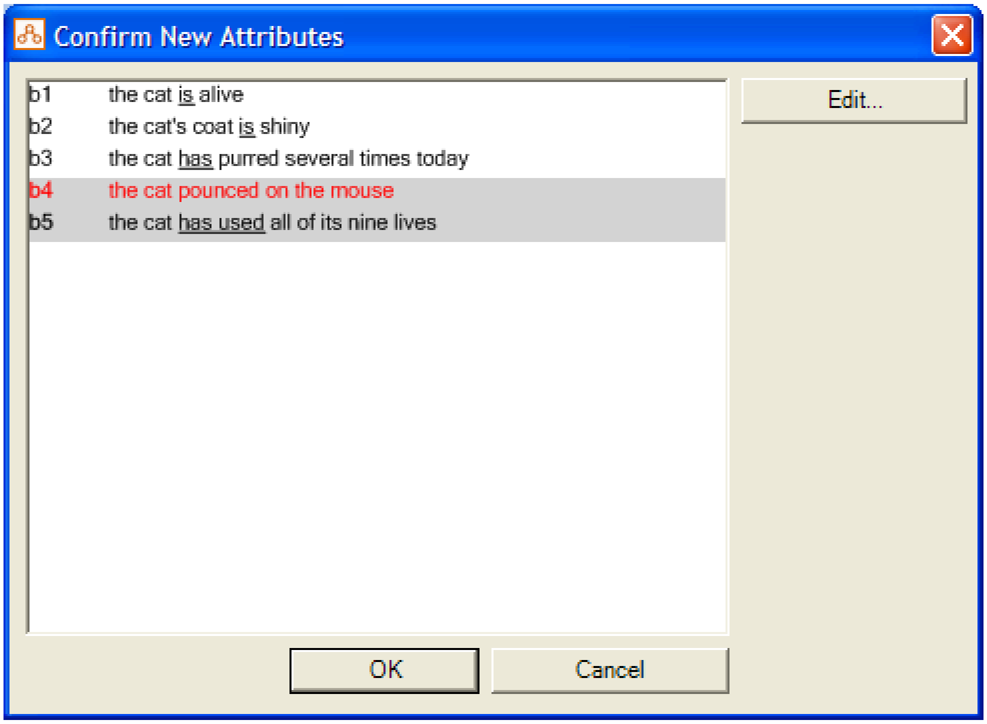
Review the attribute parses

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.



b1 the cat is alive
b2 the cat's coat is shiny
b3 the cat has purred several times today
b4 the cat pounced on the mouse
b5 the cat has used all of its nine lives

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)	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. If the underlined verb is not the correct verb around which the parse should be based, you need to select an alternate parse. 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 **Confirm New Attributes** dialog, select the attribute and click the **Edit** button. Select the parse in the top box to view the sentence forms for that parse in the box below.
Attributes shown in red (and also highlighted in gray)	The attribute does not contain a recognized verb and no sentence forms have been generated	Add the verb to the custom verbs list 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	Add the verb to the custom verbs list 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, a 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.

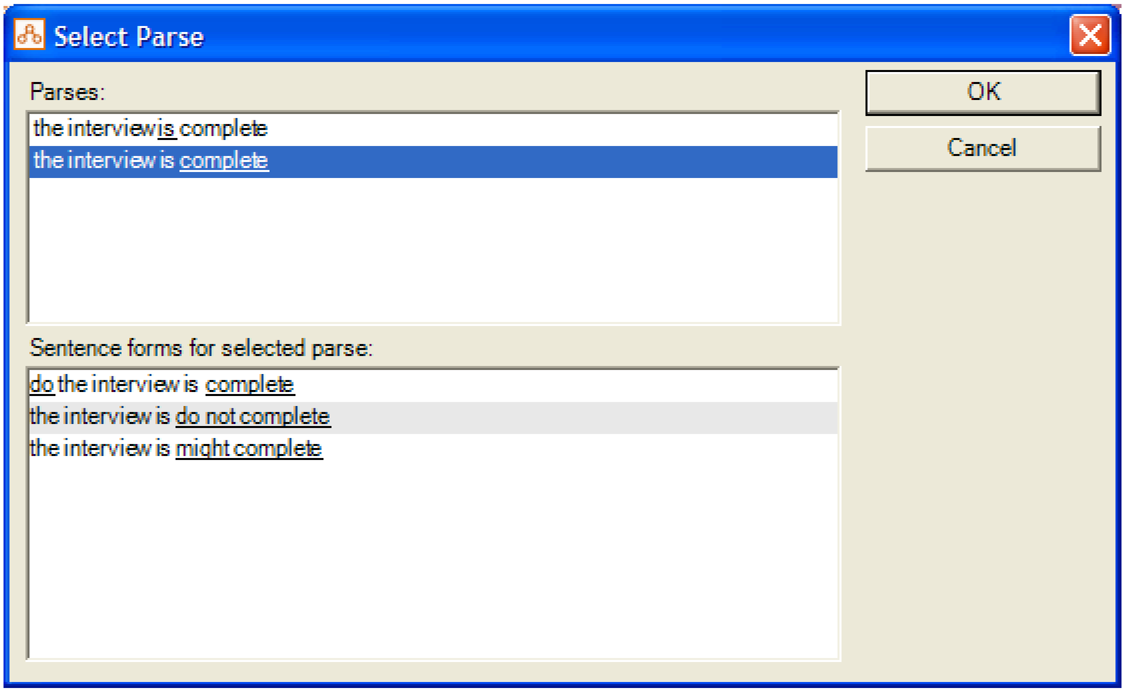
Show me an example of an attribute which needs to be reparsed

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



Parses:

the interview is complete

the interview is complete

Sentence forms for selected parse:

do the interview is complete

the interview is do not complete

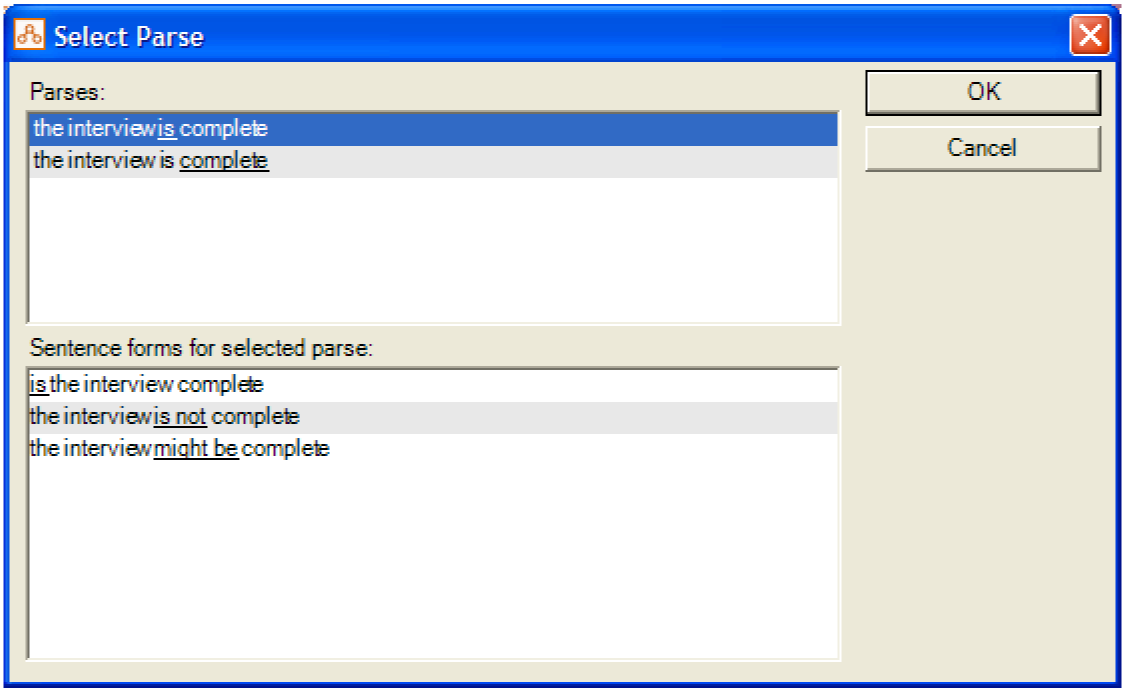
the interview is might complete

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



Parses:

the interview is complete

the interview is complete

Sentence forms for selected parse:

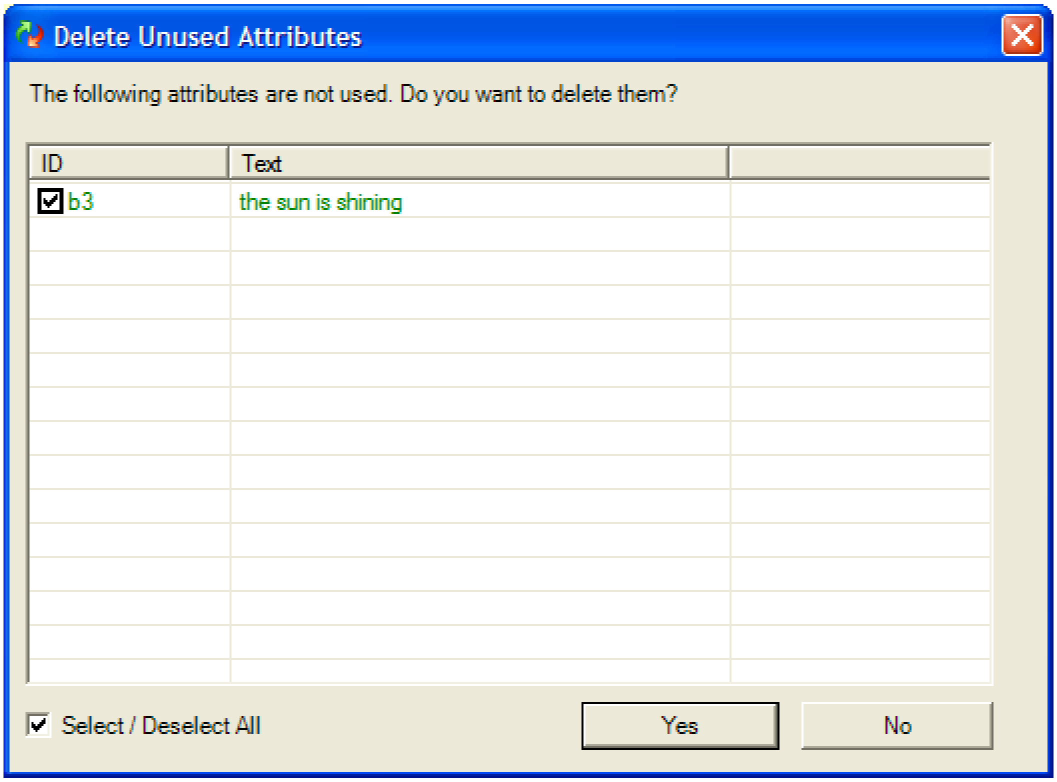
is the interview complete

the interview is not complete

the interview might be complete

Delete unused attributes

If there are any attributes in your Word or Excel document that have been added but are not used, or that were previously used but that are no longer used, you will be prompted to delete these attributes on compile:



The following attributes are not used. Do you want to delete them?

ID	Text	
☑ b3	the sun is shining	

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:

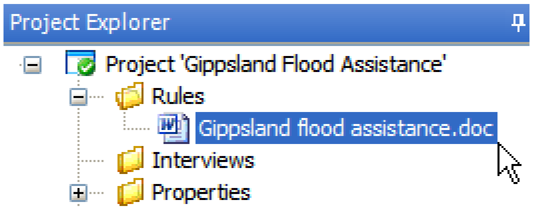
> **[b7] the claimant satisfies the Financial Qualification if**
>
> [b24] the claimant's weekly rent doubled is more than one half of the claimant's weekly net pay
>
> [(p2*2)>(p3/2)] (the claimant's weekly rent * 2) > (the claimant's weekly net pay / 2)

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.

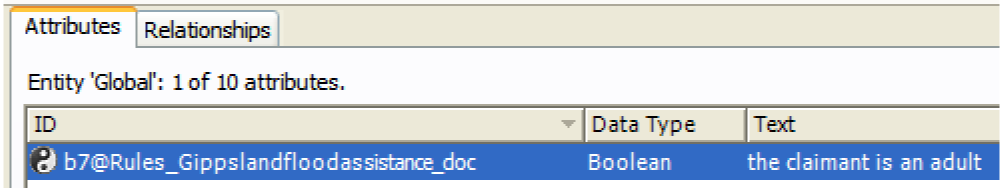
Because IDs are consecutively generated as b1, b2 etc on a per-document basis, Oracle Policy Modeling needs to distinguish between IDs from one rule document and another.

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:



Entity 'Global': 1 of 10 attributes.

ID	Data Type	Text
b7@Rules_Gippslandfloodassistance_doc	Boolean	the claimant is an adult

Customize document IDs

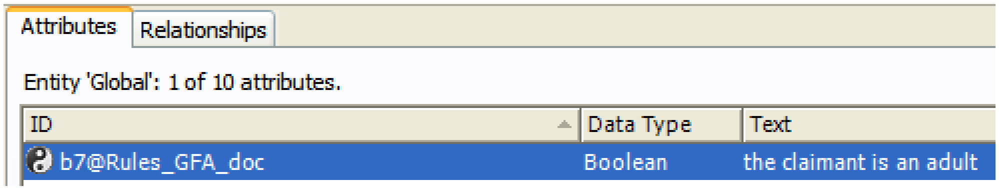
Documents with long file and folder names can become somewhat unwieldy. To avoid this problem, Oracle Policy Modeling allows you to rename the document ID used in the automatic generation of attribute IDs. To do this:

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:



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.

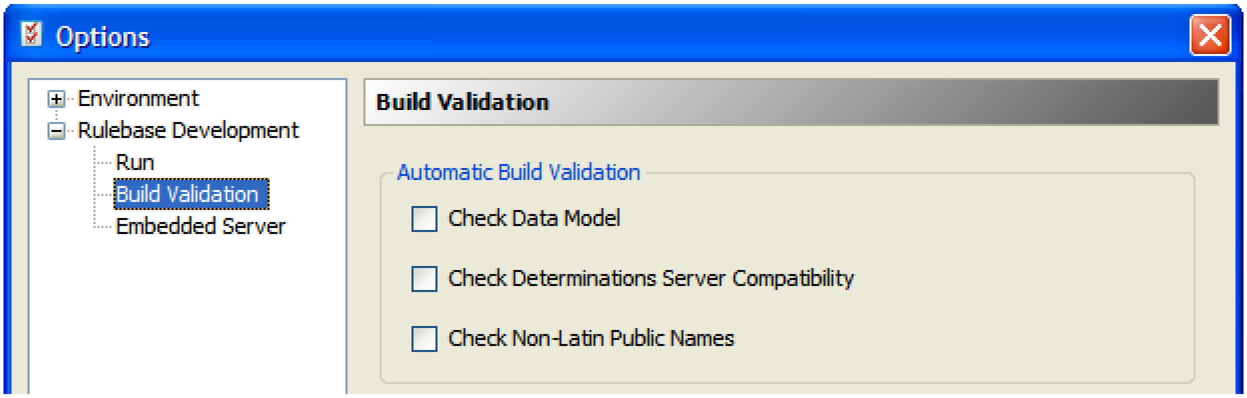
To build your rulebase, select **Build** from the Oracle Policy Modeling **Build** menu. A check will firstly be done to ensure that there are no source documents that need compilation. If there are you will be prompted to recompile these before continuing.

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**.



☐ Check Data Model

☐ Check Determinations Server Compatibility

☐ Check Non-Latin Public Names

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 reworded. 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

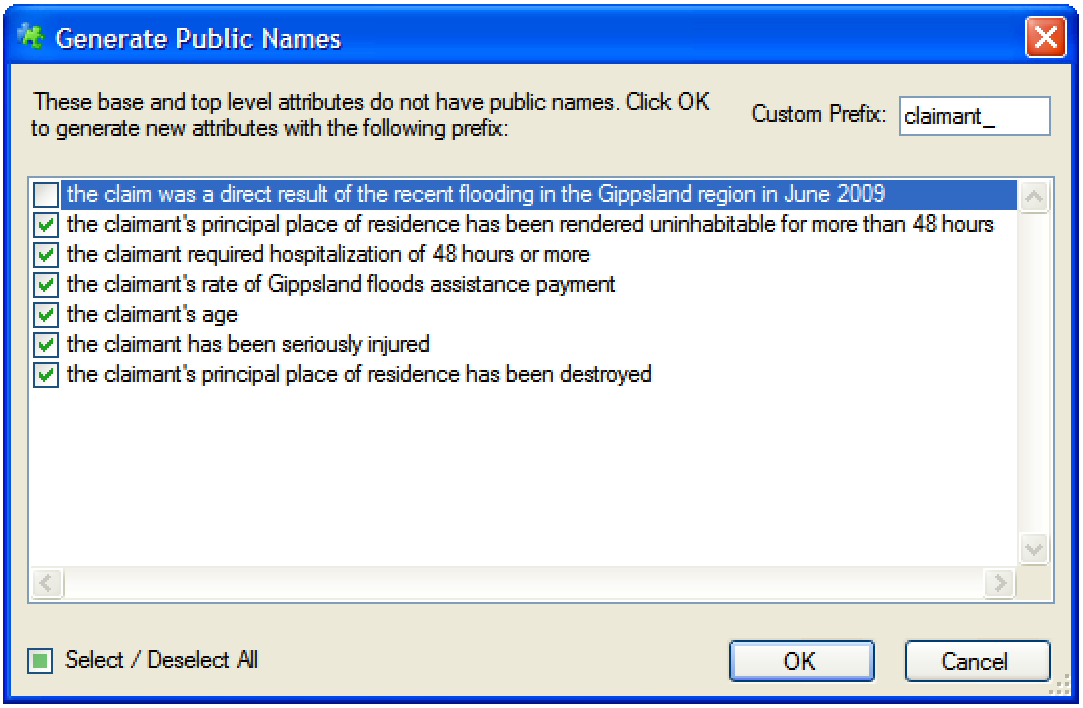
Maintain public names

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_").

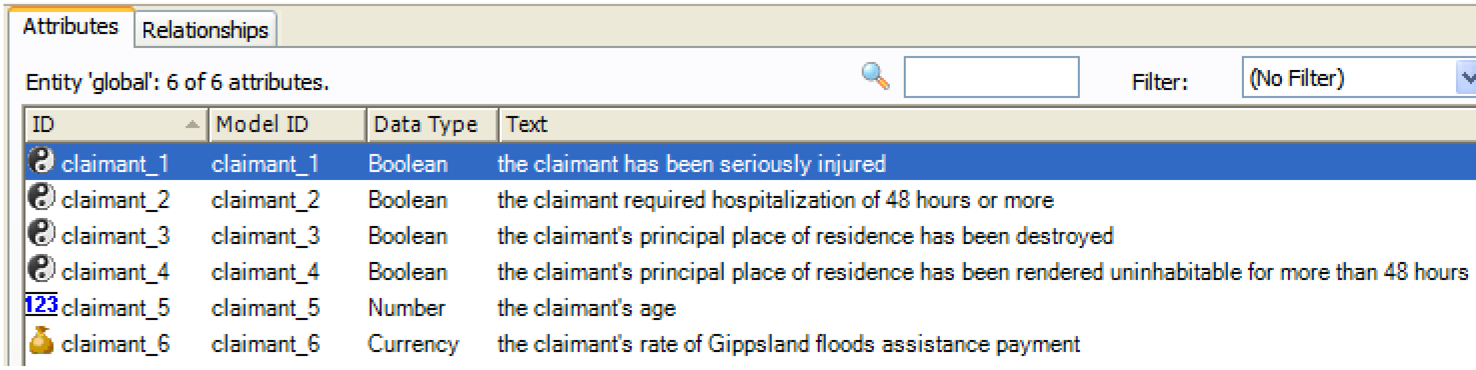


These base and top level attributes do not have public names. Click OK to generate new attributes with the following prefix:

Custom Prefix: claimant_

- ☐ the claim was a direct result of the recent flooding in the Gippsland region in June 2009
- ☑ the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours
- ☑ the claimant required hospitalization of 48 hours or more
- ☑ the claimant's rate of Gippsland floods assistance payment
- ☑ the claimant's age
- ☑ the claimant has been seriously injured
- ☑ the claimant's principal place of residence has been destroyed

5. Click **OK.** The Attributes list in the properties file will now show each attribute and its public name (ID).



Entity 'global': 6 of 6 attributes.

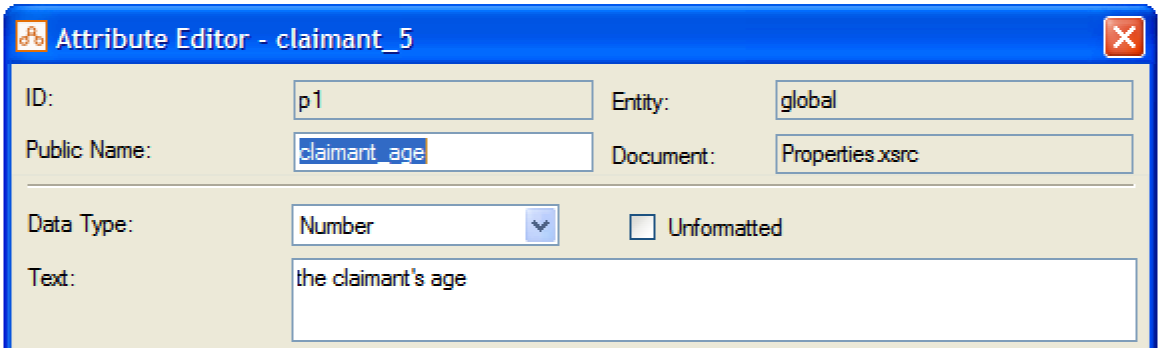
ID	Model ID	Data Type	Text
claimant_1	claimant_1	Boolean	the claimant has been seriously injured
claimant_2	claimant_2	Boolean	the claimant required hospitalization of 48 hours or more
claimant_3	claimant_3	Boolean	the claimant's principal place of residence has been destroyed
claimant_4	claimant_4	Boolean	the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours
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).



ID: p1

Entity: global

Public Name: claimant_age

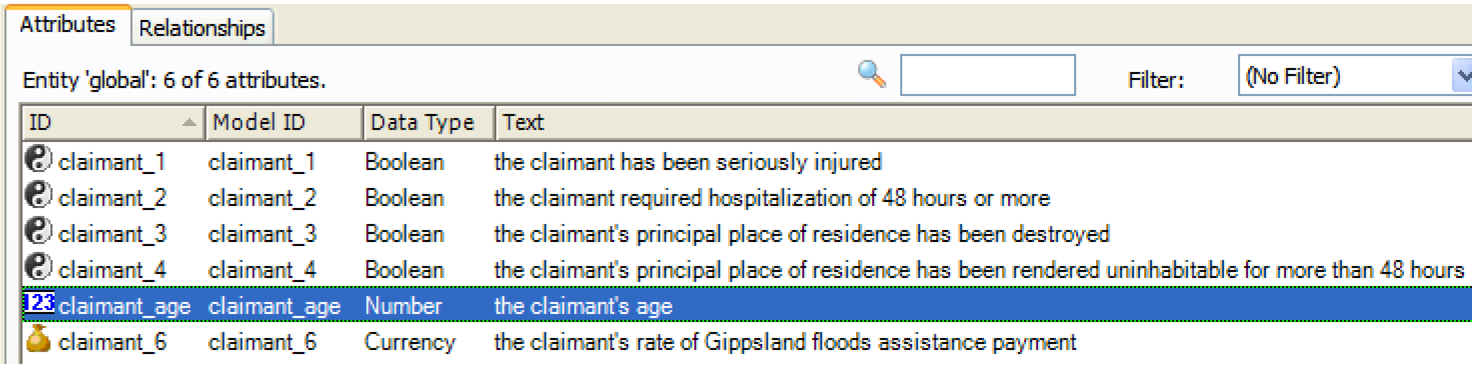
Document: Properties.xsrc

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.



Entity 'global': 6 of 6 attributes.

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
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, "hhm_" 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. 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.

b. 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:

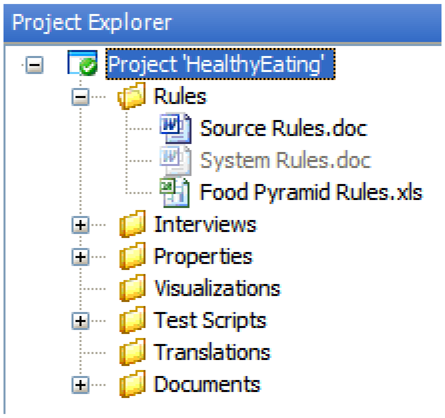
1. Right-click anywhere in the Error List window.
2. Select **Copy List**.
3. Open Microsoft Excel or Word (or another authoring tool) and paste the list.
4. Save the list.

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
Build Options	
-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
Validation Options	
-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
Report Options	
-cd	Analyzes a *.coverage file and produces a document-oriented report (.xml)
-cg	Analyzes a *.coverage file and produces a goal-oriented report(.xml)
Upgrade Options	
-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 *-upgrade* flag. When set, write-protection will be removed for read-only project files. When not set, read-only project files will still be copied to the upgraded project directory but won't be processed.
Help Options	

Parameter	Description
-h	Prints the help message
--diagnostics	Generates diagnostic information. The project path and other parameters are ignored.

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
```

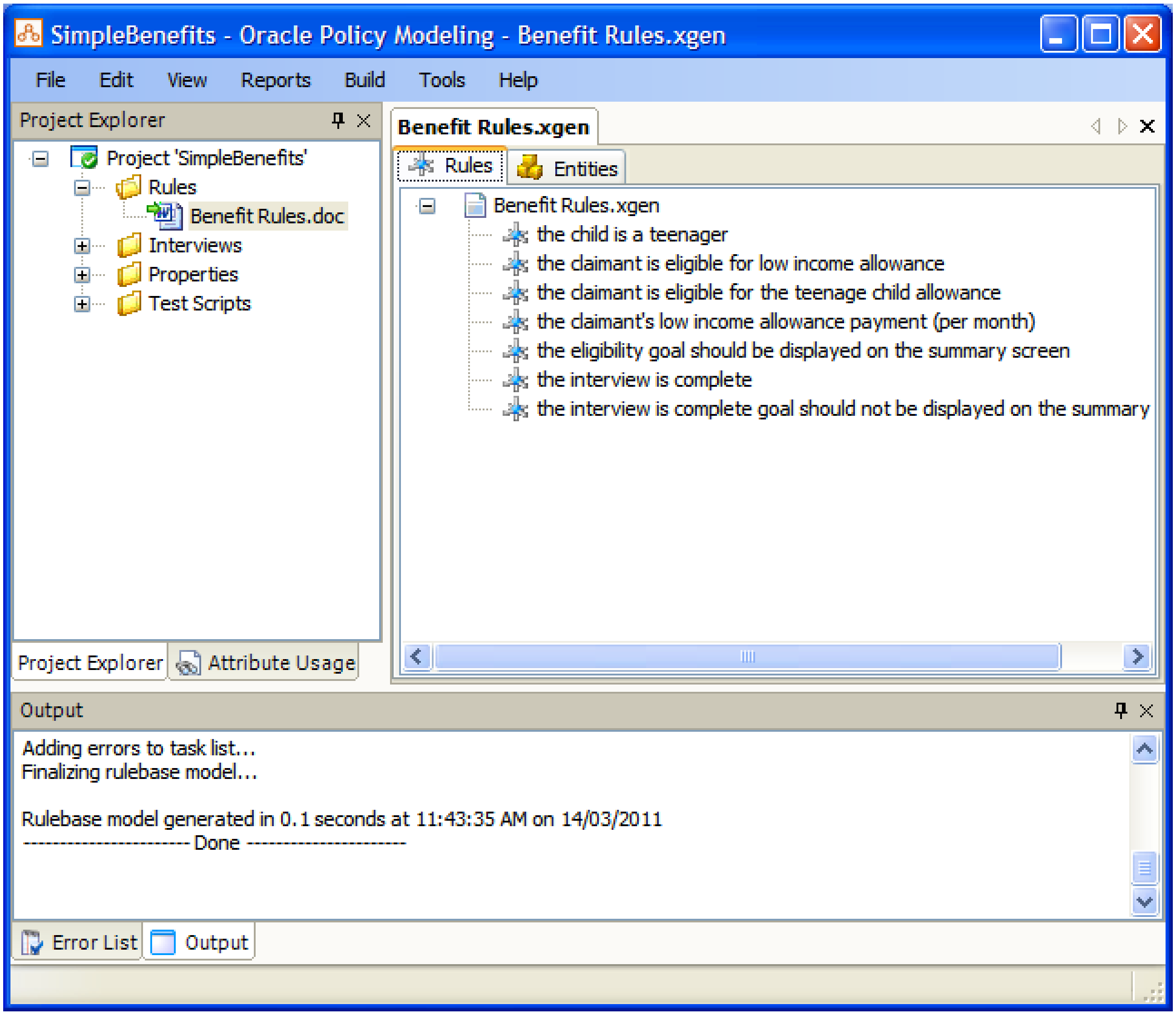
Finding and reporting

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.



```
Adding errors to task list...
Finalizing rulebase model...

Rulebase model generated in 0.1 seconds at 11:43:35 AM on 14/03/2011
---------------------- Done ---------------------
```

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 create a new project
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 import a project
Export...	Opens the Export Project dialog box where you select the destination and file name for the project interchange file which is used the export the project 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 statistics 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 Tools | Options 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 attribute in the build model
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 find a particular screen 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 find rules that use a particular attribute.
Build Model	Opens the Build Model view in the top right hand pane. This view shows the full list of entities and attributes 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 seeing the structure of a rule and for finding the input data needed to reach a particular conclusion.
Data Model	Opens the Data Model view in the top right hand pane. This shows the rulebase data model 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 errors 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 Logical Loop Check report 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 Multiply Proven Attributes report 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 Top Level Attributes report 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 Base Level Attributes report 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 Uncollected Attributes report 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 attributes that are collected on more than one question screen.
Dependent Base Attributes	Generates a Dependent Base Attributes report 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 spell-checking the screens.
Untranslated Text	Generates an Untranslated Text 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 Custom Properties Report then opens in the right hand pane.
Inferred Screen Attributes	Generates an Inferred Screen Attributes report 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. Test Specifications for each selected script are then displayed in the right hand pane.
Run Multiple Test Scripts...	Opens the Run Multiple Test Scripts 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 coverage 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 analyze 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 Project Properties.
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 Oracle Web Determinations or Oracle Determinations Server.
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 link to this module.
Generate Commentary Files...	Opens the Generate Commentary Files dialog box. This is used to specify the settings for the automatic generation of commentary files.

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	Compiles 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 attribute references that need repairing. 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 update of all Oracle Policy Modeling templates

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 Function Reference 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?

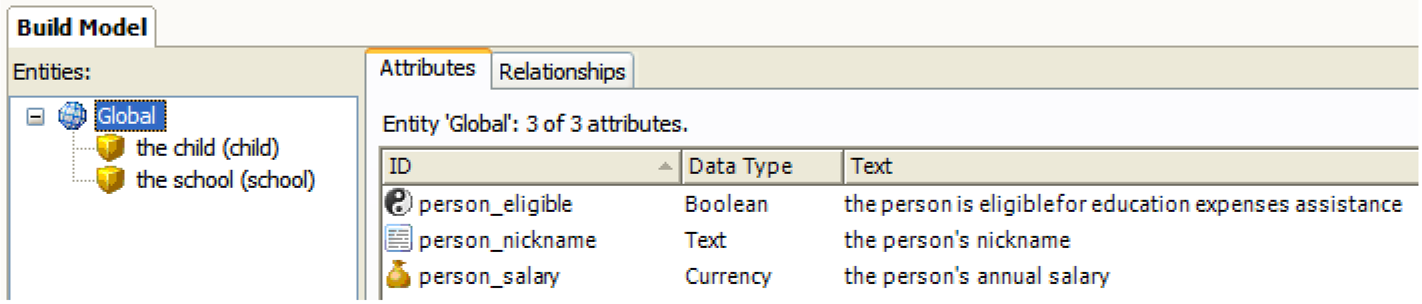
View the entities and attributes in the build model

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

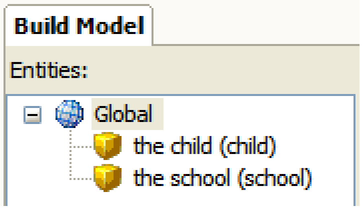
1. To open the build model view, go to **View | Build Model**.



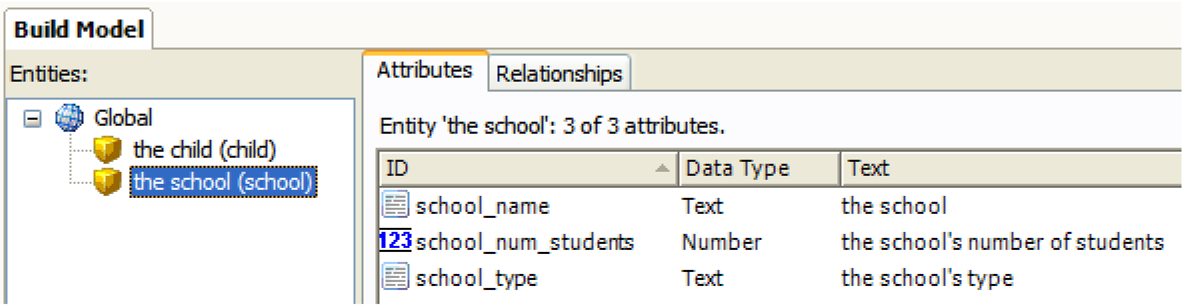
Entity 'Global': 3 of 3 attributes.

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

2. The left hand pane in the build model shows the entities in the rulebase.



3. 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.

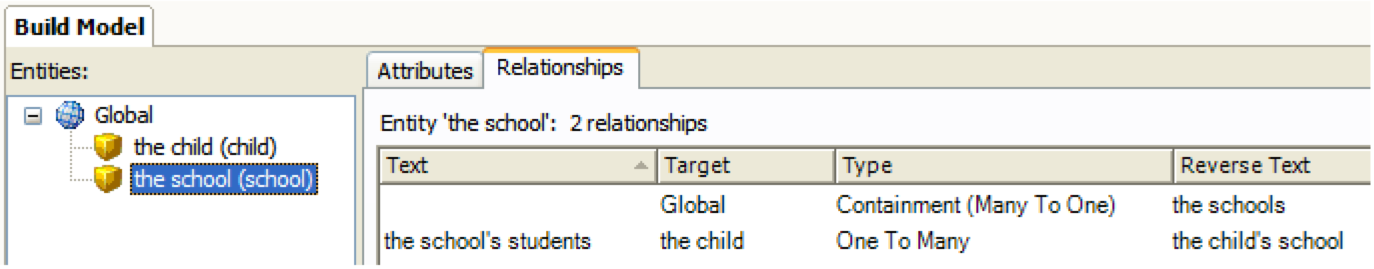


Entity 'the school': 3 of 3 attributes.

ID	Data Type	Text
school_name	Text	the school
school_num_students	Number	the school's number of students
school_type	Text	the school's type

(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.

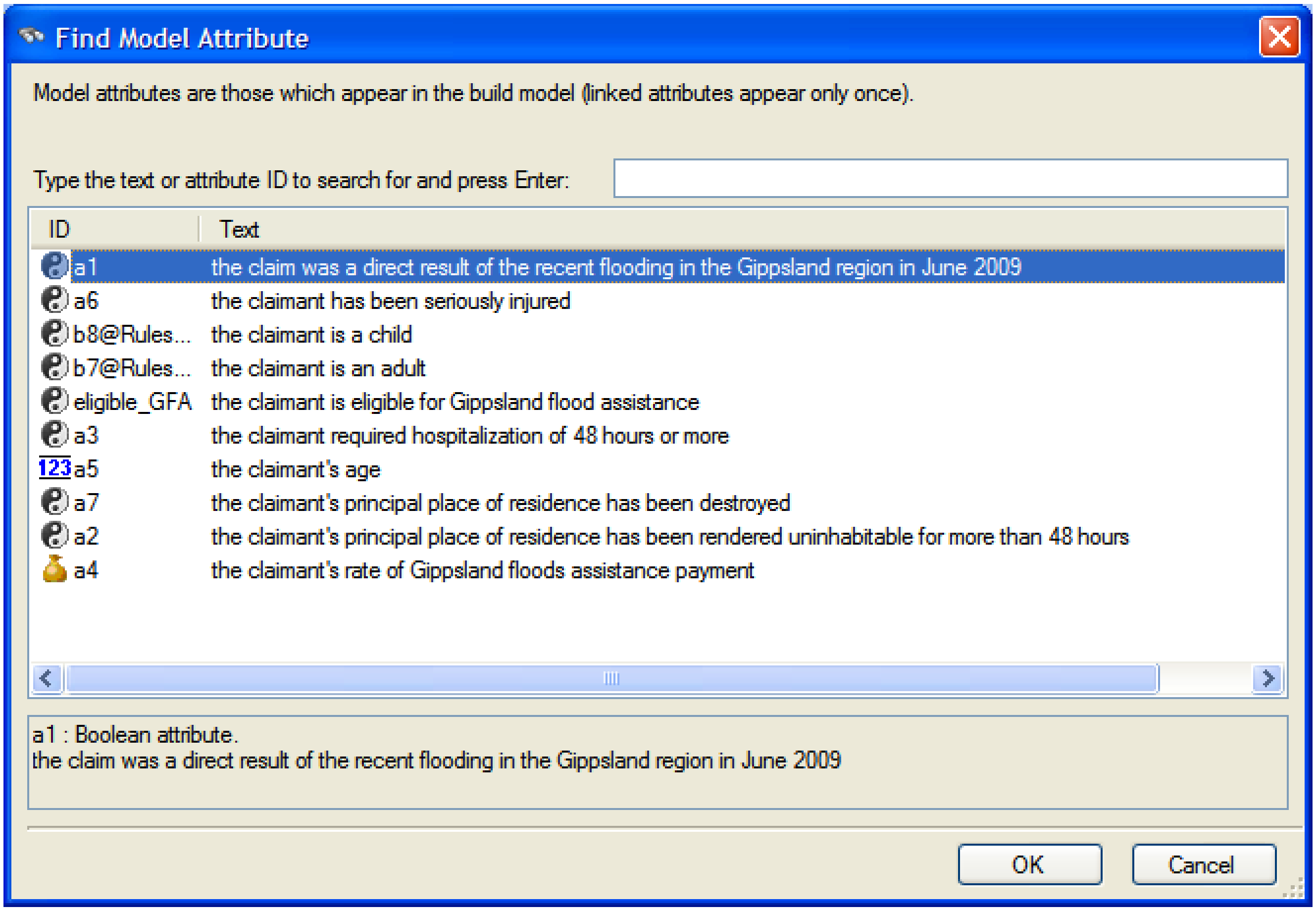


Entity 'the school': 2 relationships

Text	Target	Type	Reverse Text
	Global	Containment (Many To One)	the schools
the school's students	the child	One To Many	the child's school

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...**



Model attributes are those which appear in the build model (linked attributes appear only once).

Type the text or attribute ID to search for and press Enter:

ID	Text
a1	the claim was a direct result of the recent flooding in the Gippsland region in June 2009
a6	the claimant has been seriously injured
b8@Rules...	the claimant is a child
b7@Rules...	the claimant is an adult
eligible_GFA	the claimant is eligible for Gippsland flood assistance
a3	the claimant required hospitalization of 48 hours or more
a5	the claimant's age
a7	the claimant's principal place of residence has been destroyed
a2	the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours
a4	the claimant's rate of Gippsland floods assistance payment

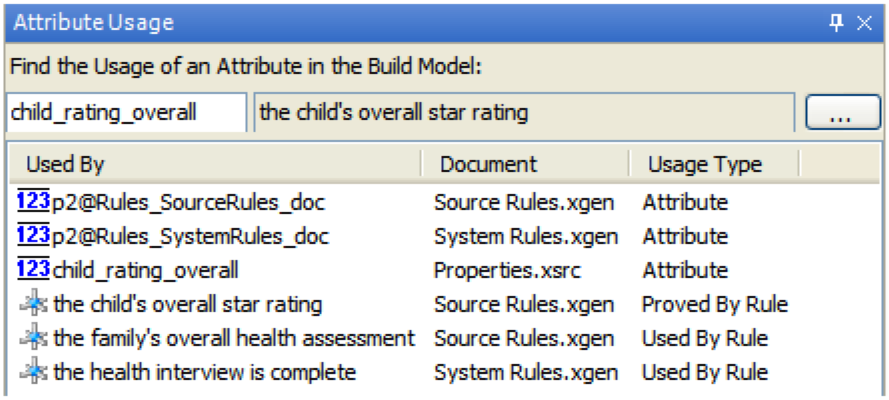
a1 : Boolean attribute.
the claim was a direct result of the recent flooding in the Gippsland region in June 2009

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 Usage of an Attribute in the Build Model:

child_rating_overall | the child's overall star rating

Used By	Document	Usage Type
123 p2@Rules_SourceRules_doc	Source Rules.xgen	Attribute
123 p2@Rules_SystemRules_doc	System Rules.xgen	Attribute
123 child_rating_overall	Properties.xsrc	Attribute
the child's overall star rating	Source Rules.xgen	Proved By Rule
the family's overall health assessment	Source Rules.xgen	Used By Rule
the health interview is complete	System Rules.xgen	Used By Rule

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:

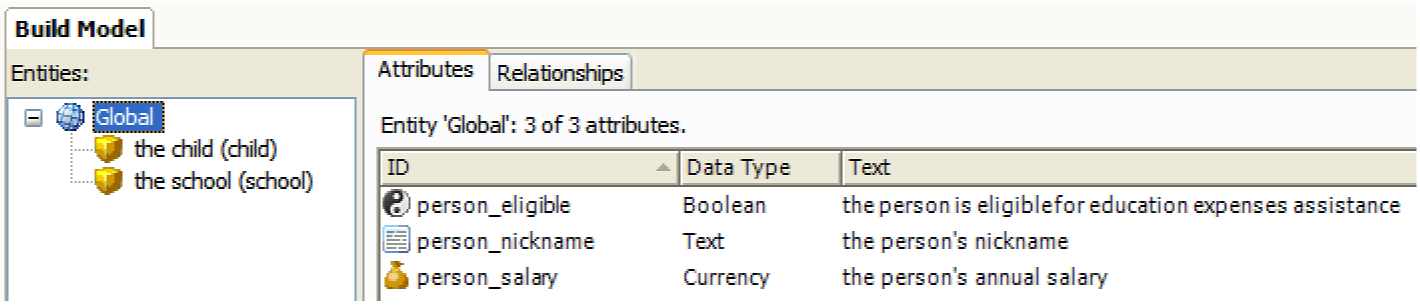
Attribute text	Entity level	Explanation
the household member is male	the household member	contains "the household member"
a household member is eligible	global	"a household member" does not match "the household member"
the former household member has left	global	"former" interrupts the attribute text
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**.

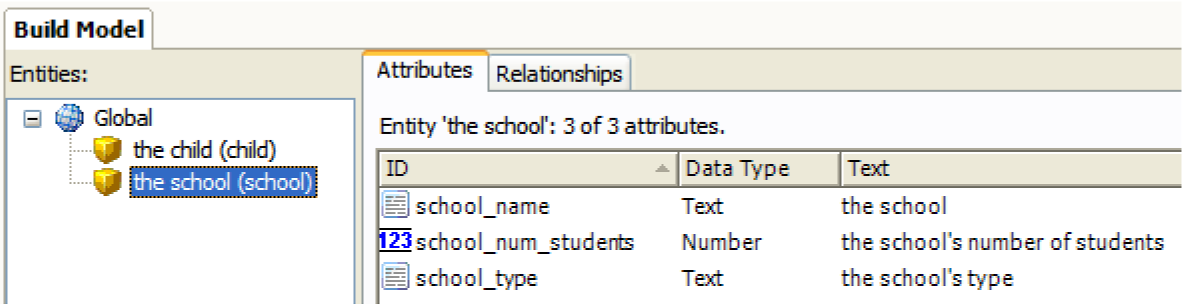
Attributes which are not within the scope of an entity are placed in the Global level. The list of global attributes are displayed in the right-hand pane:



Entity 'Global': 3 of 3 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:



Entity 'the school': 3 of 3 attributes.

ID	Data Type	Text
school_name	Text	the school
school_num_students	Number	the school's number of students
school_type	Text	the school's type

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.

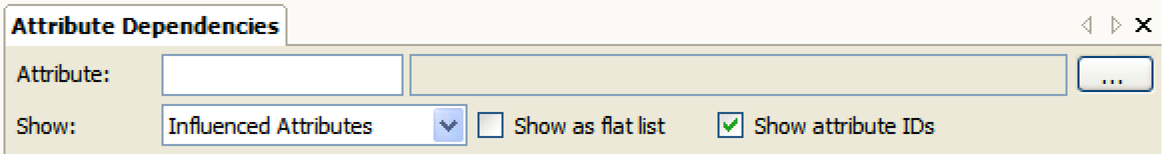
See also:

- 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.

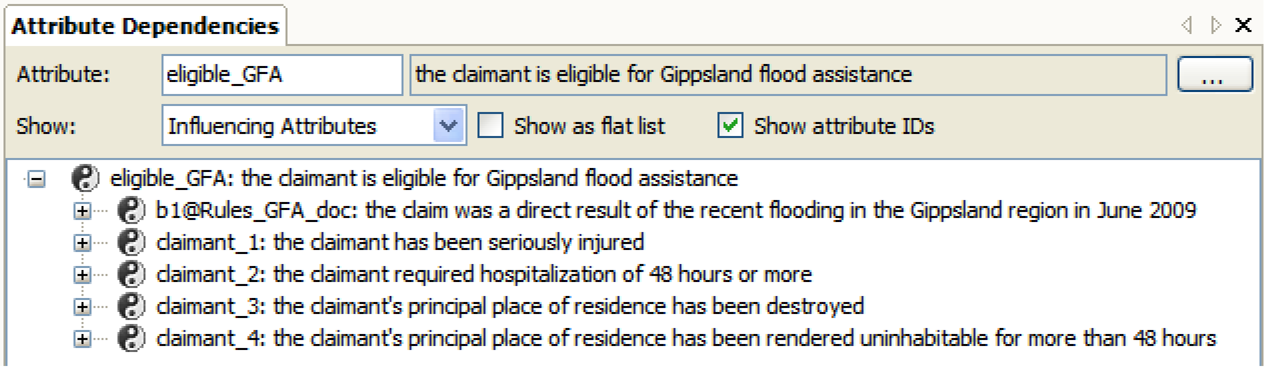


TIP: If your rulebase is very large, searching for an attribute in the Attribute Selector will be quicker if you turn off the **Filter search results on each keystroke** option under **File | Project Properties | Common Properties | General**.

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.



Attribute: eligible_GFA the claimant is eligible for Gippsland flood assistance

Show: Influencing Attributes ☐ Show as flat list ☑ Show attribute IDs

- eligible_GFA: the claimant is eligible for Gippsland flood assistance
  - b1@Rules_GFA_doc: the claim was a direct result of the recent flooding in the Gippsland region in June 2009
  - claimant_1: the claimant has been seriously injured
  - claimant_2: the claimant required hospitalization of 48 hours or more
  - claimant_3: the claimant's principal place of residence has been destroyed
  - claimant_4: the claimant's principal place of residence has been rendered uninhabitable for more than 48 hours

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

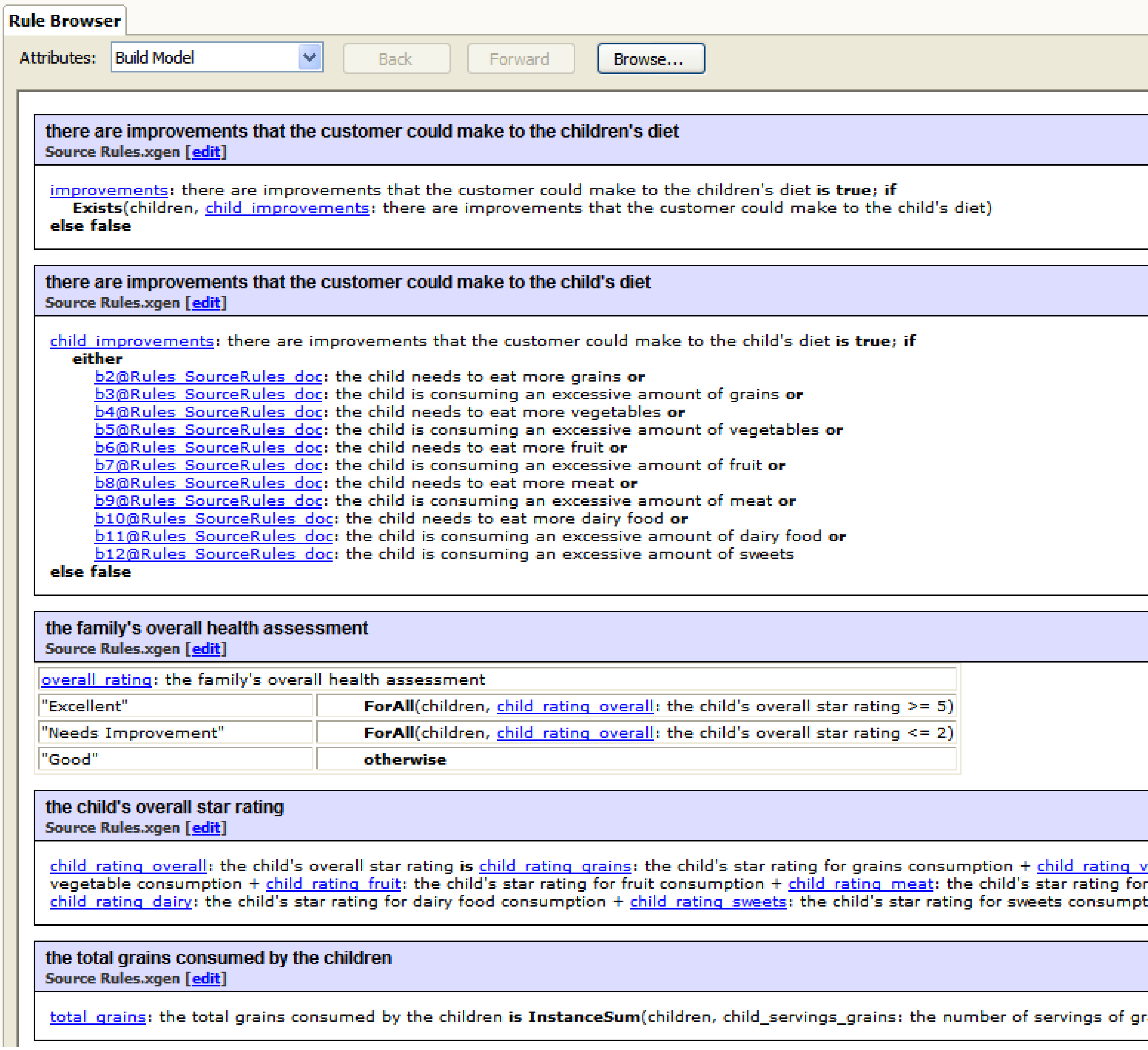
The Rule Browser is a helpful way to understand the links between rules across your rule documents.

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.



Attributes: Build Model

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]

overall_rating: the family's overall health assessment	
"Excellent"	**ForAll**(children, child_rating_overall: the child's overall star rating >= 5)
"Needs Improvement"	**ForAll**(children, child_rating_overall: the child's overall star rating <= 2)
"Good"	**otherwise**

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 consumpt

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 gr

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

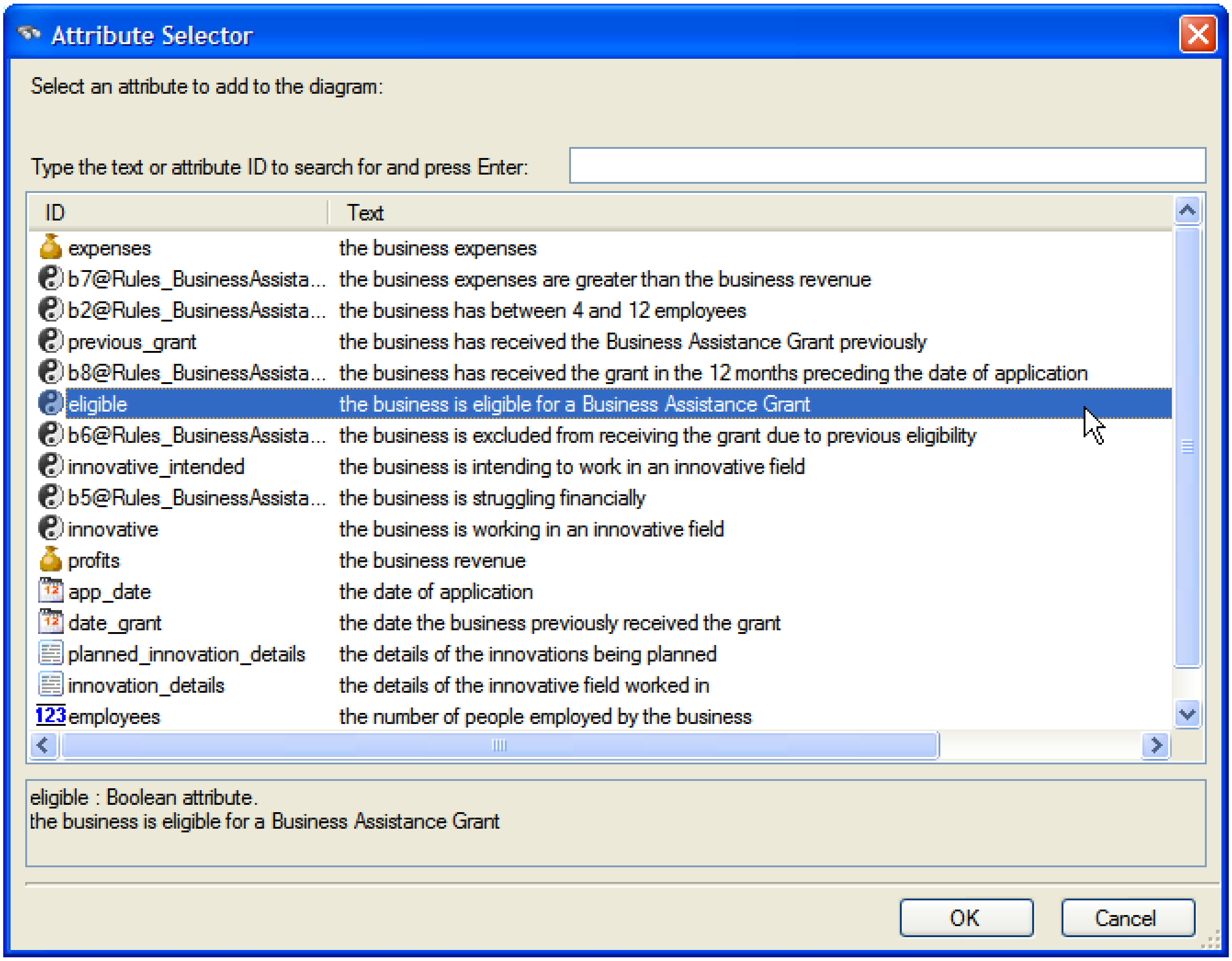
To add a new visual browser file to your project:

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**.



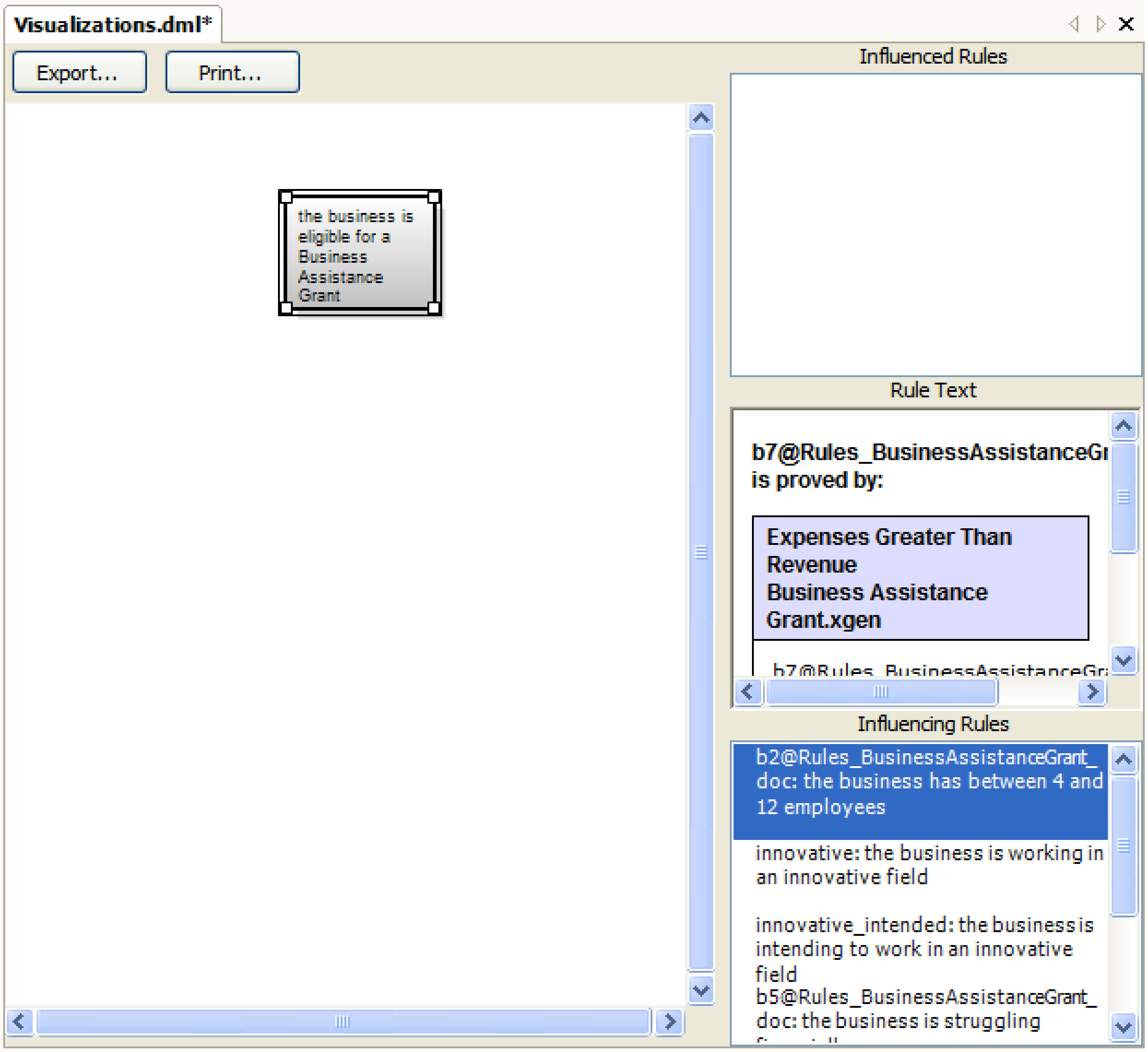
Select an attribute to add to the diagram:

Type the text or attribute ID to search for and press Enter:

ID	Text
expenses	the business expenses
b7@Rules_BusinessAssista...	the business expenses are greater than the business revenue
b2@Rules_BusinessAssista...	the business has between 4 and 12 employees
previous_grant	the business has received the Business Assistance Grant previously
b8@Rules_BusinessAssista...	the business has received the grant in the 12 months preceding the date of application
eligible	the business is eligible for a Business Assistance Grant
b6@Rules_BusinessAssista...	the business is excluded from receiving the grant due to previous eligibility
innovative_intended	the business is intending to work in an innovative field
b5@Rules_BusinessAssista...	the business is struggling financially
innovative	the business is working in an innovative field
profits	the business revenue
app_date	the date of application
date_grant	the date the business previously received the grant
planned_innovation_details	the details of the innovations being planned
innovation_details	the details of the innovative field worked in
employees	the number of people employed by the business

eligible : Boolean attribute.
the business is eligible for a Business Assistance Grant

The attribute will be added as a node to the left hand pane:



b7@Rules_BusinessAssistanceGr
is proved by:

Expenses Greater Than Revenue Business Assistance Grant.xgen

b7@Rules_BusinessAssistanceGr

b2@Rules_BusinessAssistanceGrant_ doc: the business has between 4 and 12 employees

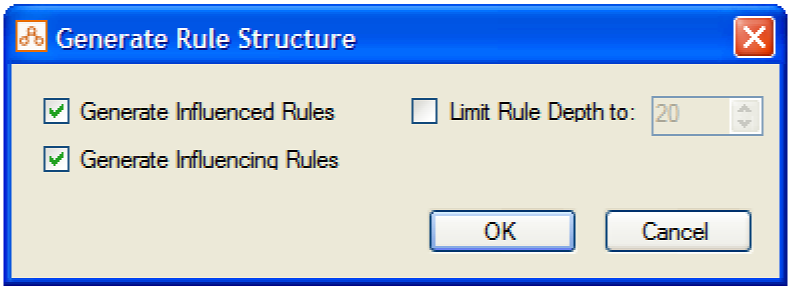
innovative: the business is working in an innovative field

innovative_intended: the business is intending to work in an innovative field

b5@Rules_BusinessAssistanceGrant_ doc: the business is struggling

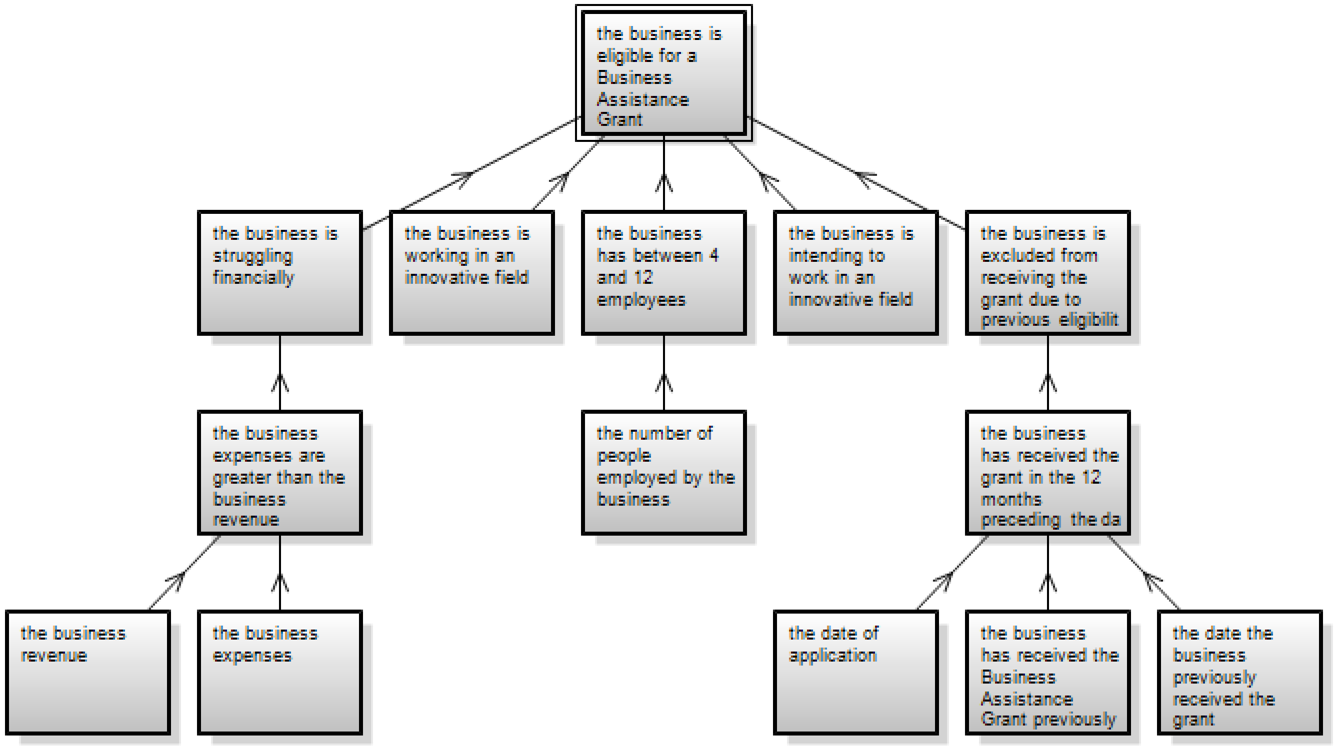
Note that the influenced/influencing rules and rule text for that attribute are displayed in the right hand boxes.

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.



☑ Generate Influenced Rules
☐ Limit Rule Depth to: 20
☑ Generate Influencing Rules

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.

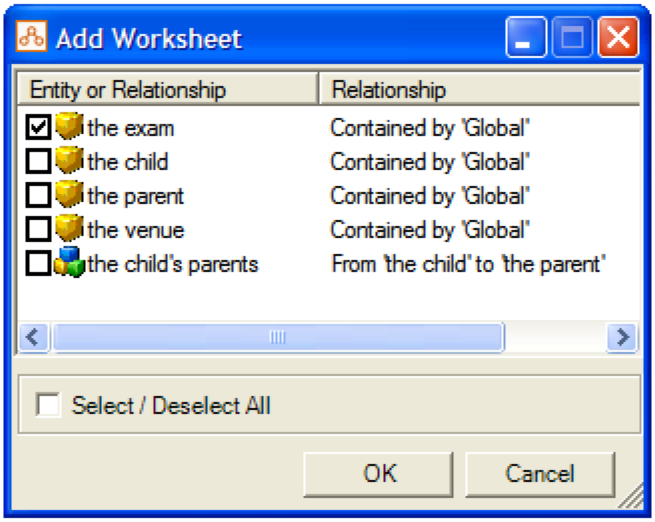
In the Project Explorer in Oracle Policy Modeling, double-click the what-if analysis document to open it in Excel.

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:

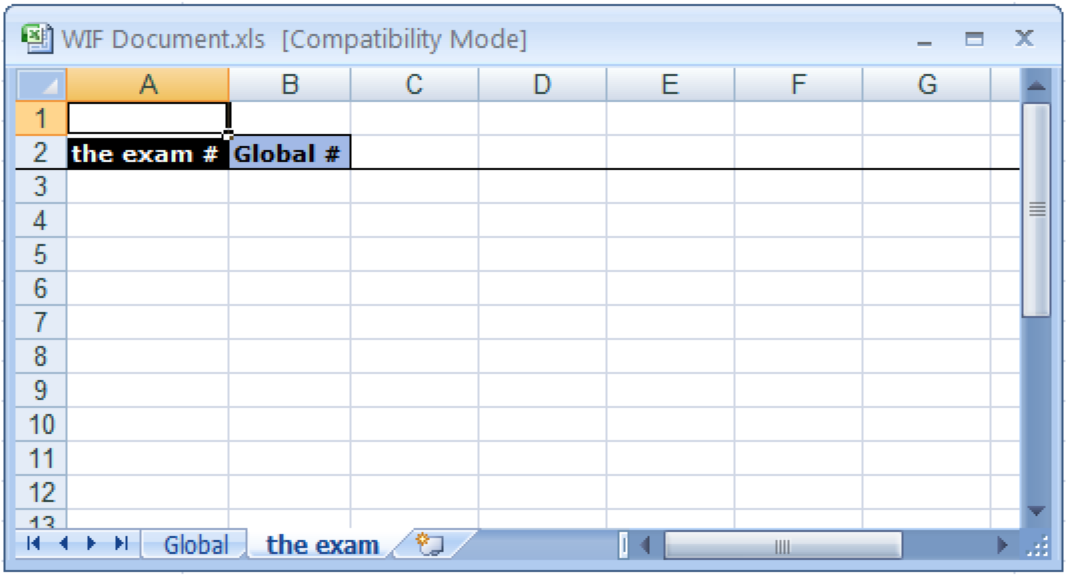
1. On the Oracle Policy Modeling toolbar, select the **Add Worksheet** button.
2. In the **Add Worksheet** dialog, select the checkbox for each entity or many-to-many relationship that you want to add. (Note that only those entities and many-to-many relationships that do not already exist as worksheets 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.)



Entity or Relationship	Relationship
☑ the exam	Contained by 'Global'
☐ the child	Contained by 'Global'
☐ the parent	Contained by 'Global'
☐ the venue	Contained by 'Global'
☐ the child's parents	From 'the child' to 'the parent'

☐ Select / Deselect All

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 #).



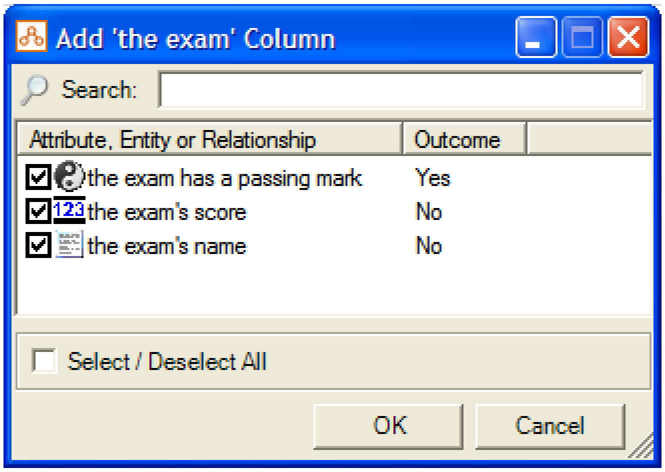
A	B
the exam #	Global #

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:

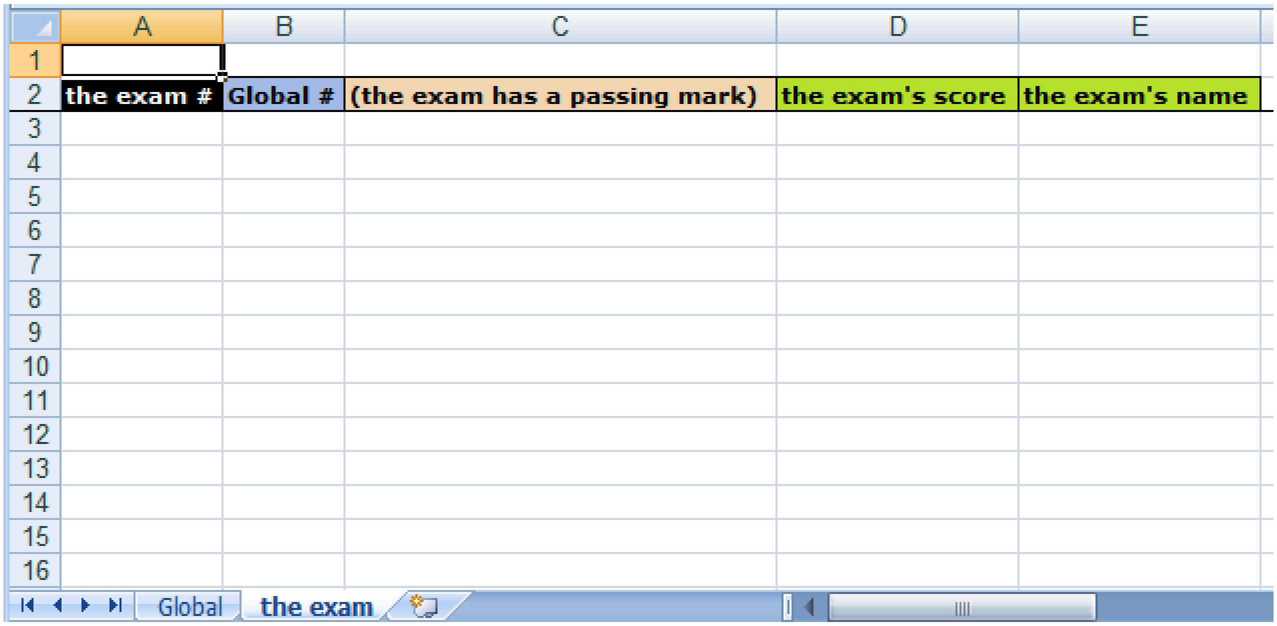
1. Select the tab for the entity that the attribute or relationship relates to. (For one-to-many and many-to-one relationships, the relationship column is added to the entity on the many side of the relationship.)
2. On the Oracle Policy Modeling toolbar, select the **Add Column** button.
3. In the **Add 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.)



Attribute, Entity or Relationship	Outcome
☑ the exam has a passing mark	Yes
☑ the exam's score	No
☑ the exam's name	No

☐ Select / Deselect All

4. Click **OK**. The columns are added to the active worksheet.



the exam #	Global #	(the exam has a passing mark)	the exam's score	the exam's name

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).

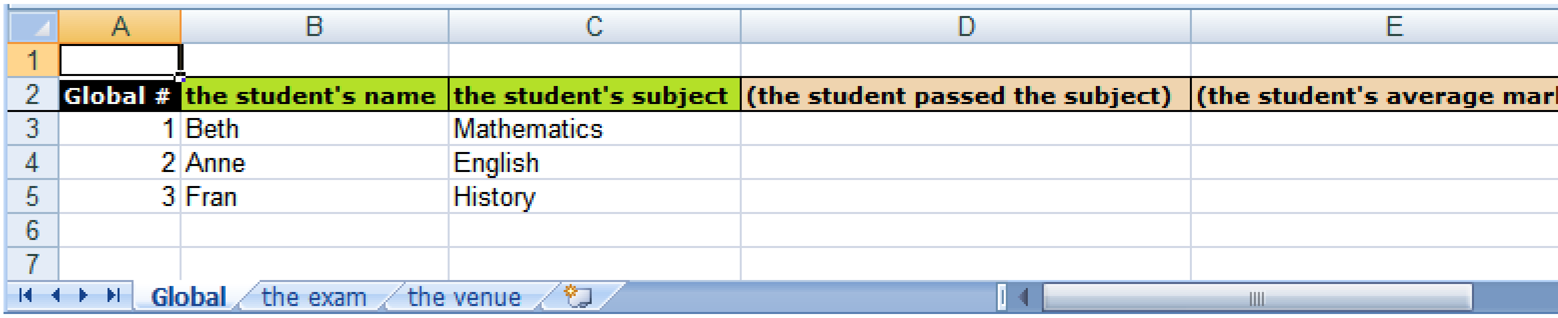
ii. Rows - you can have blank rows.

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).

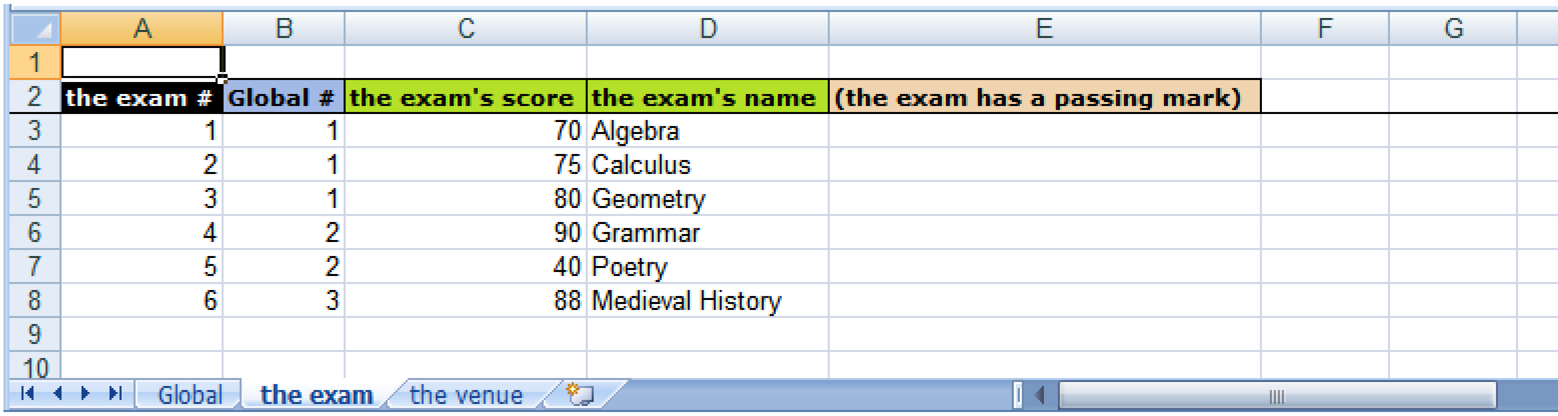


Global #	the student's name	the student's subject	(the student passed the subject)	(the student's average mar
1	Beth	Mathematics		
2	Anne	English		
3	Fran	History		

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	the exam #	Global #	the exam's score	the exam's name	(the exam has a passing mark)		
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							

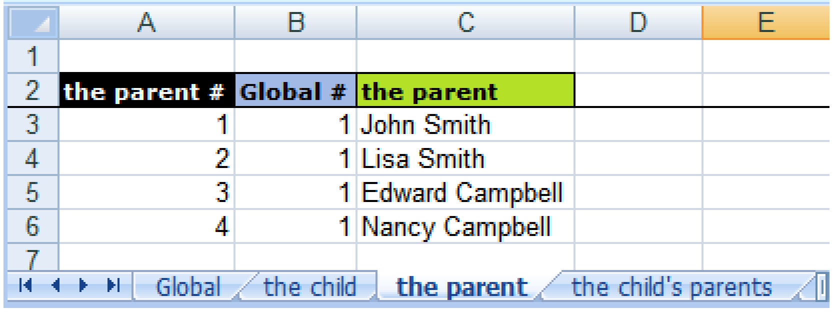
In the examples above, Beth (Global ID #1) has taken three exams (Algebra, Calculus and Geometry), Anne (Global ID #2) has taken two exams (Grammar and Poetry) and Fran (Global ID #3) has taken one exam (Medieval History).

Enter data for many-to-many relationships

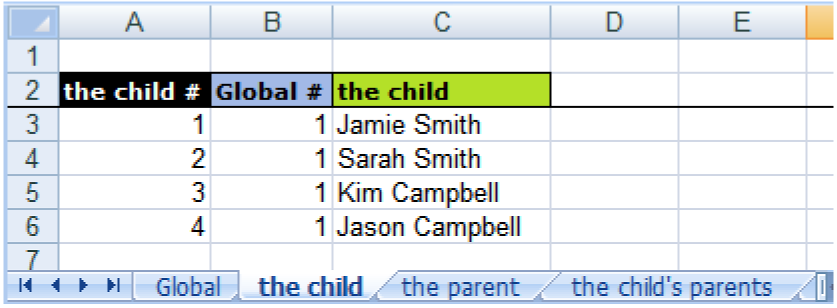
On a many-to-many relationship worksheet you need to:

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:

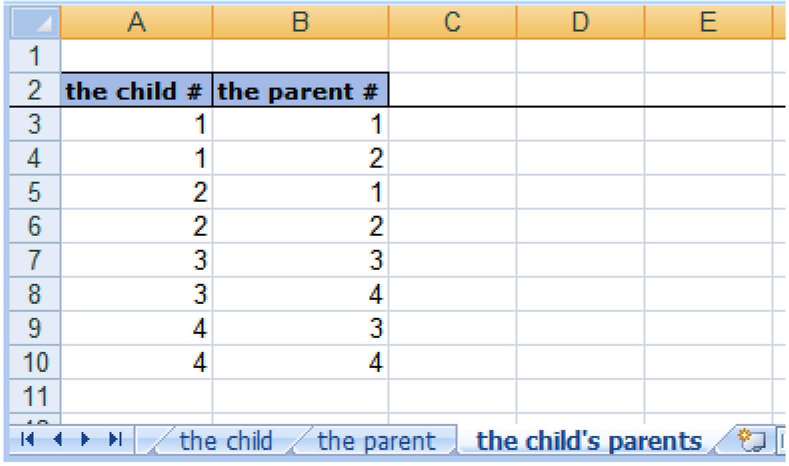


the parent #	Global #	the parent
1	1	John Smith
2	1	Lisa Smith
3	1	Edward Campbell
4	1	Nancy Campbell



the child #	Global #	the child
1	1	Jamie Smith
2	1	Sarah Smith
3	1	Kim Campbell
4	1	Jason Campbell

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:



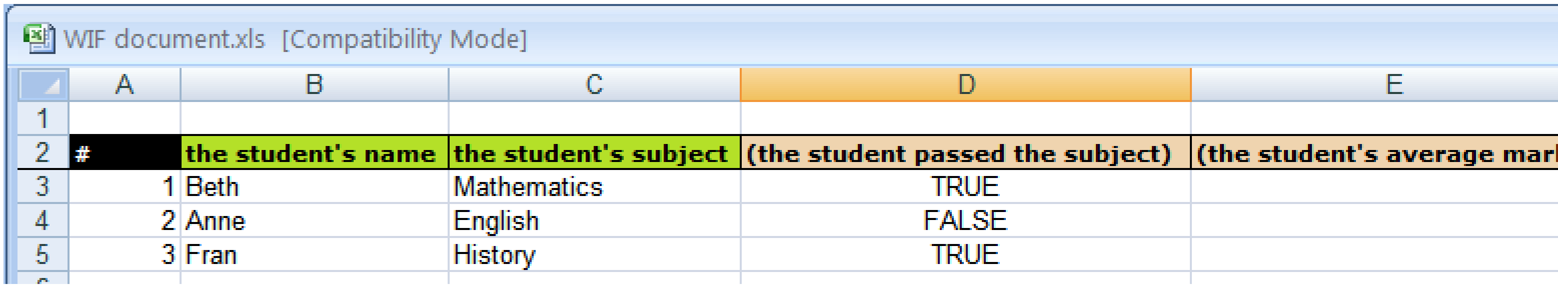
the child #	the parent #
1	1
1	2
2	1
2	2
3	3
3	4
4	3
4	4

This tells us that:

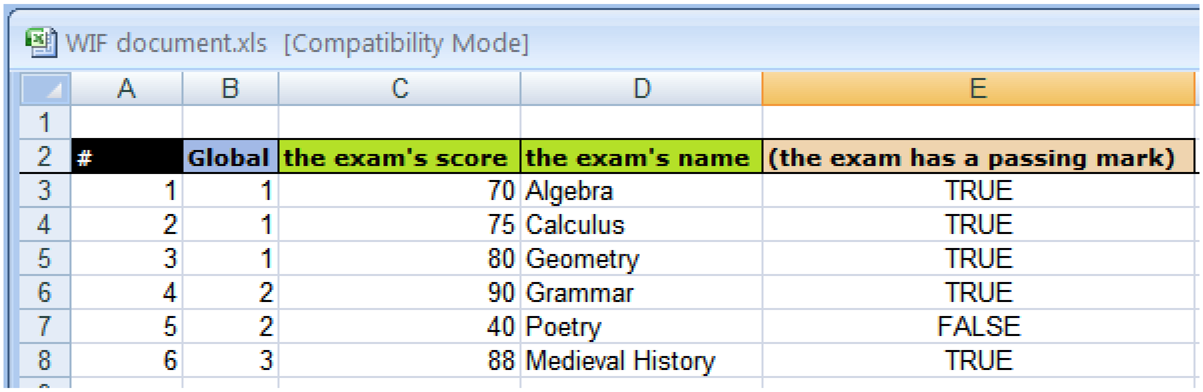
- Jamie Smith’s parents are John Smith and Lisa Smith.
- Sarah Smith’s parents are John Smith and Lisa Smith.
- Kim Campbell’s parents are Edward Campbell and Nancy Campbell.
- Jason Campbell’s parents are Edward Campbell and Nancy Campbell.

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	#	the student's name	the student's subject	(the student passed the subject)	(the student's average mar
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.
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	The unique identifier is to include: • The creating developer's initials • An abbreviation to identify the section of the rulebase being tested • A sequential number. For example, the tenth unit test case created by John Smith for Retirement Pensions Category C would be called unit_JSRPC10.xml. This format allows developers to readily identify their own test cases.
Formal Test Script	The formal test script is to be maintained by the testing team. Use the unique identifier assigned to the test case in the formal test script. 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. This will help to maintain a database of test case IDs and descriptions. 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.
Client	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.
Business Development/Partners	As for Client.

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:

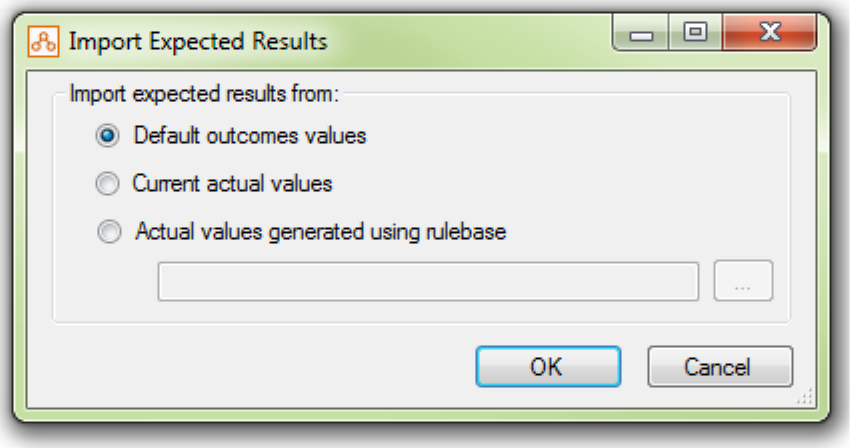
Option	Behavior
Set Expected Value...	Opens the **Edit Expected Result** dialog box where you can specify a particular value for the expected result, an expected result of uncertain, or an expected result of

Option	Behavior
	unknown. You can also specify change points for the expected result.
Set Expected Value to Default (<default expected result value>)	Defaults the expected result to the value specified as the default value in the **Edit Outcome** dialog box.
Set Expected Value to Current Value	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.
Set Expected Value to true	Sets the expected value to 'true'. (This option is only available for boolean attributes.)
Set Expected Value to false	Sets the expected value to 'false'. (This option is only available for boolean attributes.)
Set Expected Value to Unknown	Sets the expected value to 'unknown'.
Set Expected Value to Uncertain	Sets the expected value to 'uncertain'.

3. 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:

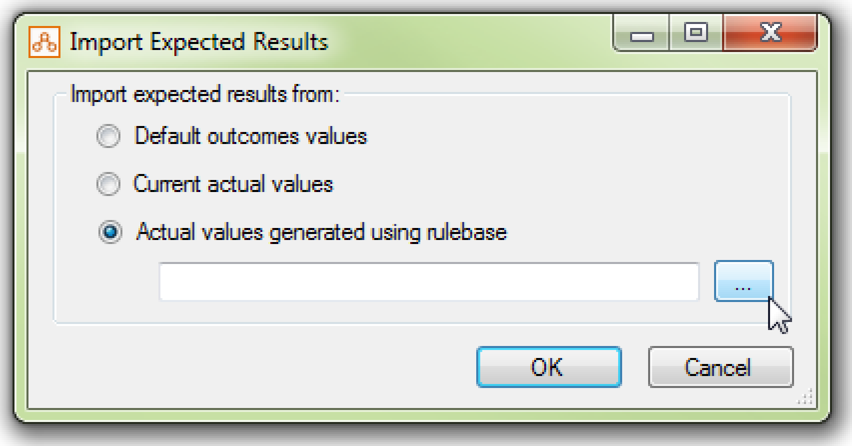
1. Right-click the test case on the Test Cases tab in your test script file and select **Import Expected Results...**
2. In the **Import Expected Results** dialog, select where you want to import the expected results from. The options are:
   - Default outcomes values
   - Current actual values
   - Actual values generated using rulebase



3. (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.

   Import Expected Results

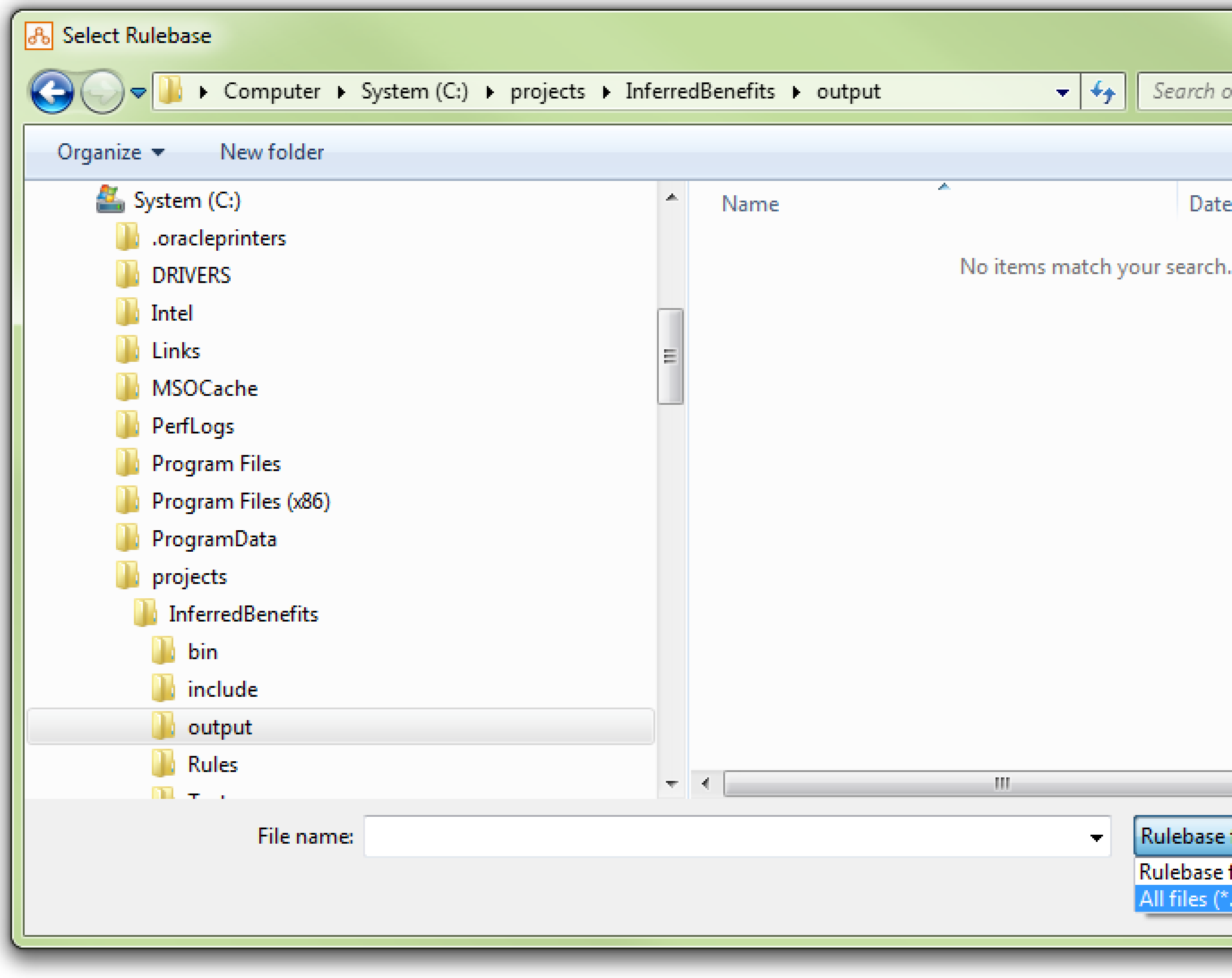
   Import expected results from:

   ○ Default outcomes values

   ○ Current actual values

   ● Actual values generated using rulebase

   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 (*.*)**.



No items match your search.

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

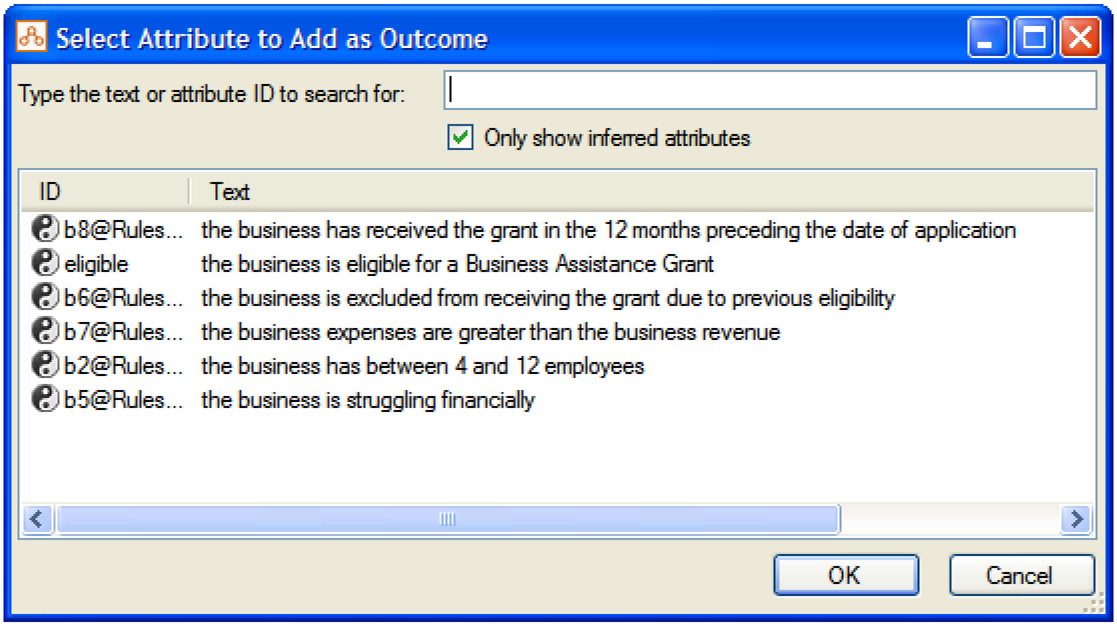
Attributes from any entity can be added as outcomes.

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:

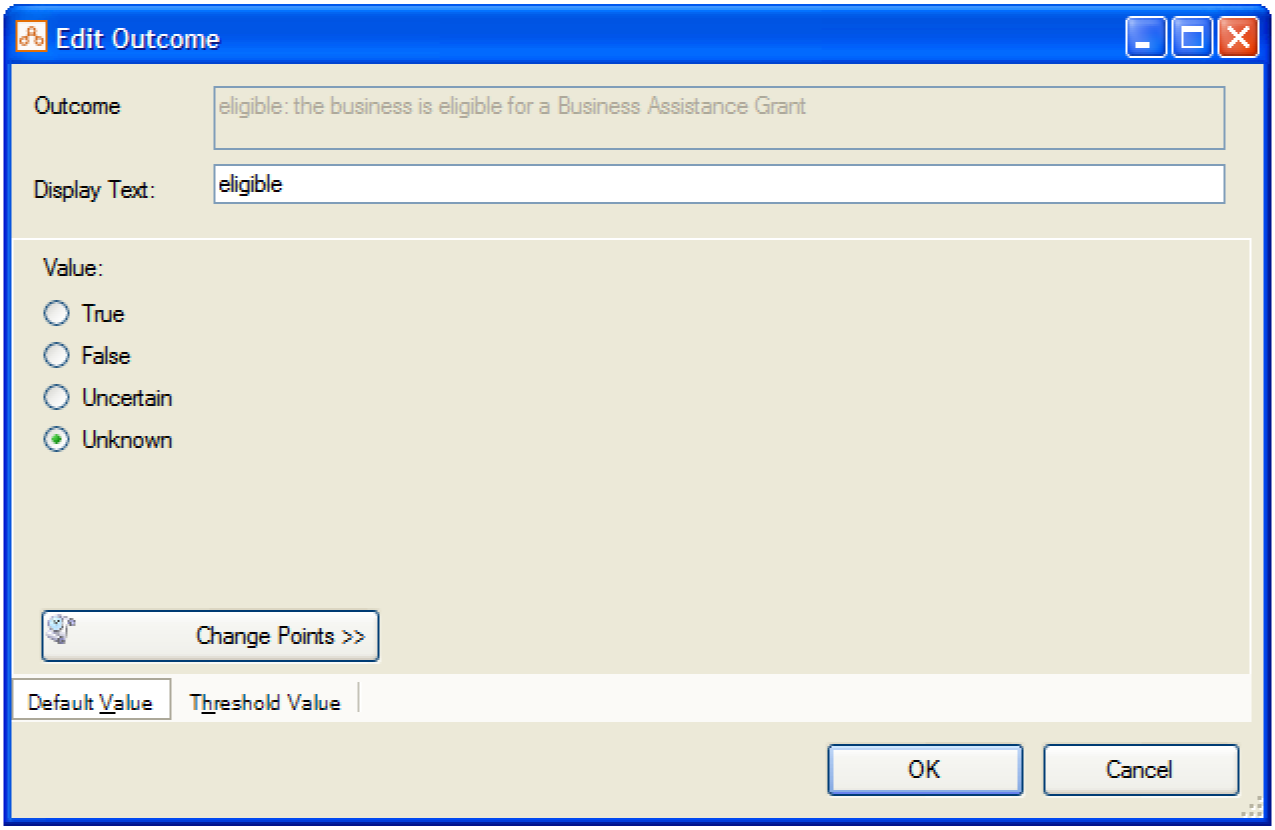
1. Right-click anywhere in the outcome set editor and select **Add New Outcome...**.
   The **Select Attribute to Add as Outcome** dialog will be displayed.



ID	Text
b8@Rules...	the business has received the grant in the 12 months preceding the date of application
eligible	the business is eligible for a Business Assistance Grant
b6@Rules...	the business is excluded from receiving the grant due to previous eligibility
b7@Rules...	the business expenses are greater than the business revenue
b2@Rules...	the business has between 4 and 12 employees
b5@Rules...	the business is struggling financially

   (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.



Outcome: eligible: the business is eligible for a Business Assistance Grant

Display Text: eligible

Value:

○ True

○ False

○ Uncertain

● Unknown

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

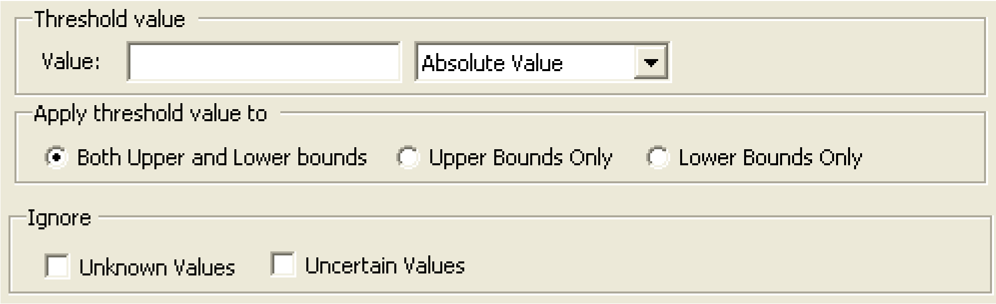
To add an attribute as an outcome from the test case editor:

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: • Both upper and lower bounds – the threshold will be applied as Y – T ≤ X ≤ Y + T (default) • Upper bounds only – the threshold will be applied as Y – T ≤ X < Y • Lower bounds only – the threshold will be applied as Y ≤ X ≤ Y + T where X = Actual Result, Y = Expected Result and T= threshold value.
Ignore		Specifies whether unknown and or/uncertain values should be ignored, as follows: • Unknown values – this means that a test will pass if Expected Value = Actual Value (to within whatever threshold is specified) OR Actual Value = unknown. • Uncertain values – this means that a test will pass if Expected Value = Actual Value (to within whatever threshold is specified) OR Actual Value = uncertain.

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...**

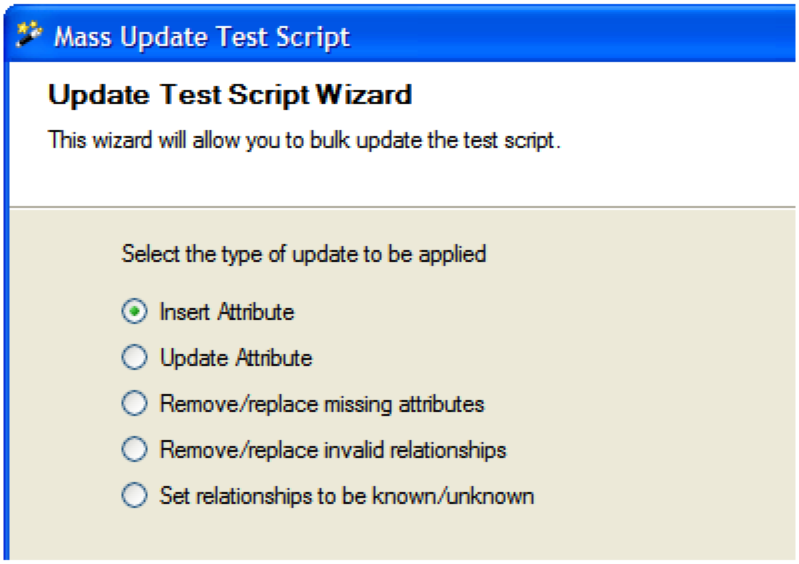
NOTE: If an entity no longer exists in the rulebase then all attributes belonging to that entity will be flagged as invalid.

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.

   Update Test Script Wizard

   This wizard will allow you to bulk update the test script.

   Select the type of update to be applied

   ● Insert Attribute
   ○ Update Attribute
   ○ Remove/replace missing attributes
   ○ Remove/replace invalid relationships
   ○ Set relationships to be known/unknown

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**.

Entity/Relationship

Entity: the child

Relationship: the child's parents

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?

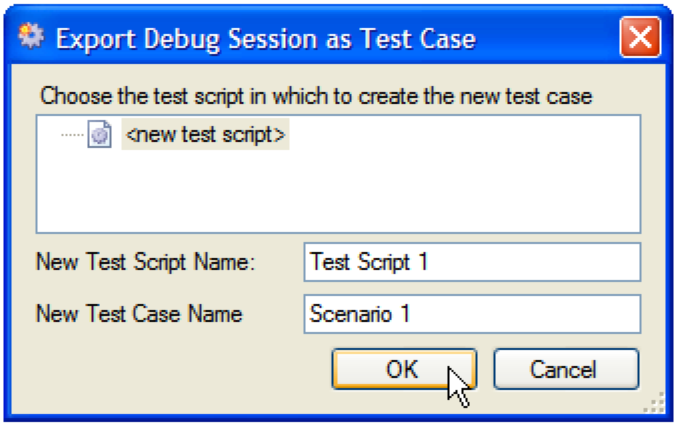
Export interview data directly into a new test case from the debugger

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:

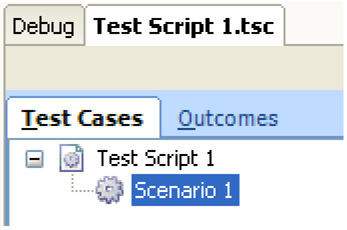
1. Open the debugger by selecting **Build | Build and Debug**.
2. Start an investigation and enter the desired values. (For more information on using the debugger, see Debug a rulebase.)
3. Click on the arrow on the right side of the **Export** button (located in the top right of the **Debug** view), 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**.



New Test Script Name: Test Script 1

New Test Case Name Scenario 1

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.

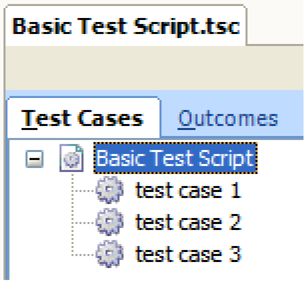
Rulebase: Business Assistance Grant A Locale: en-US User ID: guest Case ID: test case 1

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	Numeric values must adhere to the following conditions: • The '.' character is the decimal separator • Thousands separators are not supported • Currency symbols are not supported • For scientific notation, the '+' is not supported in the exponent	Blank values are considered UNCERTAIN
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: • String values are case insensitive, so "YES" is the same as "yes" and "Yes" • Leading zeros are not truncated from numeric values • "True", "Yes" and 1 will be parsed as TRUE • "False", "No" and 0 will be parsed as FALSE	Blank values are considered UNCERTAIN
Date	Date values must adhere to the format "yyyy-MM-dd" where: • yyyy is the four-digit year • MM is the two-digit month, including leading zero for values below 10 • dd is the two-digit day, including leading zero for values below 10	Blank values are considered UNCERTAIN
Datetime	Datetime values must adhere to the format "yyyy-MM-dd HH:mm:ss" where: • yyyy is the four-digit year • MM is the two-digit month, including leading zero for values below 10 • dd is the two-digit day, including leading zero for values below 10 • HH is the two-digit 24-hour hour value, including leading zero for values below 10 • mm is the two-digit minute value, including leading zero for values below 10 • ss is the two-digit seconds value, including leading zero for values below 10	Blank values are considered UNCERTAIN
Time	Time values must adhere to the format "HH:mm:ss" where: • HH is the two-digit 24-hour hour value, including leading zero	Blank values are considered UNCERTAIN

Value Type	Format Description	Blank Value
	for values below 10 - mm is the two-digit minute value, including leading zero for values below 10 - ss is the two-digit seconds value, including leading zero for values below 10	

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 the 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.

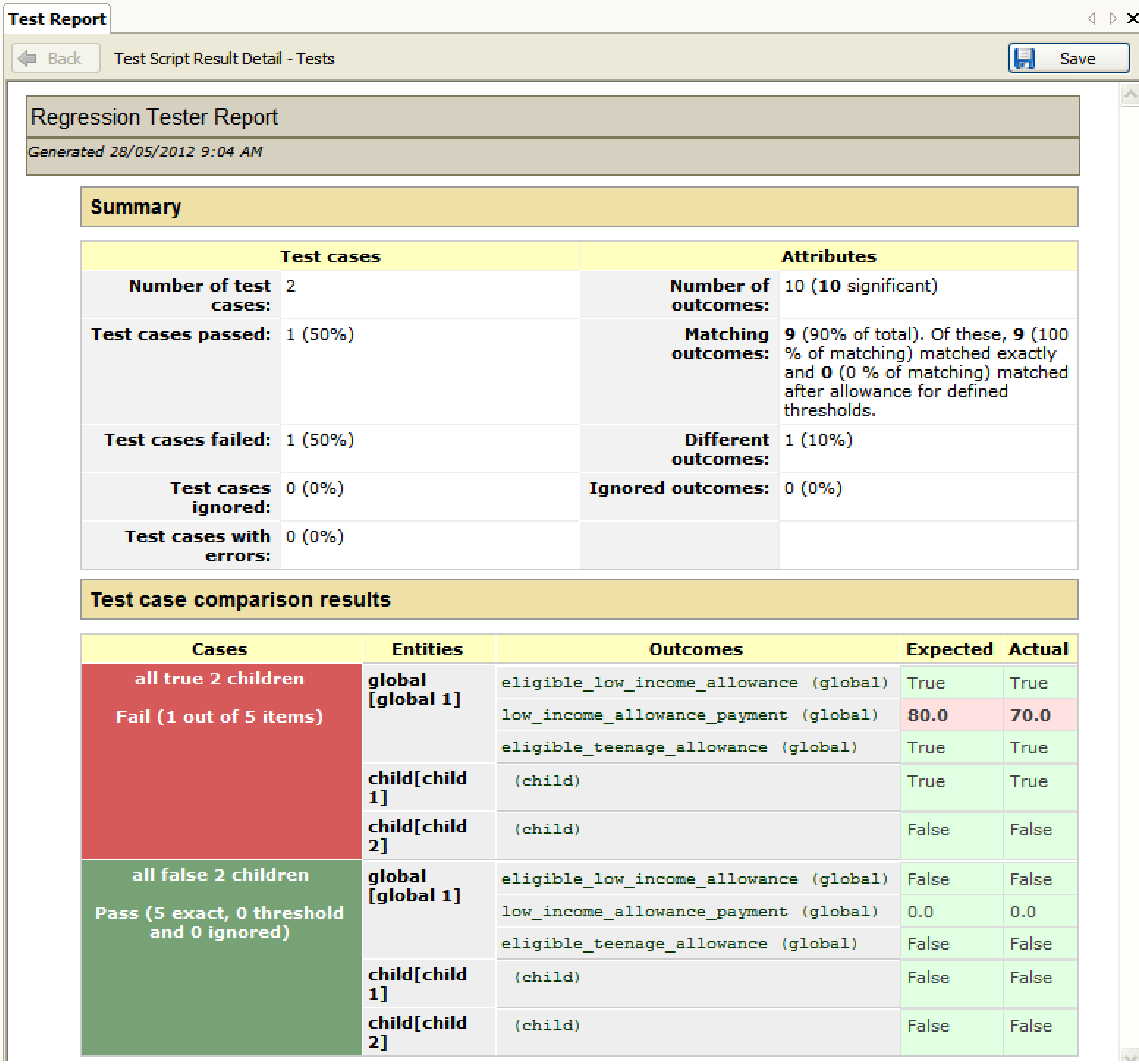
NOTE: You should re-run your test script/s whenever the rulebase changes to guarantee that the results are still correct.

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.



Regression Tester Report

Generated 28/05/2012 9:04 AM

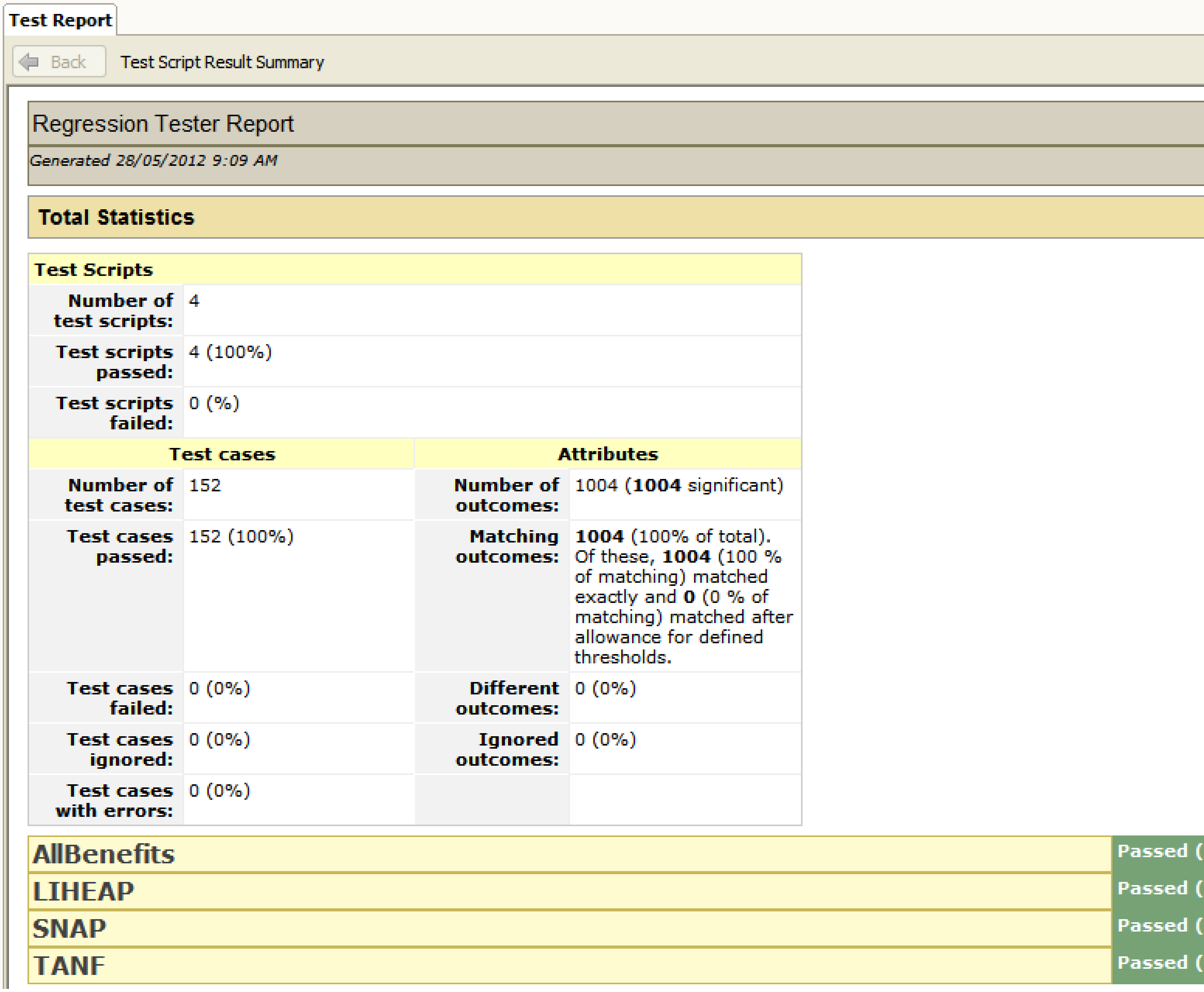
Summary

Test cases		Attributes	
Number of test cases:	2	**Number of outcomes:**	10 (**10** significant)
Test cases passed:	1 (50%)	**Matching outcomes:**	**9** (90% of total). Of these, **9** (100 % of matching) matched exactly and **0** (0 % of matching) matched after allowance for defined thresholds.
Test cases failed:	1 (50%)	**Different outcomes:**	1 (10%)
Test cases ignored:	0 (0%)	**Ignored outcomes:**	0 (0%)
Test cases with errors:	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.



Regression Tester Report

Generated 28/05/2012 9:09 AM

Total Statistics

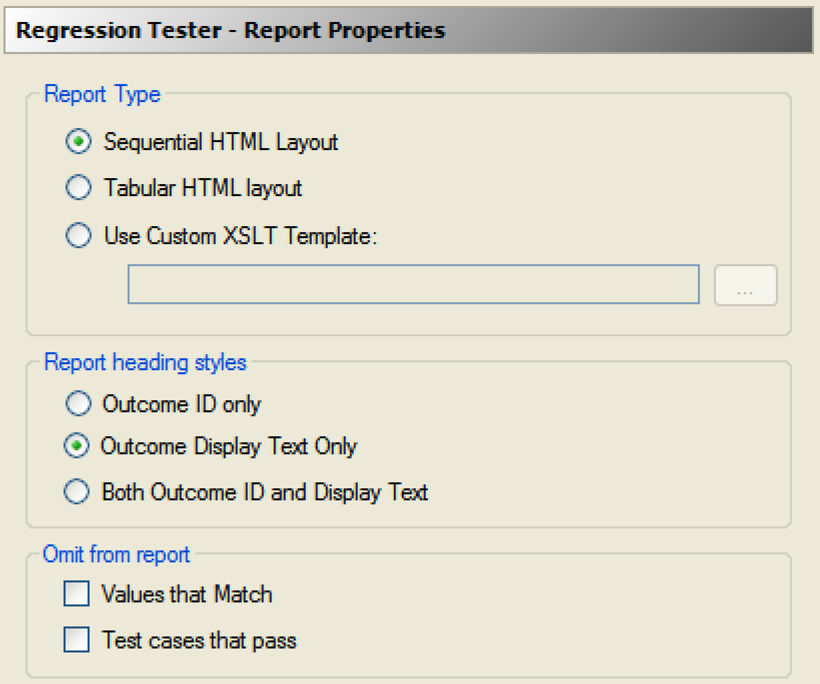
Test Scripts			
Number of test scripts:	4		
Test scripts passed:	4 (100%)		
Test scripts failed:	0 (%)		
Test cases		**Attributes**	
Number of test cases:	152	**Number of outcomes:**	1004 (**1004** significant)
Test cases passed:	152 (100%)	**Matching outcomes:**	**1004** (100% of total). Of these, **1004** (100 % of matching) matched exactly and **0** (0 % of matching) matched after allowance for defined thresholds.
Test cases failed:	0 (0%)	**Different outcomes:**	0 (0%)
Test cases ignored:	0 (0%)	**Ignored outcomes:**	0 (0%)
Test cases with errors:	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	The Test Report can be rendered in two distinct layouts – sequential or tabular. • the sequential layout lists results for cases down the page • the tabular layout presents results in a grid (cases rows and attribute columns) Alternatively, you can specify a custom XSLT template for the regression tester to use when generating the Test Report.
Report heading styles	There are three options for report headings: • Outcome ID only - this will cause reports to display with attribute IDs (either model ID or public name) as headings. • 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. • Both outcome ID and display text - will display both the attribute IDs and the Display Text.
Omit from report	You have the option to omit from the Test Report: • Values that match - this excludes attributes with outcome values that match the test case, and/or • Test cases that pass - this excludes test cases that have passed.

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

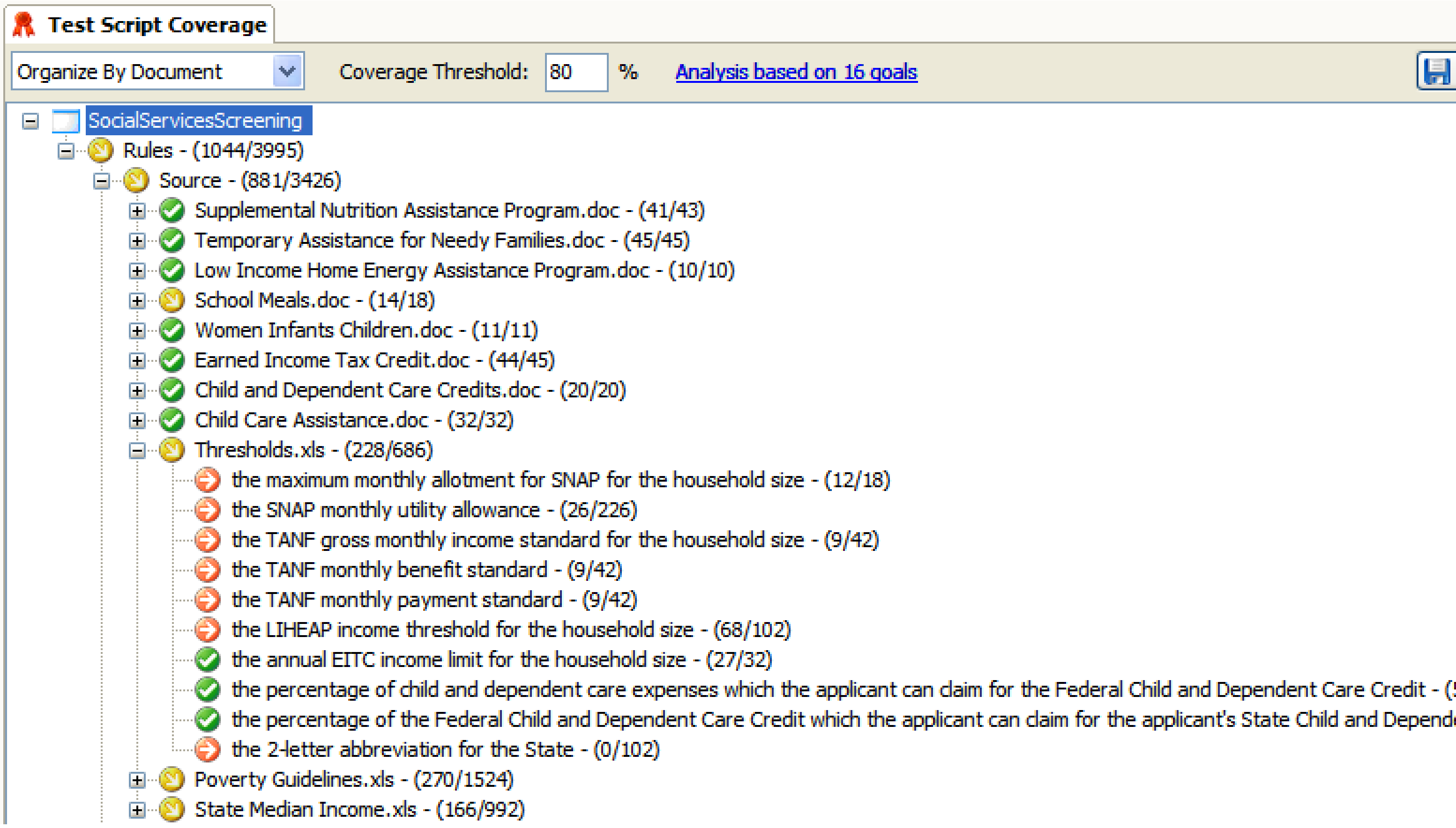
Change the platform used by the analyzer

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.



Coverage Threshold: 80 %

- 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 - (
        - 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)

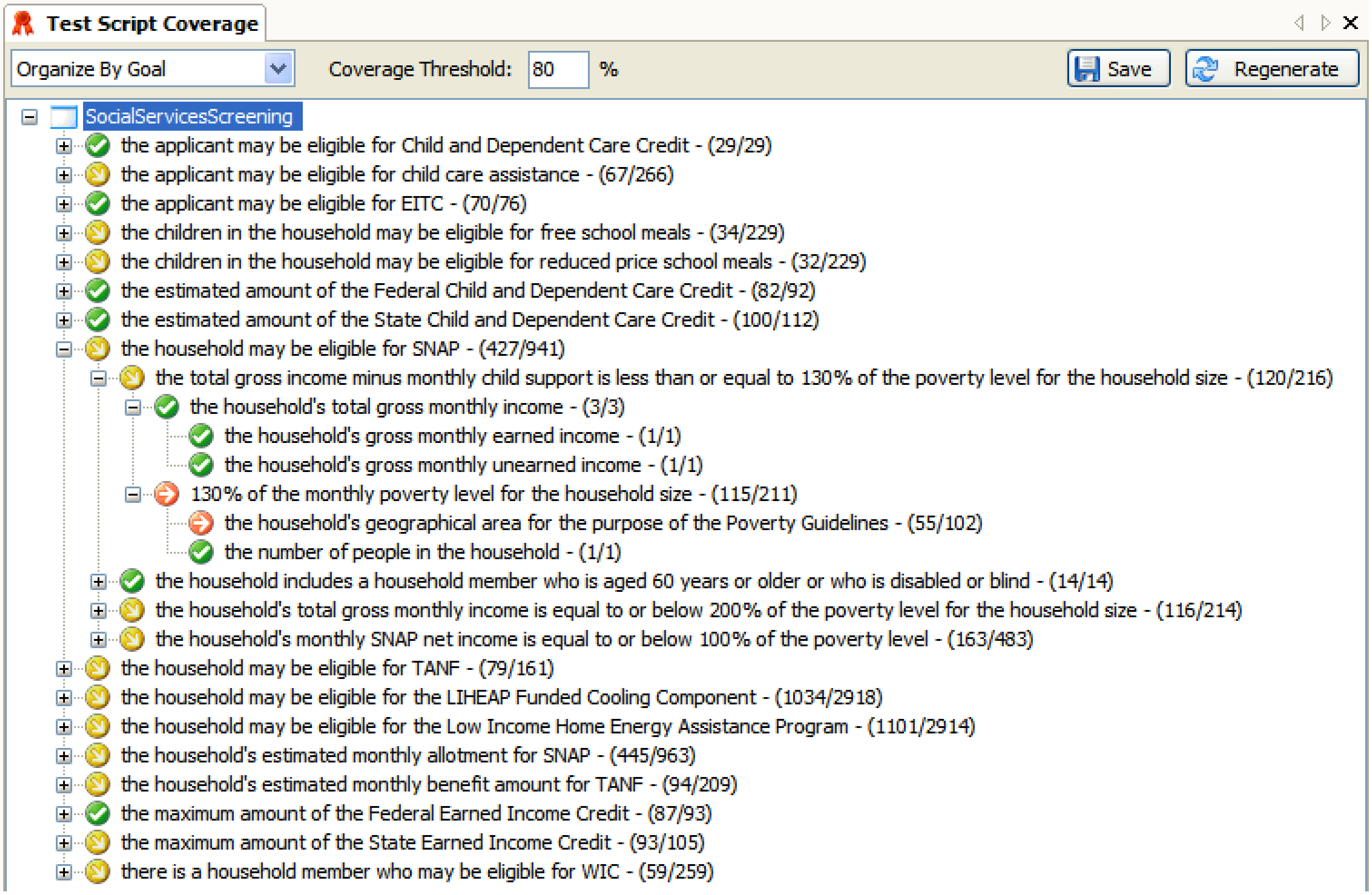
View the report organized by document

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.



Organize By Goal

Coverage Threshold: 80 %

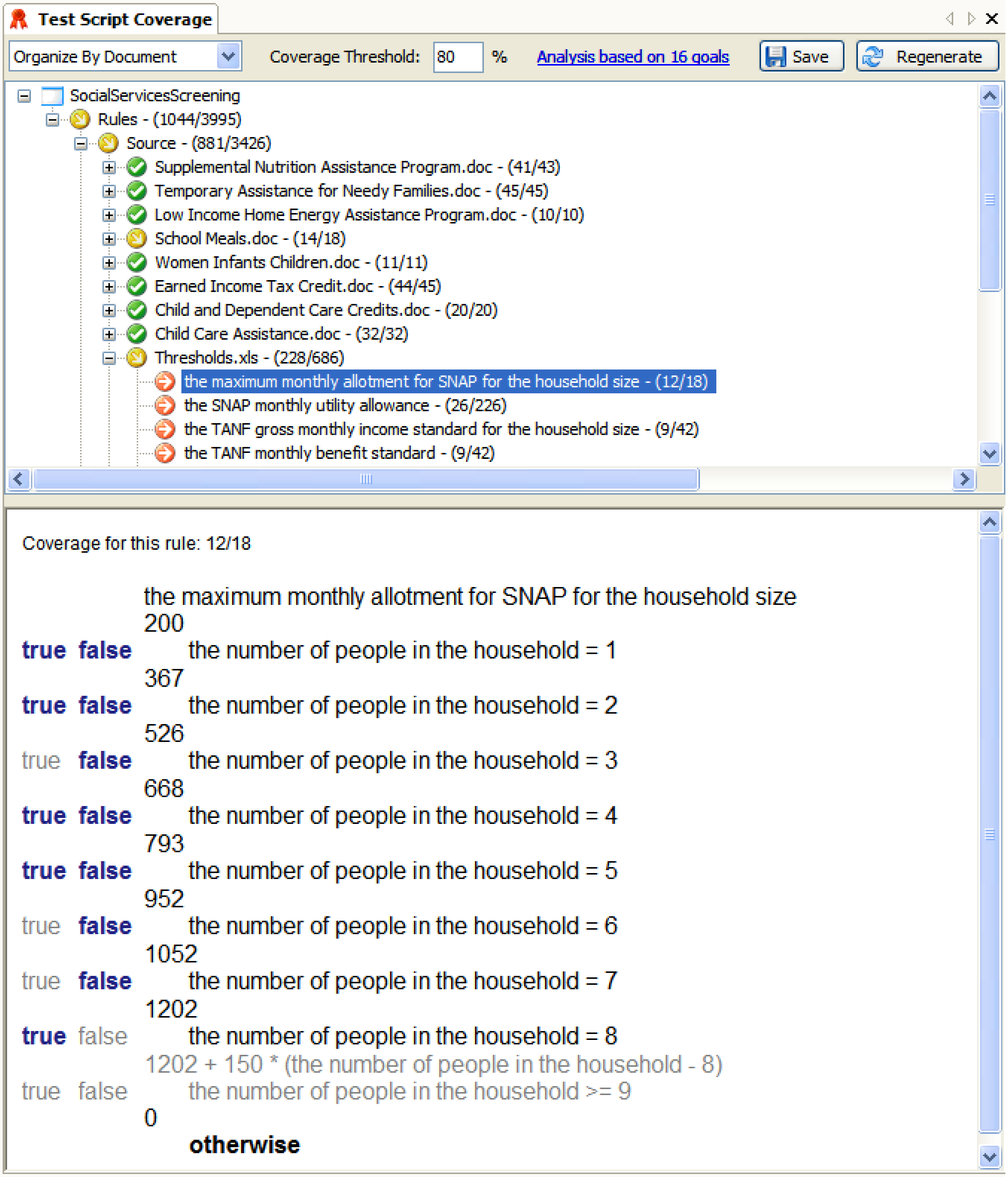
- SocialServicesScreening
  - the applicant may be eligible for Child and Dependent Care Credit - (29/29)
  - the applicant may be eligible for child care assistance - (67/266)
  - the applicant may be eligible for EITC - (70/76)
  - the children in the household may be eligible for free school meals - (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:



- 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)

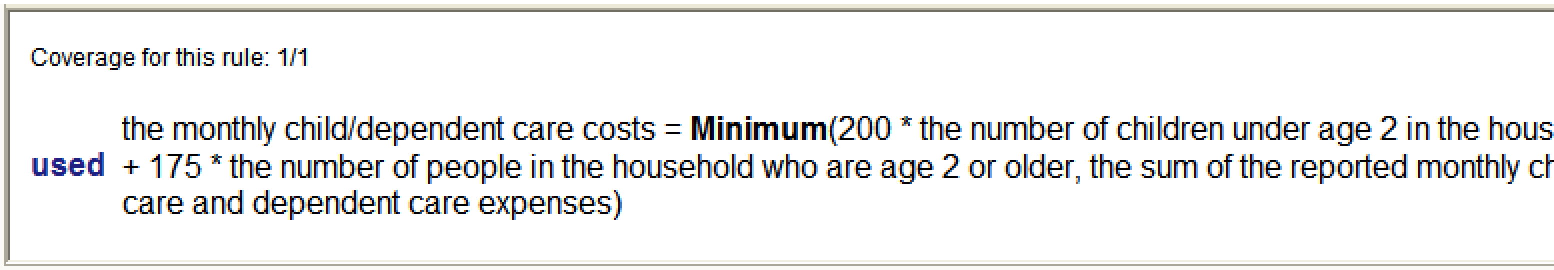
true false the number of people in the household >= 9

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 hous
+ 175 * the number of people in the household who are age 2 or older, the sum of the reported monthly ch
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

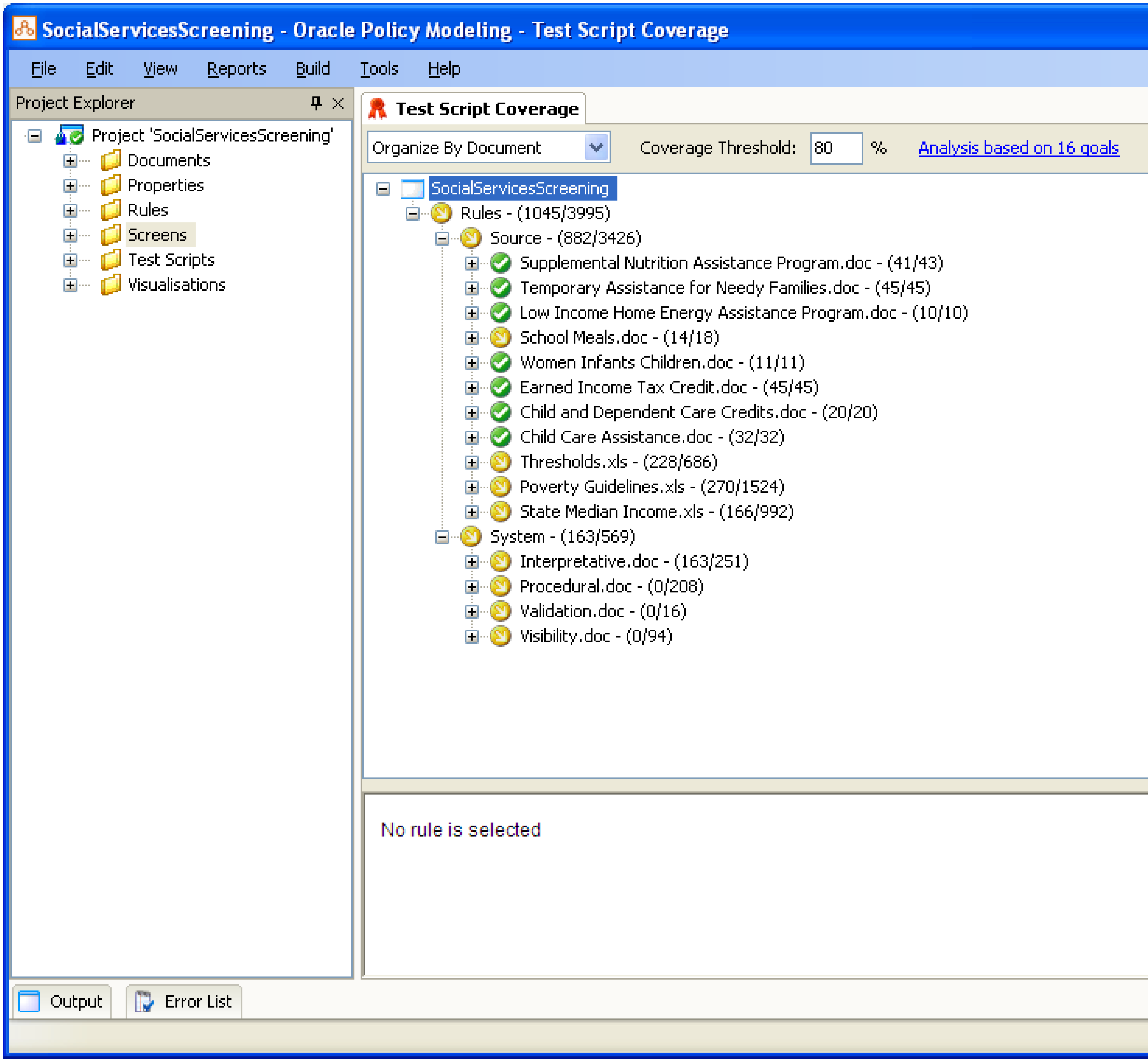
For a suite of test cases to provide any confidence in the correct workings of a rulebase, it is important that the majority of the rules in the project are meaningfully exercised by test cases.

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**.



- SocialServicesScreening
  - Rules - (1045/3995)
    - Source - (882/3426)
      - ✓ 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 - (45/45)
      - ✓ Child and Dependent Care Credits.doc - (20/20)
      - ✓ Child Care Assistance.doc - (32/32)
      - Thresholds.xls - (228/686)
      - Poverty Guidelines.xls - (270/1524)
      - State Median Income.xls - (166/992)
    - System - (163/569)
      - Interpretative.doc - (163/251)
      - Procedural.doc - (0/208)
      - Validation.doc - (0/16)
      - Visibility.doc - (0/94)

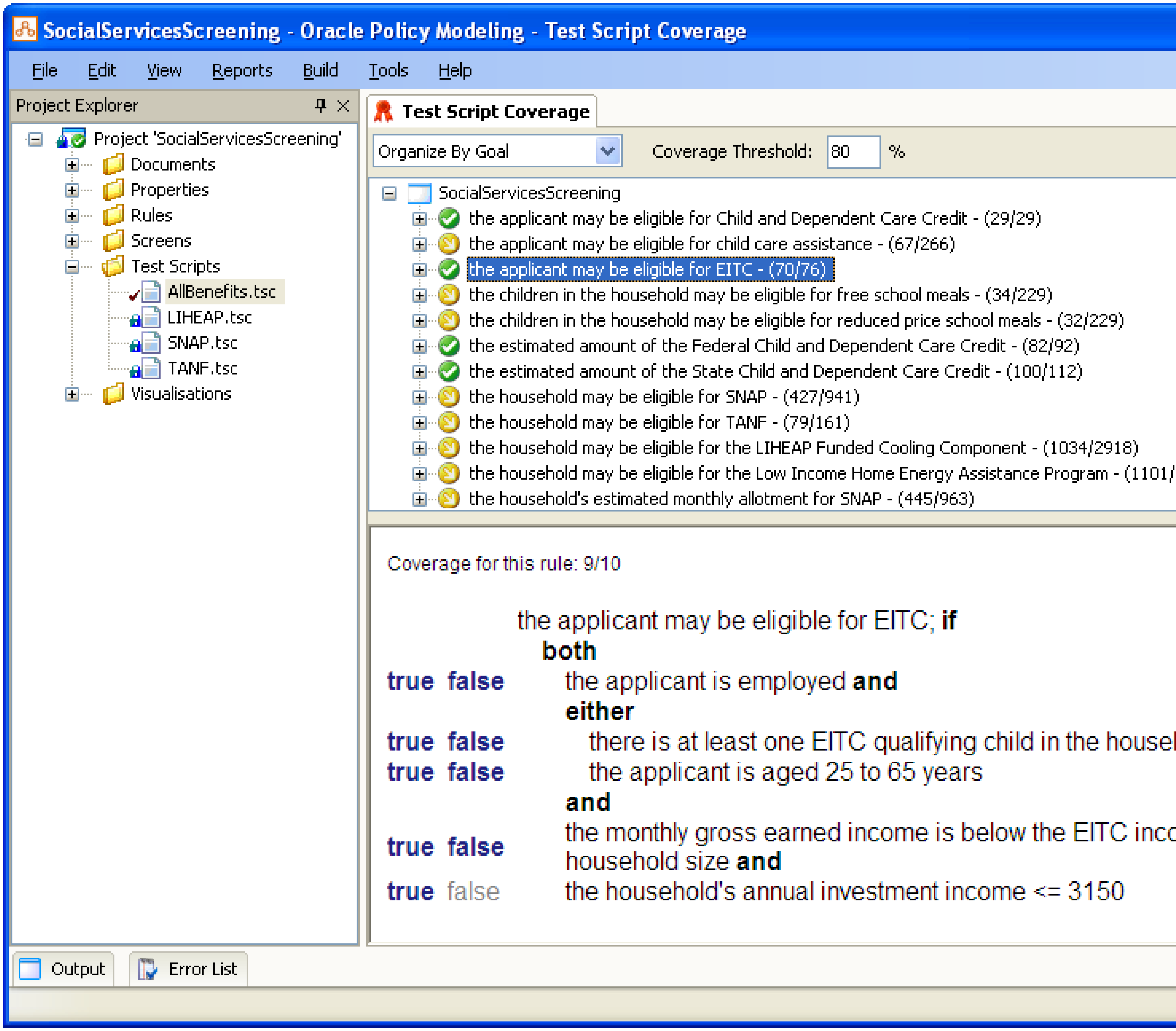
No rule is selected

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.



Project 'SocialServicesScreening'
- Documents
- Properties
- Rules
- Screens
- Test Scripts
  - AllBenefits.tsc
  - LIHEAP.tsc
  - SNAP.tsc
  - TANF.tsc
- Visualisations

Organize By Goal

Coverage Threshold: 80 %

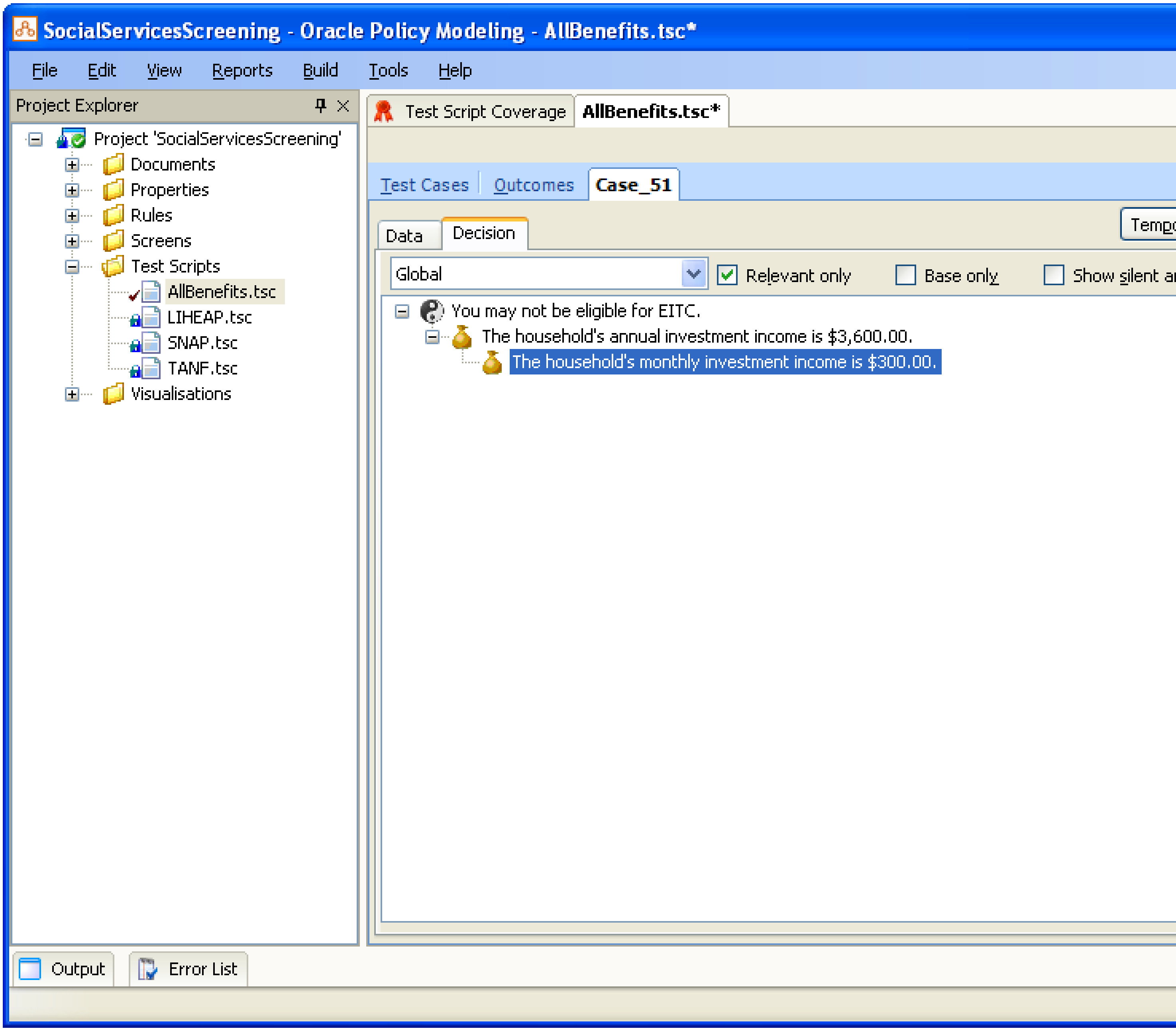
SocialServicesScreening
- ✓ the applicant may be eligible for Child and Dependent Care Credit - (29/29)
- the applicant may be eligible for child care assistance - (67/266)
- ✓ the applicant may be eligible for EITC - (70/76)
- the children in the household may be eligible for free school meals - (34/229)
- 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 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/
- the household's estimated monthly allotment for SNAP - (445/963)

Coverage for this rule: 9/10

the applicant may be eligible for EITC; **if**
both
true **false** the applicant is employed **and**
either
true **false** there is at least one EITC qualifying child in the housel
true **false** the applicant is aged 25 to 65 years
and
true **false** the monthly gross earned income is below the EITC inco household size **and**
true false the household's annual investment income <= 3150

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'.

5. 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 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.



Global ☑ Relevant only ☐ Base only ☐ Show silent a

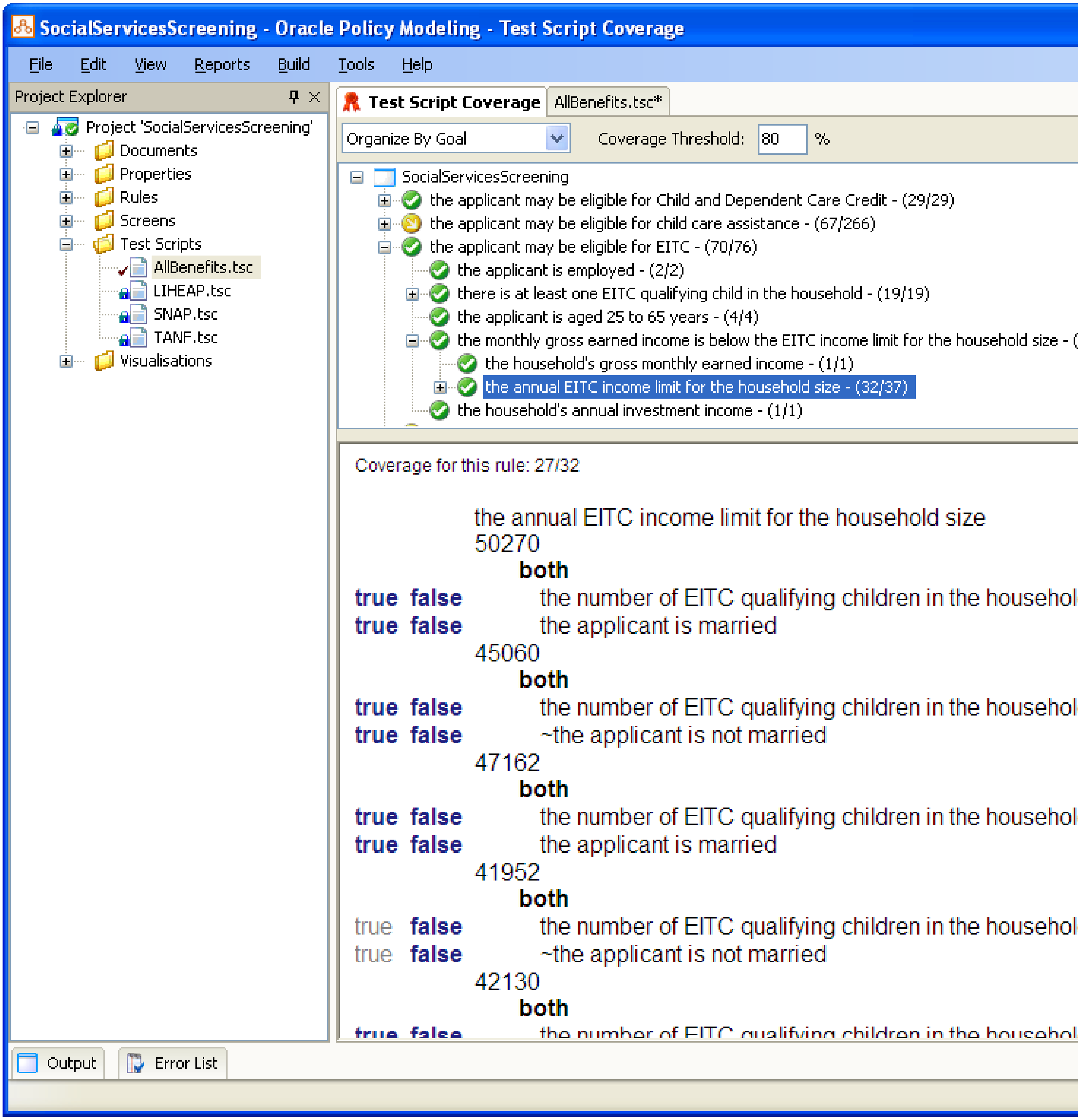
You may not be eligible for EITC.
The household's annual investment income is $3,600.00.
The household's monthly investment income is $300.00.

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".



Project 'SocialServicesScreening'
- Documents
- Properties
- Rules
- Screens
- Test Scripts
  - AllBenefits.tsc
  - LIHEAP.tsc
  - SNAP.tsc
  - TANF.tsc
- Visualisations

Organize By Goal

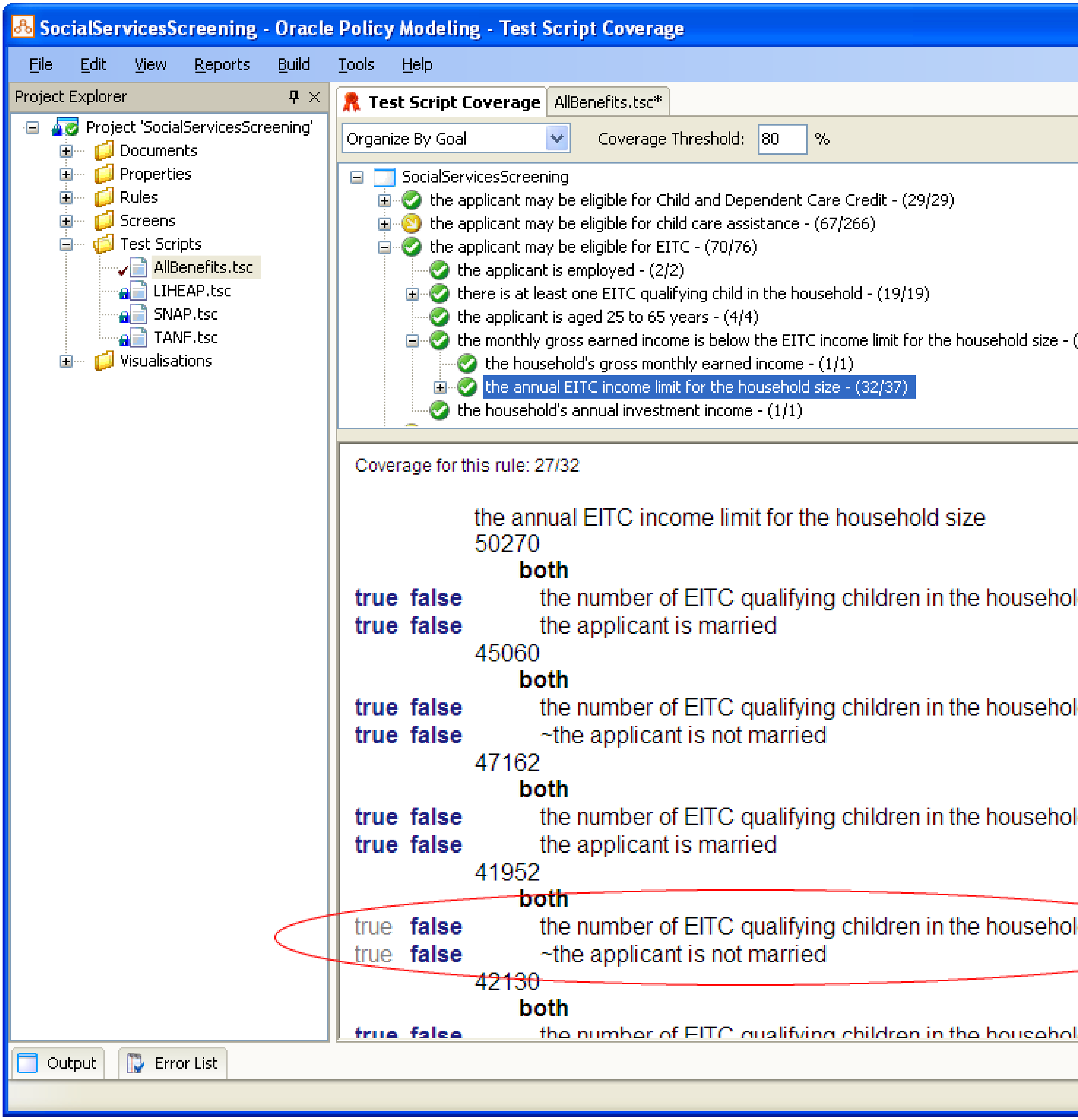
Coverage Threshold: 80 %

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 - (
    - 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 househol
true false the applicant is married
45060
both
true false the number of EITC qualifying children in the househol
true false ~the applicant is not married
47162
both
true false the number of EITC qualifying children in the househol
true false the applicant is married
41952
both
true false the number of EITC qualifying children in the househol
true false ~the applicant is not married
42130
both
true false the number of EITC qualifying children in the househol

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.



Organize By Goal

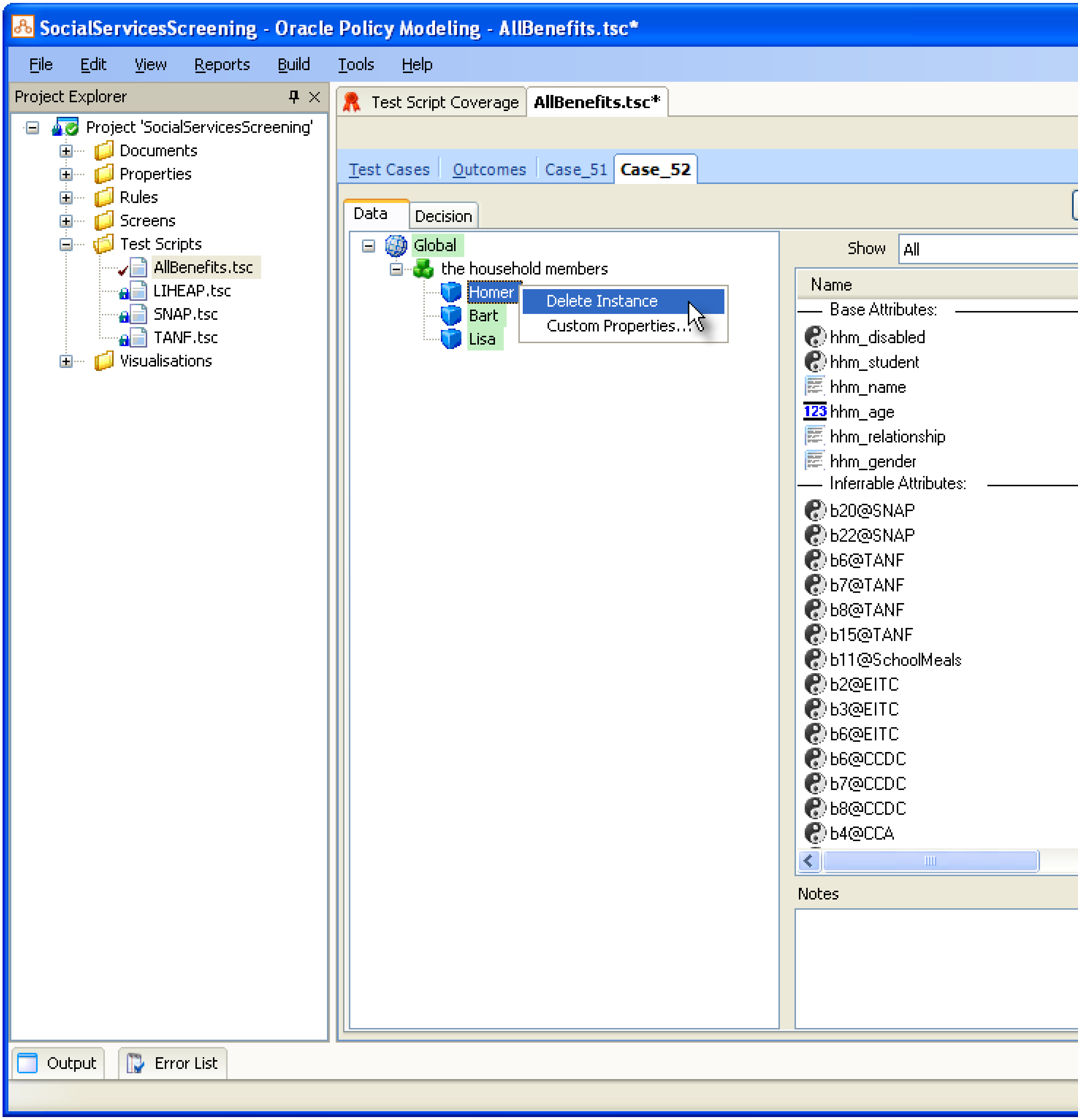
Coverage Threshold: 80 %

```
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 - (
      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)
```

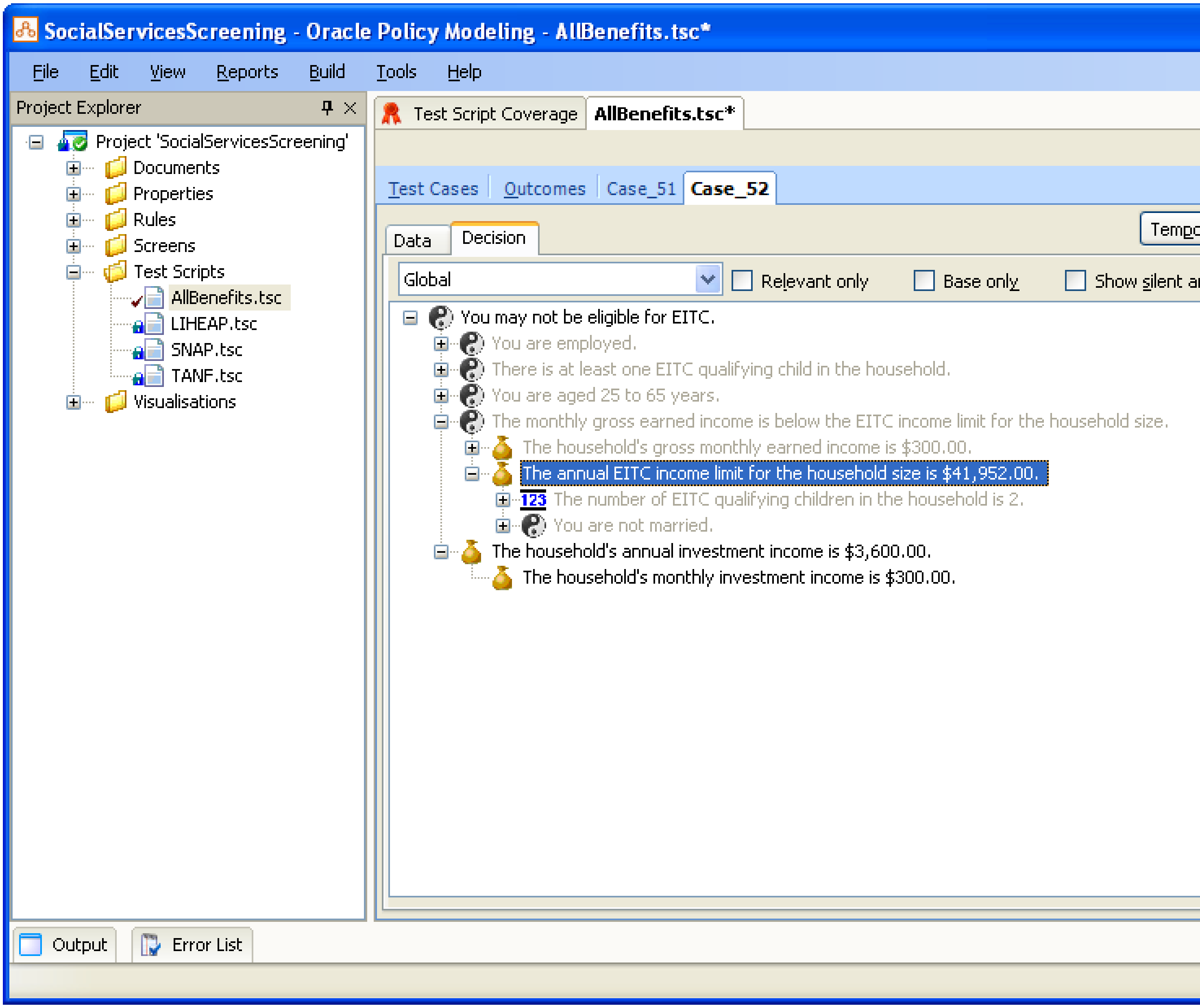
Coverage for this rule: 27/32

```
                the annual EITC income limit for the household size
                50270
                    both
true  false           the number of EITC qualifying children in the househol
true  false           the applicant is married
                45060
                    both
true  false           the number of EITC qualifying children in the househol
true  false           ~the applicant is not married
                47162
                    both
true  false           the number of EITC qualifying children in the househol
true  false           the applicant is married
                41952
                    both
true  false           the number of EITC qualifying children in the househol
true  false           ~the applicant is not married
                42130
                    both
true  false           the number of EITC qualifying children in the househol
```

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.


SocialServicesScreening - Oracle Policy Modeling - AllBenefits.tsc*
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*
Test Cases
Outcomes
Case_51
Case_52
Data
Decision
Global
the household members
Homer
Bart
Lisa
Delete Instance
Custom Properties...
Show
All
Name
Base Attributes:
hhm_disabled
hhm_student
hhm_name
hhm_age
hhm_relationship
hhm_gender
Inferrable 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".



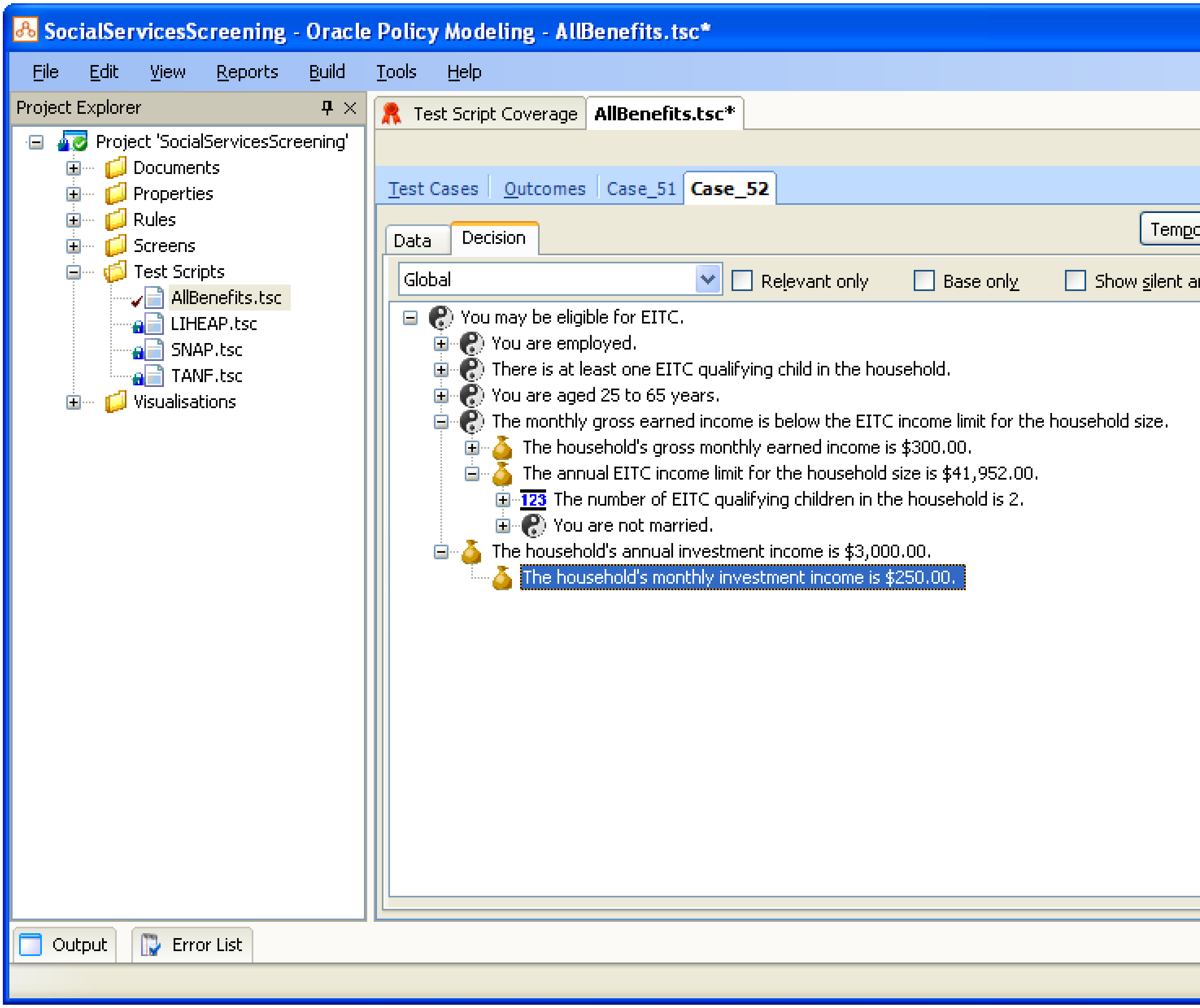
Global

☐ Relevant only ☐ Base only ☐ Show silent a

- You may not be eligible for EITC.
  - You are employed.
  - There is at least one EITC qualifying child in the household.
  - You are aged 25 to 65 years.
  - The monthly gross earned income is below the EITC income limit for the household size.
    - The household's gross monthly earned income is $300.00.
    - The annual EITC income limit for the household size is $41,952.00.
      - The number of EITC qualifying children in the household is 2.
      - You are not married.
  - The household's annual investment income is $3,600.00.
    - The household's monthly investment income is $300.00.

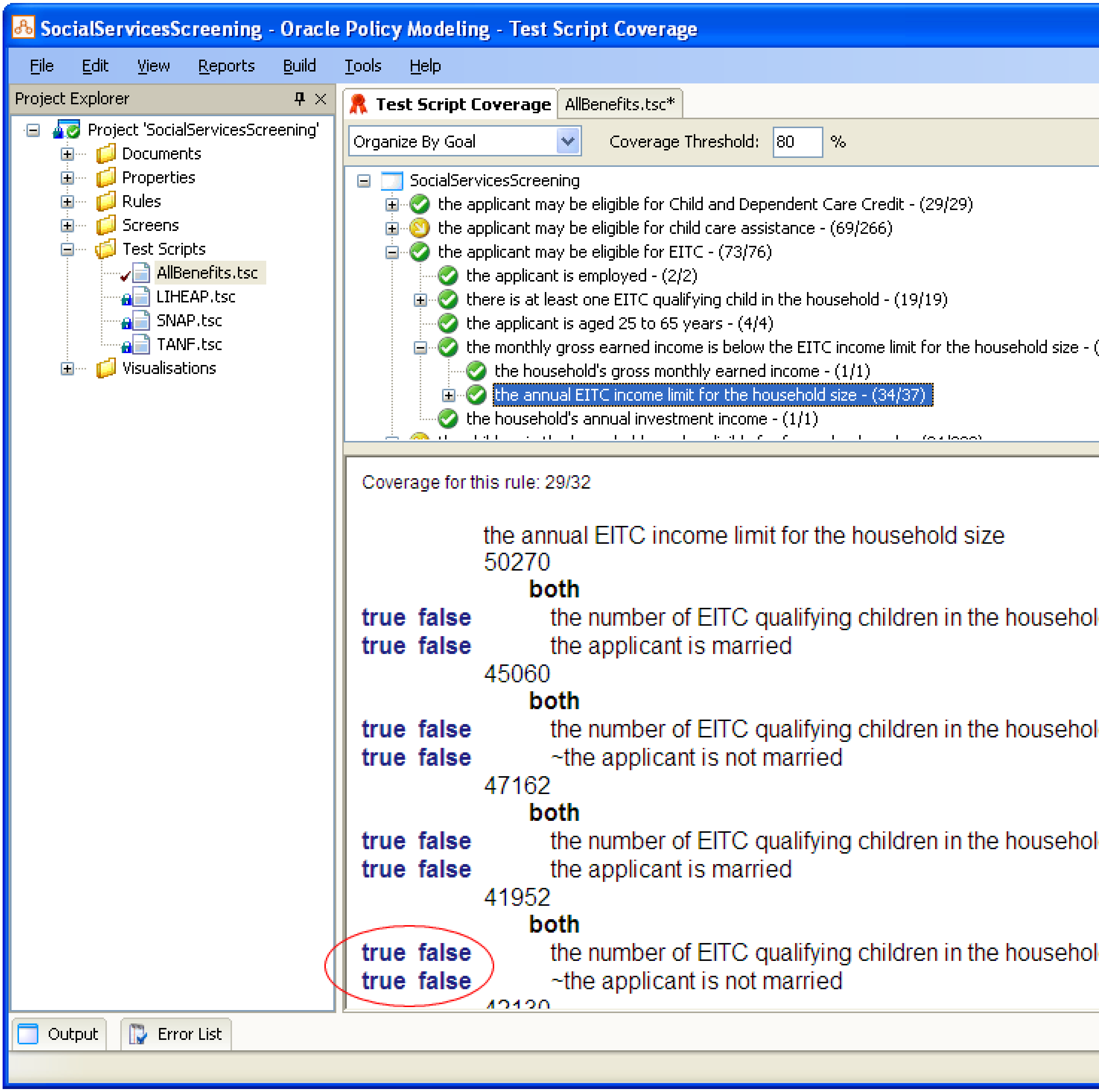
The gray values are those that not relevant to the decision. This explains why the change has not contributed to the coverage score: the change described in the previous example (which made the household not eligible for EITC) has made the income limit irrelevant. Even though the income limit was calculated 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.



- You may be eligible for EITC.
  - You are employed.
  - There is at least one EITC qualifying child in the household.
  - You are aged 25 to 65 years.
  - The monthly gross earned income is below the EITC income limit for the household size.
    - The household's gross monthly earned income is $300.00.
    - The annual EITC income limit for the household size is $41,952.00.
      - The number of EITC qualifying children in the household is 2.
      - You are not married.
  - The household's annual investment income is $3,000.00.
    - The household's monthly investment income is $250.00.

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.



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 - (
    - 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 househol
true  false                  the applicant is married
                    45060
                          both
true  false                  the number of EITC qualifying children in the househol
true  false                  ~the applicant is not married
                    47162
                          both
true  false                  the number of EITC qualifying children in the househol
true  false                  the applicant is married
                    41952
                          both
true  false                  the number of EITC qualifying children in the househol
true  false                  ~the applicant is not married
```

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

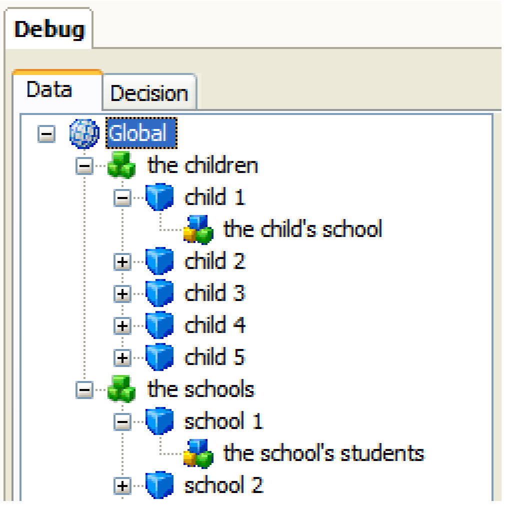
Use the integrated debugger to test the rules

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.



- Global
  - the children
    - child 1
      - the child's school
    - child 2
    - child 3
    - child 4
    - child 5
  - the schools
    - school 1
      - the school's students
    - school 2

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).

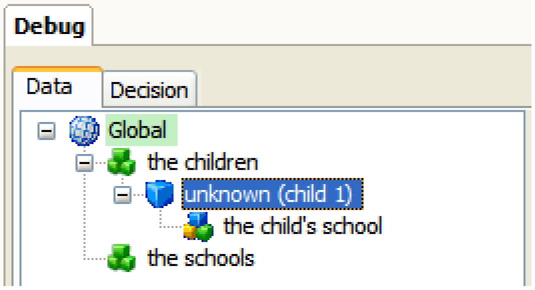
Set up entities and containment relationships

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.

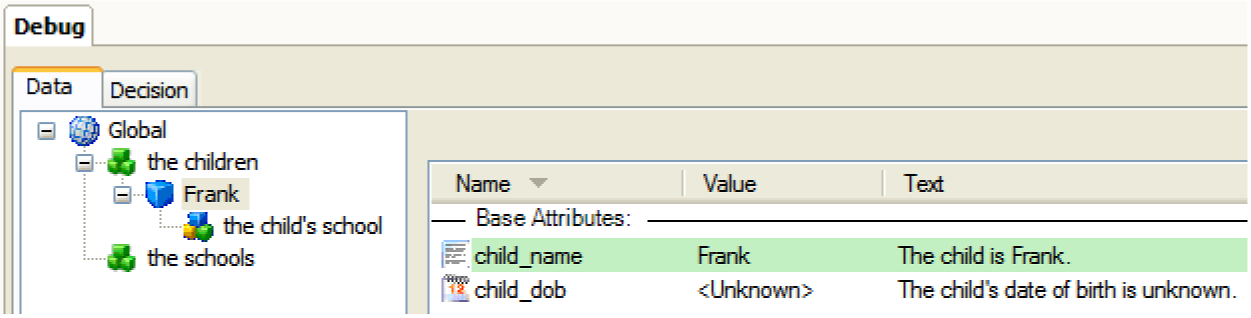
Add entity instances

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).

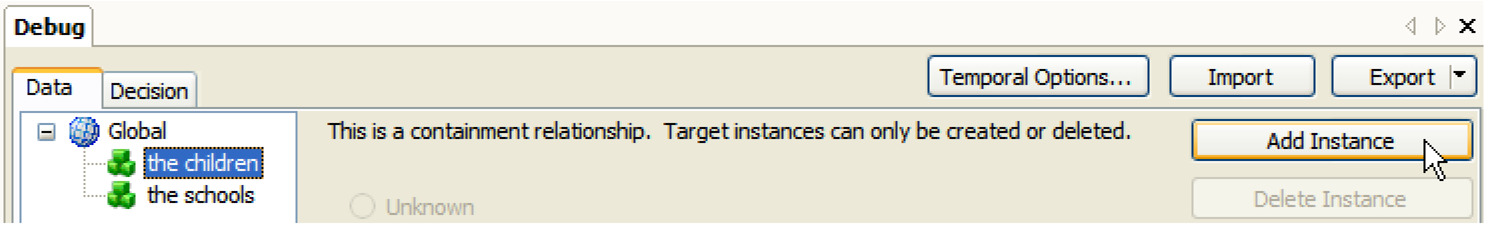


TIP: At this point it can be useful to provide a value for the identifying attribute for each of the entity instances. This will make it easier to distinguish between the entity instances when debugging. In the example above, the child's name attribute is the identifying attribute for the child entity. 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:



Name	Value	Text
Base Attributes:		
child_name	Frank	The child is Frank.
child_dob	<Unknown>	The child's date of birth is unknown.

You can also add a new entity instance by selecting the containment relationship and using the **Add Instance** button in the relationship editor:



This is a containment relationship. Target instances can only be created or deleted.

For entities contained within other entities, instances are created in the same way as above, using the containment relationships within the existing instances.

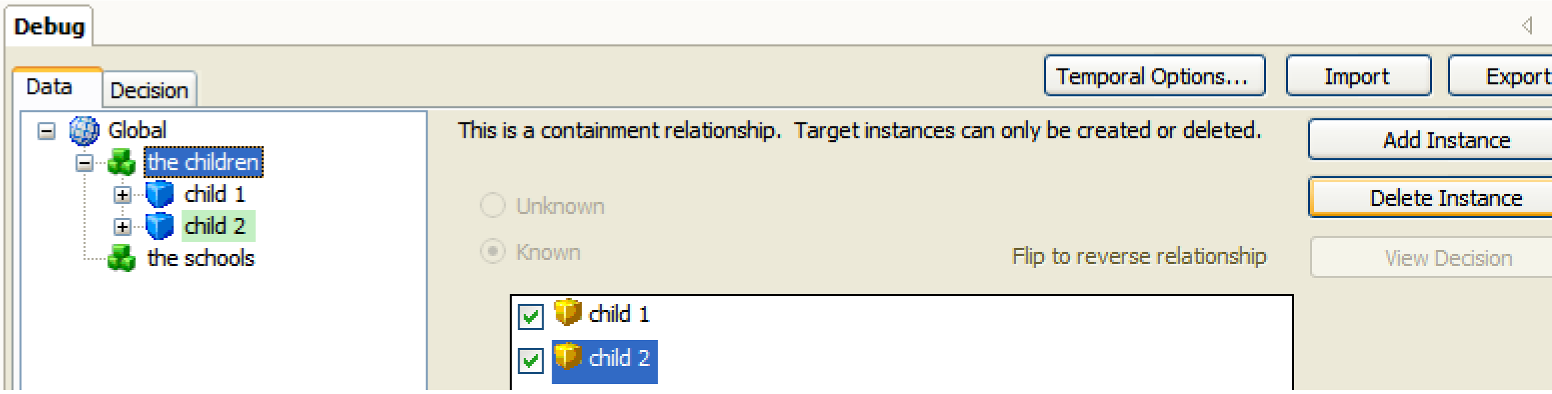
Delete entity instances

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:



This is a containment relationship. Target instances can only be created or deleted.

☑ child 1
☑ child 2

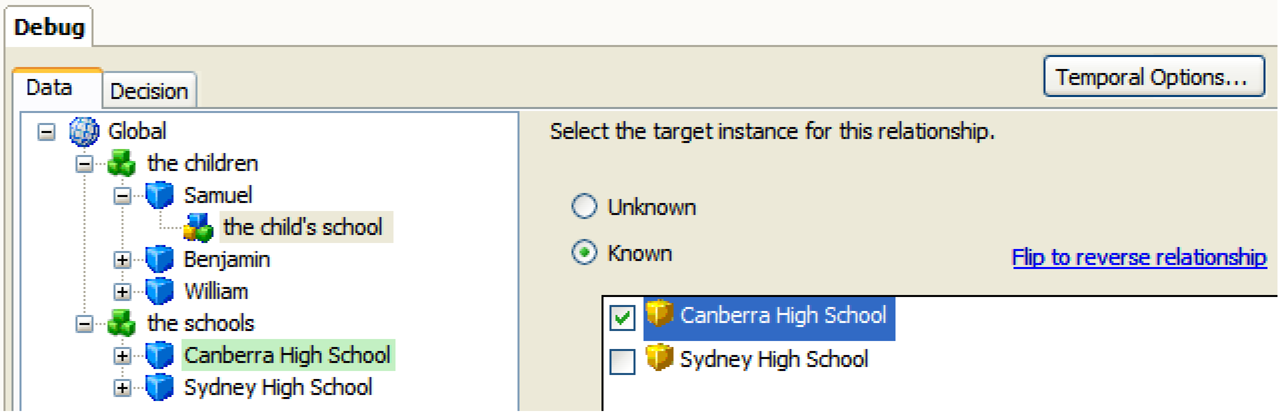
Set reference relationships

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.



Select the target instance for this relationship.

○ Unknown
● Known

☑ Canberra High School
☐ Sydney High School

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:

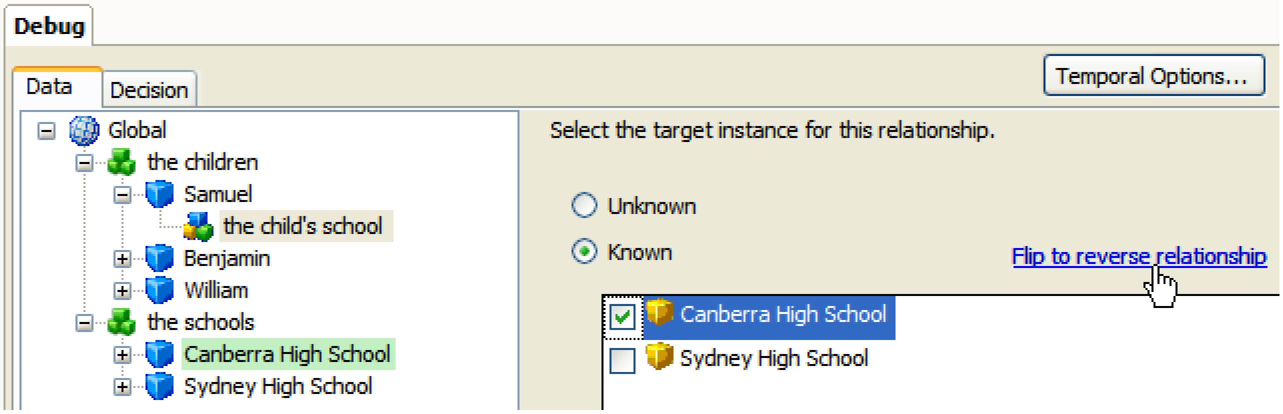
1. Select the relationship in the left hand pane of the Data view.
2. In the relationship editor in the right hand pane, deselect the check box for any entity instances that you no longer want associated with that relationship.

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.

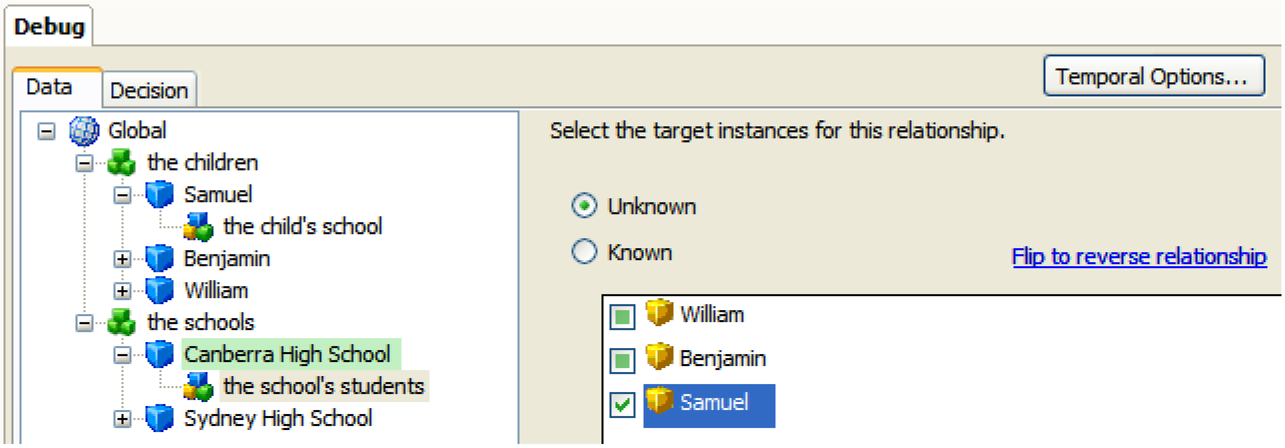


Select the target instance for this relationship.

○ Unknown
● Known

☑ Canberra High School
☐ Sydney High School

2. 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.



Select the target instances for this relationship.

● Unknown
○ Known

☑ William
☑ Benjamin
☑ Samuel

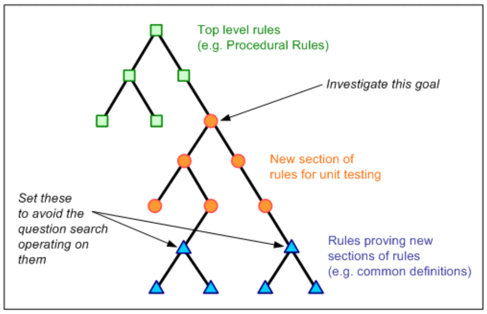
See also:

- 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.

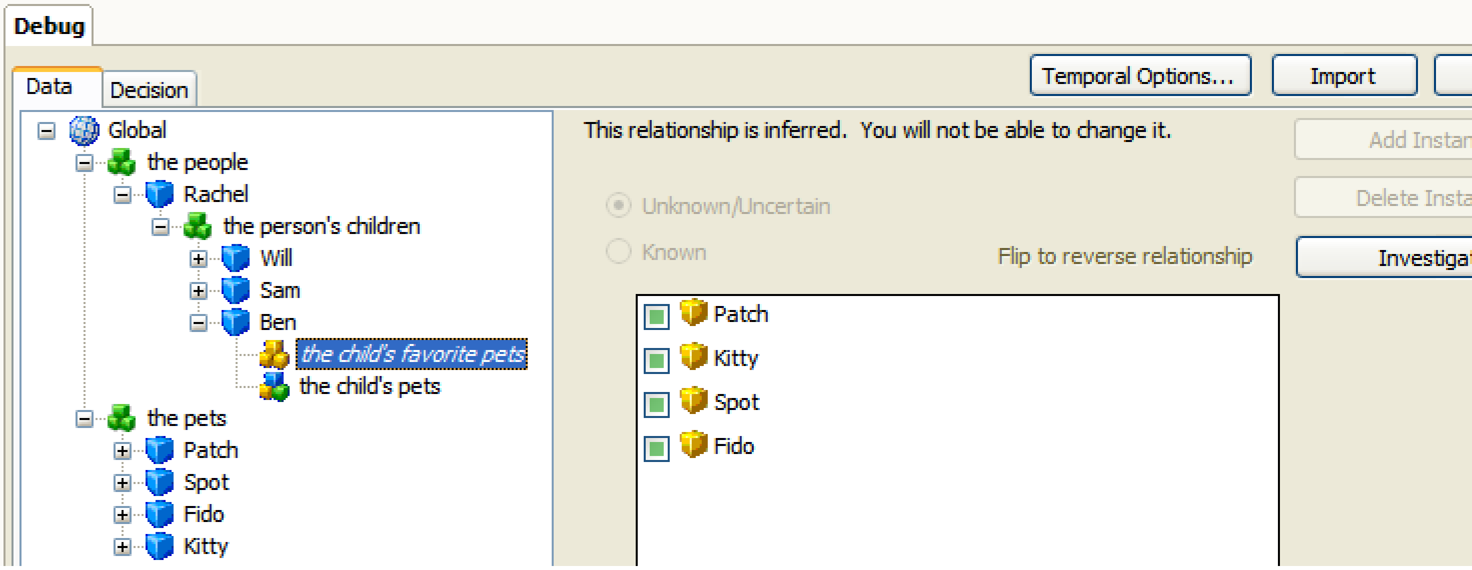
3. Work your way through the list of questions, setting answers (see above). In order to investigate any attributes which belong to an entity, you will need to add instances of that entity. Add your entity instances and continue investigating attributes until a value for the goal is known.

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.

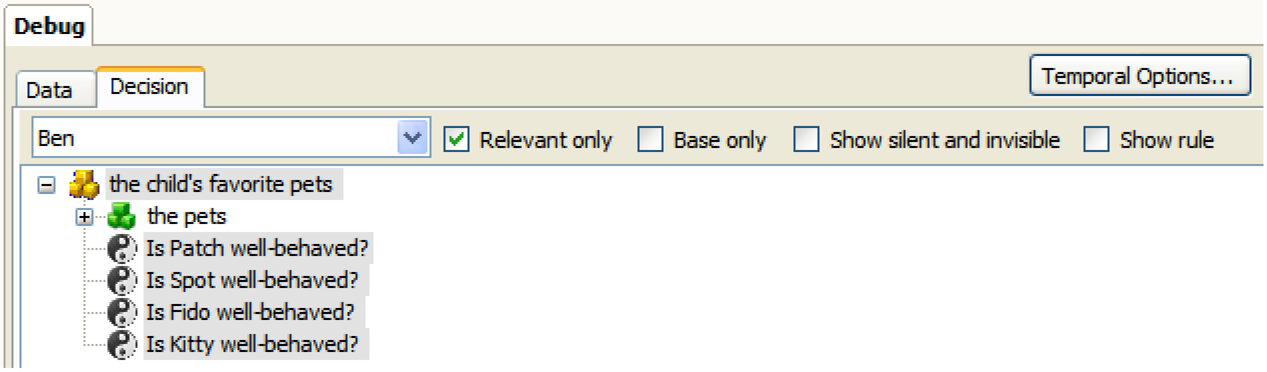


This relationship is inferred. You will not be able to change it.

○ Unknown/Uncertain
○ Known

🟩 Patch
🟩 Kitty
🟩 Spot
🟩 Fido

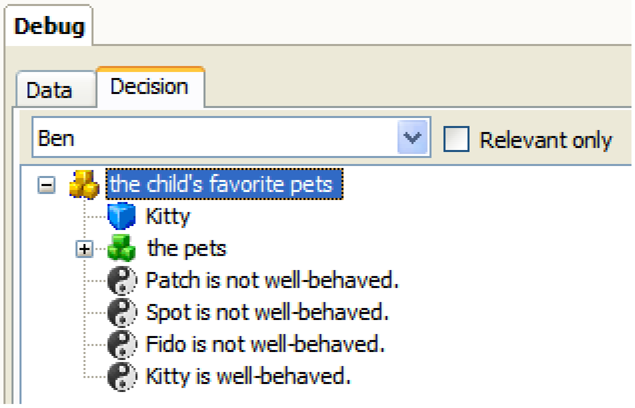
2. In the right hand pane, click the **Investigate** button. This will switch to the Decision view.



Ben

☑ Relevant only ☐ Base only ☐ Show silent and invisible ☐ Show rule

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.

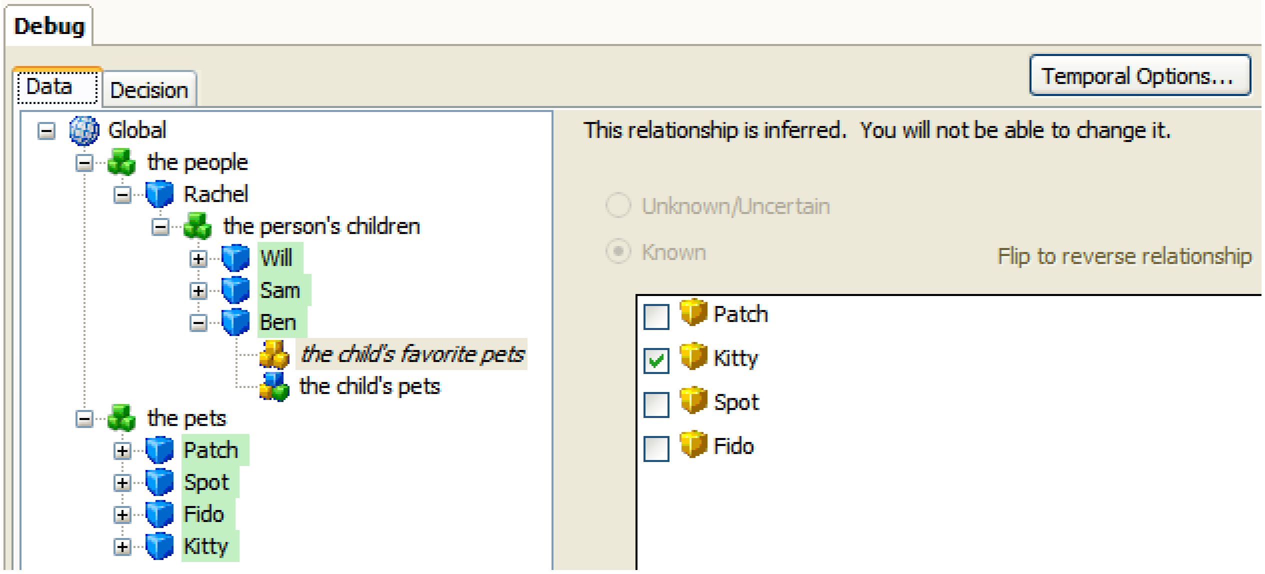


Ben

☐ Relevant only

- the child's favorite pets
  - Kitty
  - the pets
  - Patch is not well-behaved.
  - Spot is not well-behaved.
  - Fido is not well-behaved.
  - Kitty is well-behaved.

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).



- Global
  - the people
    - Rachel
      - the person's children
        - Will
        - Sam
        - Ben
          - *the child's favorite pets*
          - the child's pets
  - the pets
    - Patch
    - Spot
    - Fido
    - Kitty

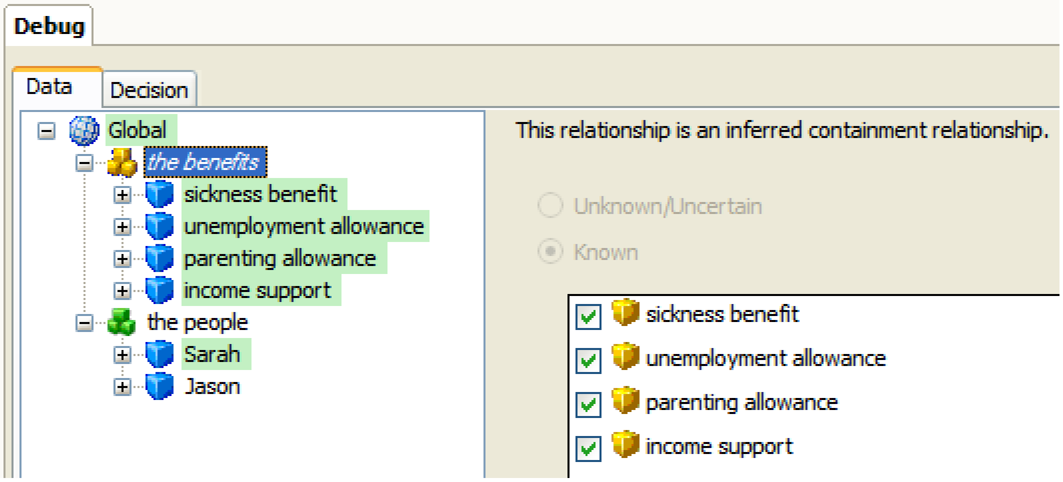
This relationship is inferred. You will not be able to change it.

○ Unknown/Uncertain

● Known

☐ Patch
☑ Kitty
☐ Spot
☐ Fido

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.



This relationship is an inferred containment relationship.

○ Unknown/Uncertain

● Known

☑ sickness benefit

☑ unemployment allowance

☑ parenting allowance

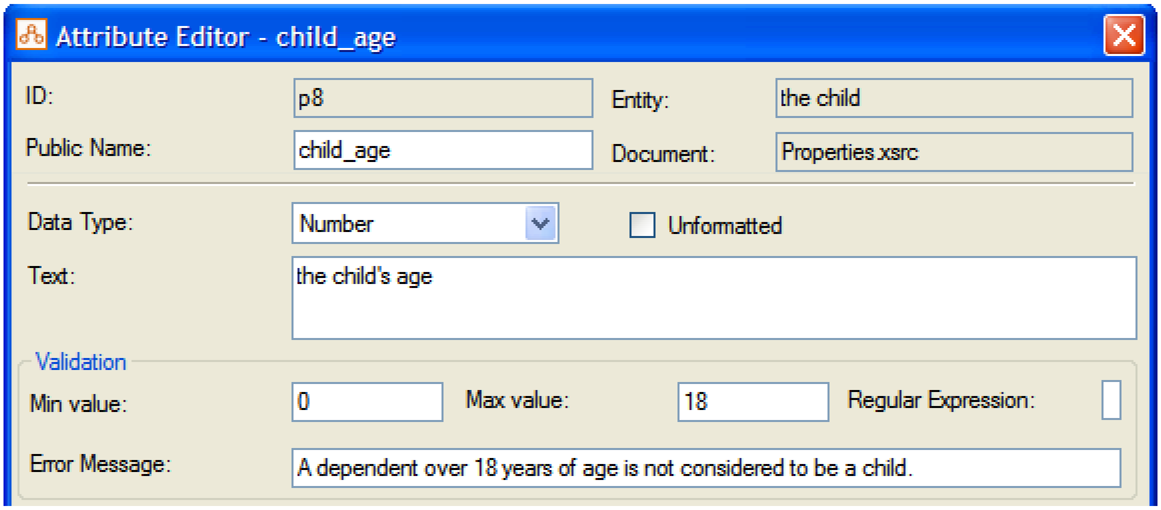
☑ income support

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:



ID: p8

Entity: the child

Public Name: child_age

Document: Properties.xsrc

Data Type: Number

☐ Unformatted

Text: the child's age

Min value: 0

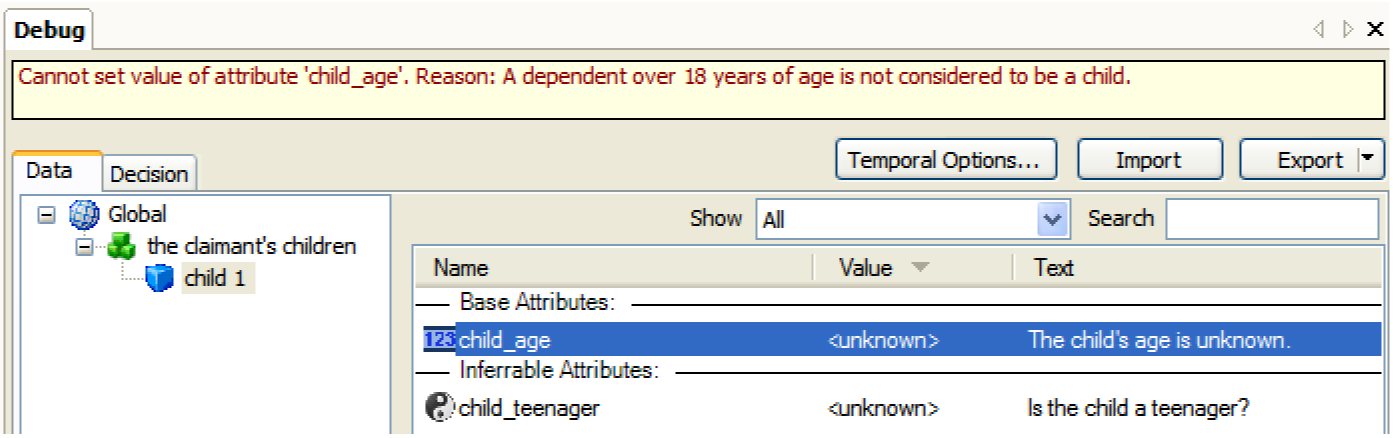
Max value: 18

Regular Expression:

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

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:



Cannot set value of attribute 'child_age'. Reason: A dependent over 18 years of age is not considered to be a child.

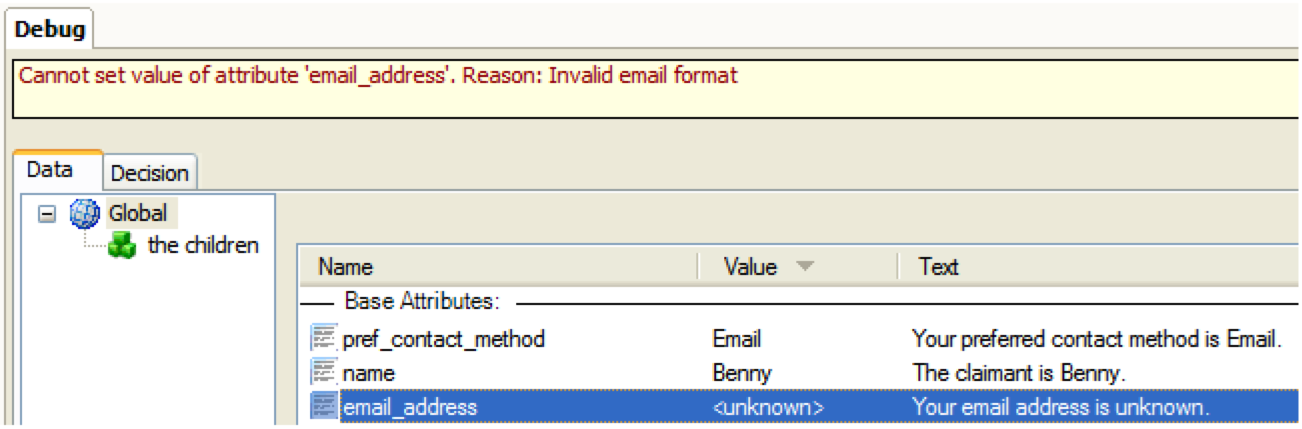
- Global
  - the claimant's children
    - child 1

Name	Value	Text
Base Attributes:		
child_age	<unknown>	The child's age is unknown.
Inferrable Attributes:		
child_teenager	<unknown>	Is the child a teenager?

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:



Cannot set value of attribute 'email_address'. Reason: Invalid email format

- Global
  - the children

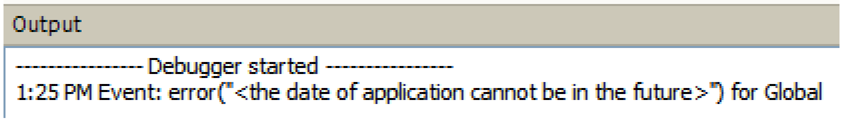
Name	Value	Text
Base Attributes:		
pref_contact_method	Email	Your preferred contact method is Email.
name	Benny	The claimant is Benny.
email_address	<unknown>	Your email address is unknown.

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.



```
---------------- Debugger started ----------------
1:25 PM Event: error("<the date of application cannot be in the future>") for Global
```

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.

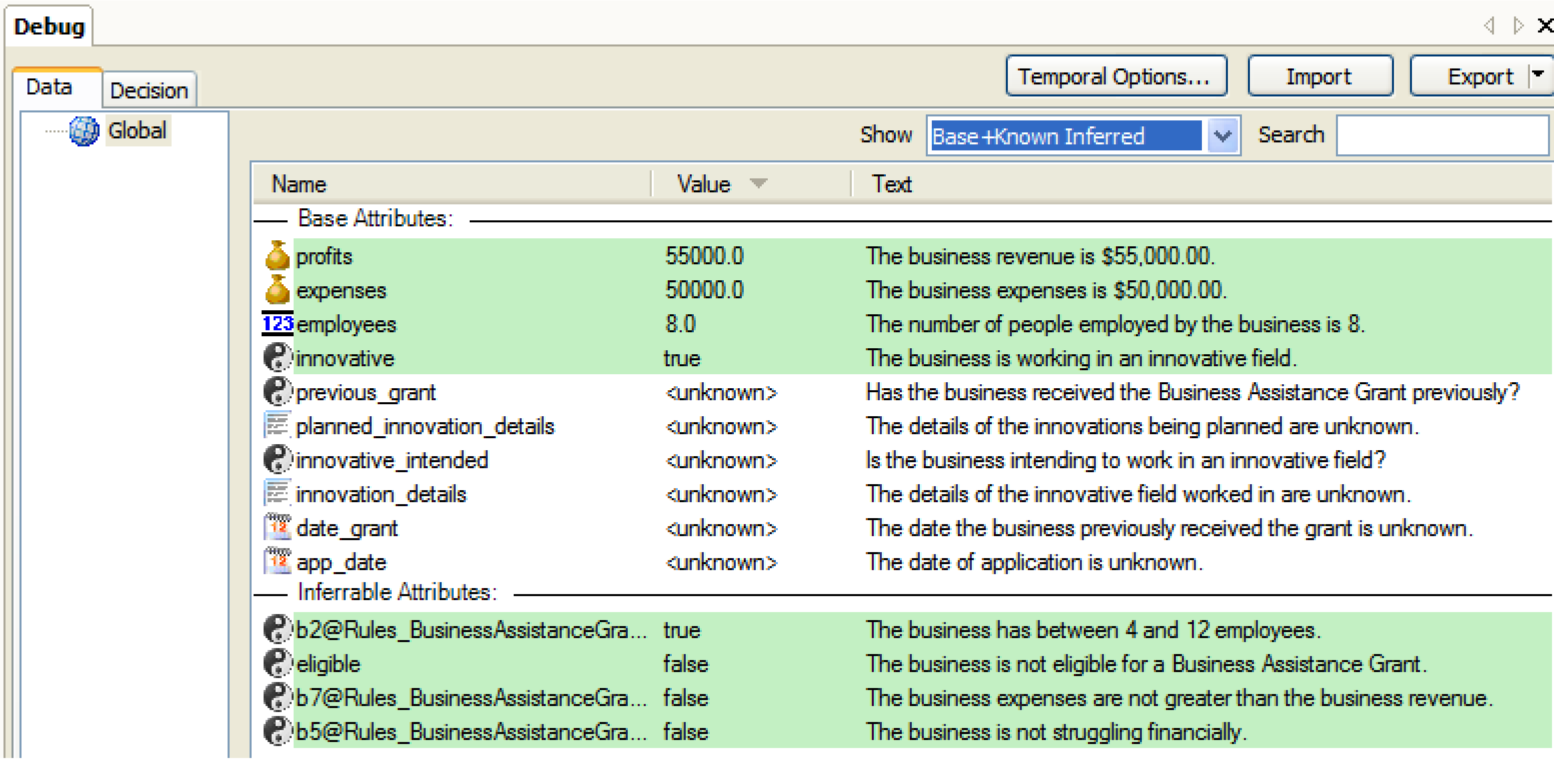
What do you want to do?

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:



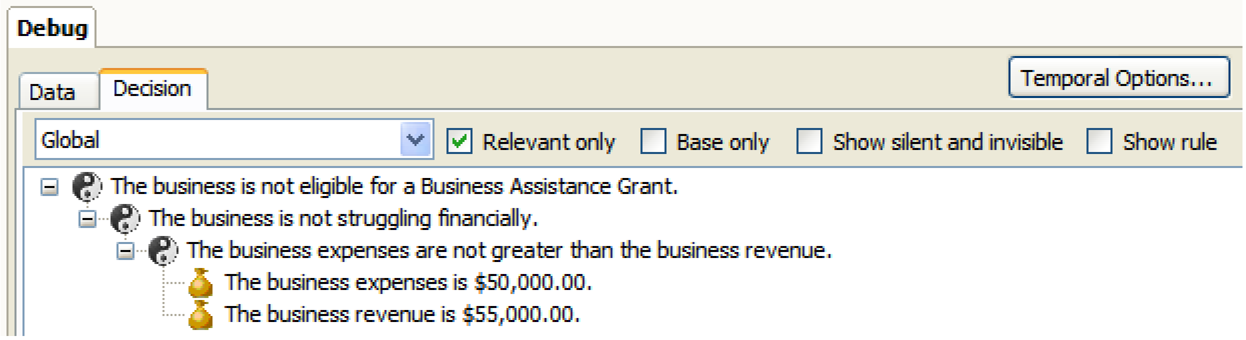
Show Base+Known Inferred

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.
Inferrable 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

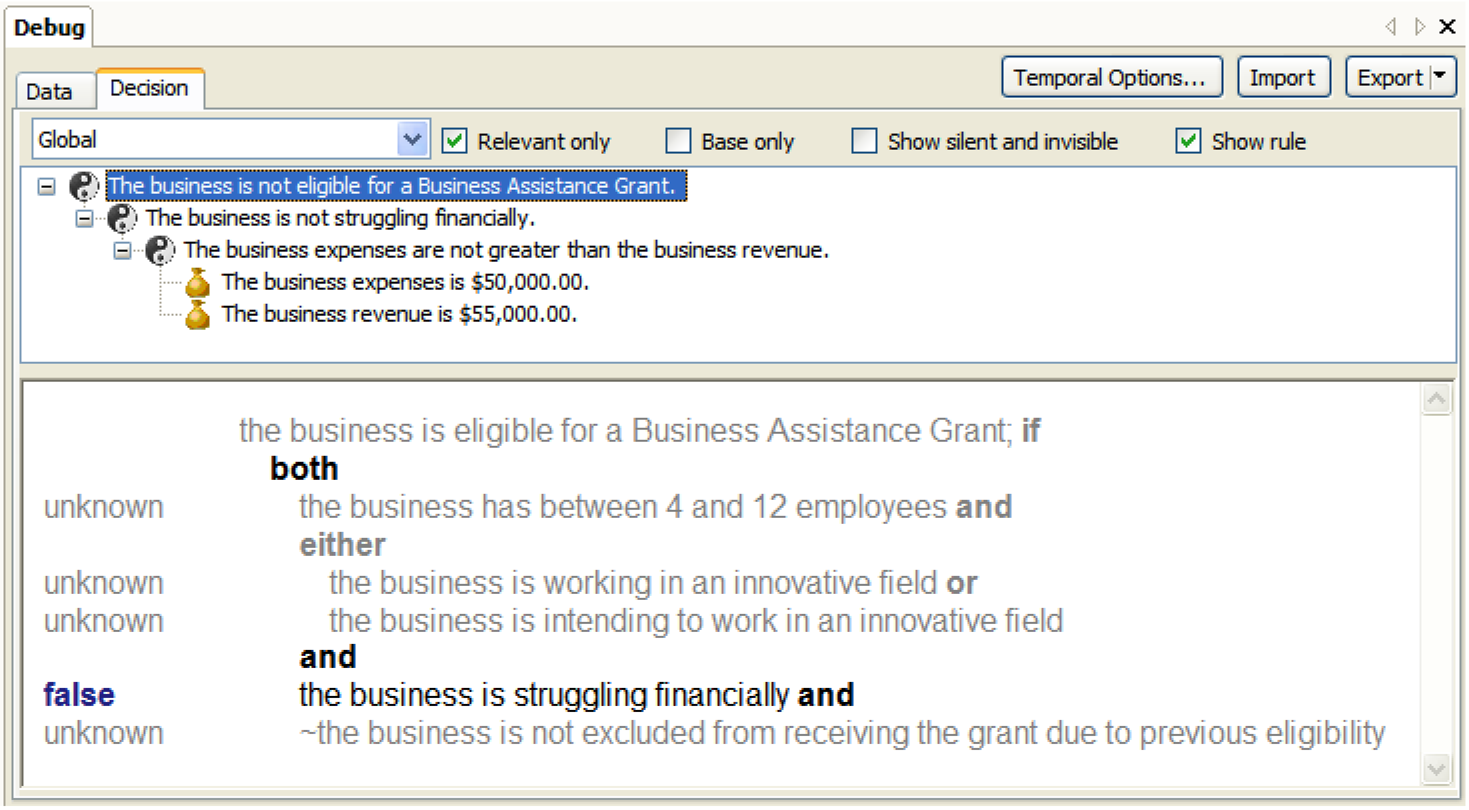
To view the decision for an attribute with a known value, you can right-click the attribute in the Data view and select **Show Decision**. This will open the Decision view with the attribute you have selected in the **Attribute** field. The decision view appears like a decision report showing all the relevant paths that contributed to the goal attribute's value.



- The business is not eligible for a Business Assistance Grant.
  - The business is not struggling financially.
    - The business expenses are not greater than the business revenue.
      - The business expenses is $50,000.00.
      - The business revenue is $55,000.00.

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.



Global
☑ Relevant only ☐ Base only ☐ Show silent and invisible ☑ Show rule

- The business is not eligible for a Business Assistance Grant.
  - The business is not struggling financially.
    - The business expenses are not greater than the business revenue.
      - The business expenses is $50,000.00.
      - The business revenue is $55,000.00.

```
                the business is eligible for a Business Assistance Grant; if
                  both
unknown             the business has between 4 and 12 employees and
                    either
unknown               the business is working in an innovative field or
unknown               the business is intending to work in an innovative field
                    and
false               the business is struggling financially and
unknown             ~the business is not excluded from receiving the grant due to previous eligibility
```

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

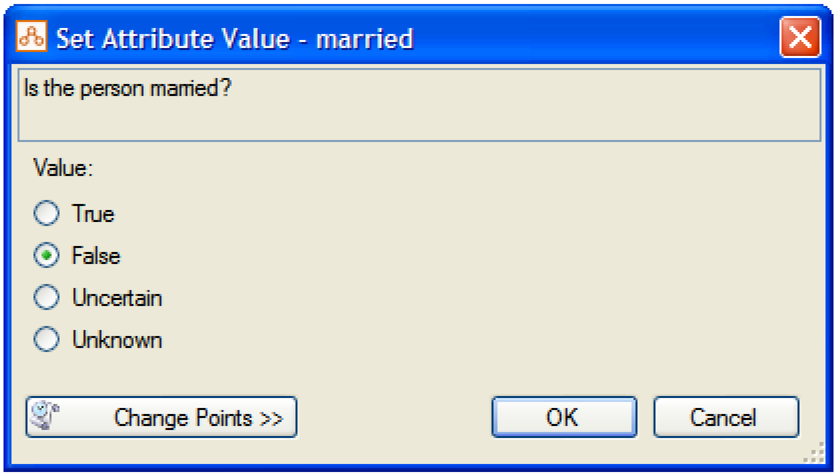
Understand temporal outcomes

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.)



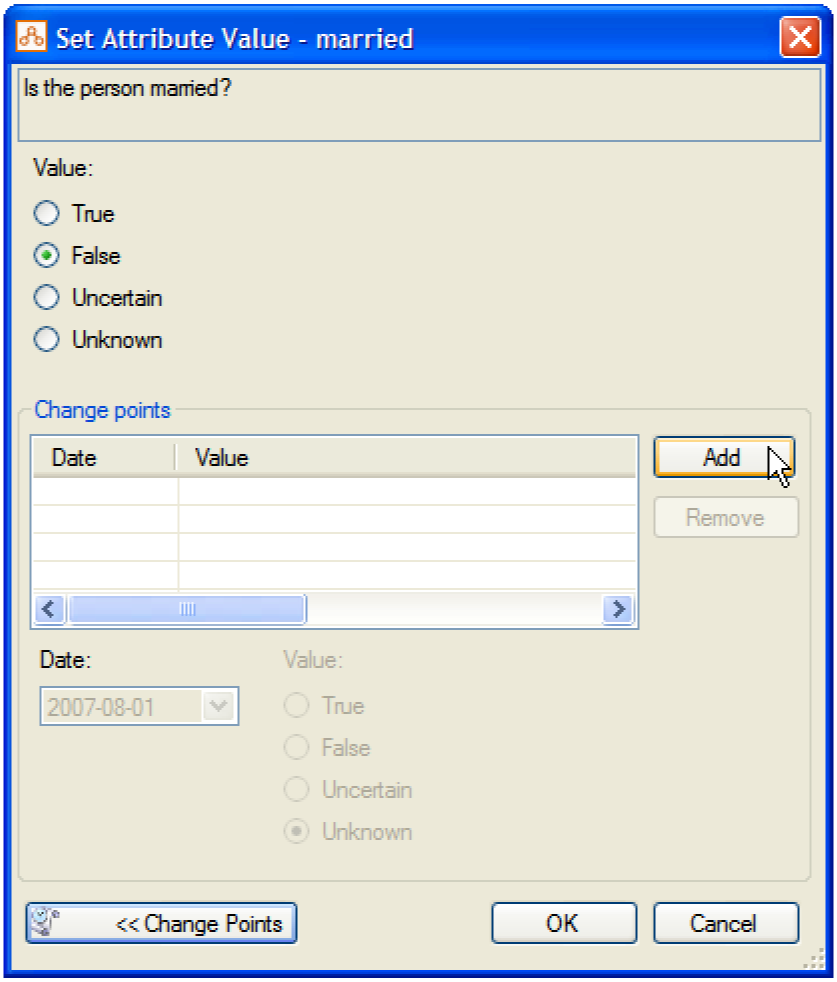
Set Attribute Value - married

Is the person married?

Value:

○ True
● False
○ Uncertain
○ Unknown

3. Select the **Change Points** button. This expands the dialog box so that you can add Change Points for the attribute.



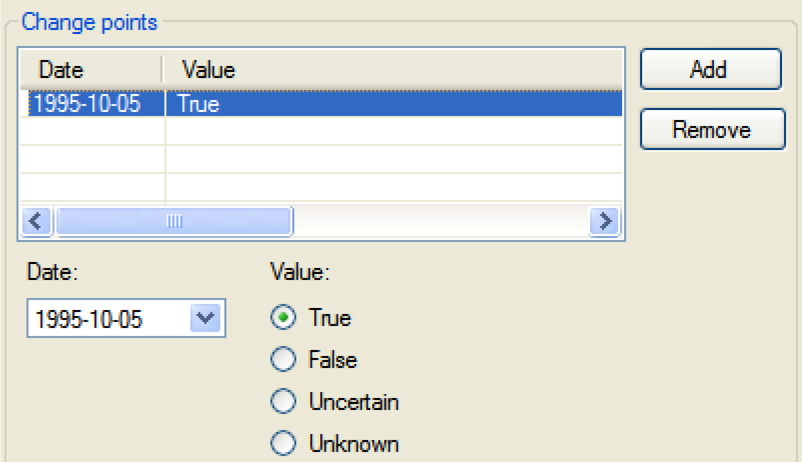
Is the person married?

Value:
○ True
● False
○ Uncertain
○ Unknown

Date: 2007-08-01

Value:
○ True
○ False
○ Uncertain
● Unknown

4. 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.)
5. 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.

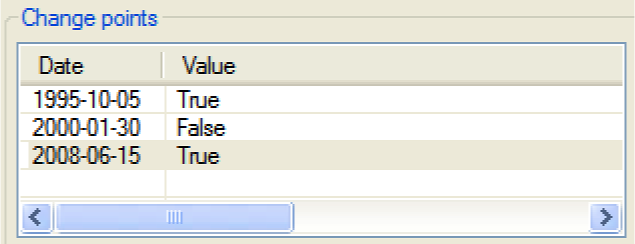


Date	Value
1995-10-05	True

Date: 1995-10-05

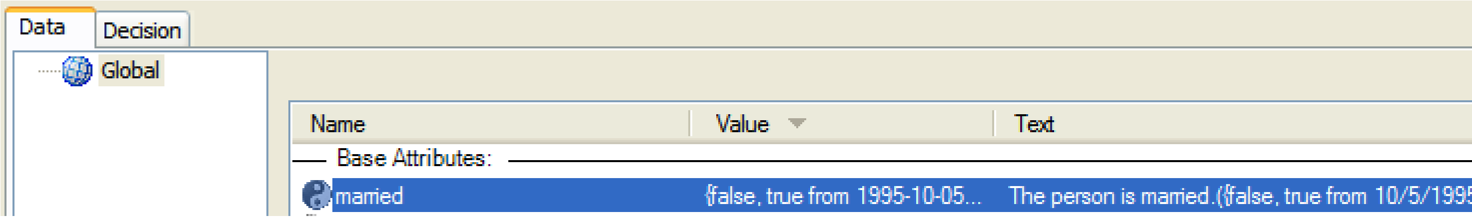
Value:
● True
○ False
○ Uncertain
○ Unknown

6. To add additional change points, repeat steps 4 and 5.



Date	Value
1995-10-05	True
2000-01-30	False
2008-06-15	True

7. 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.



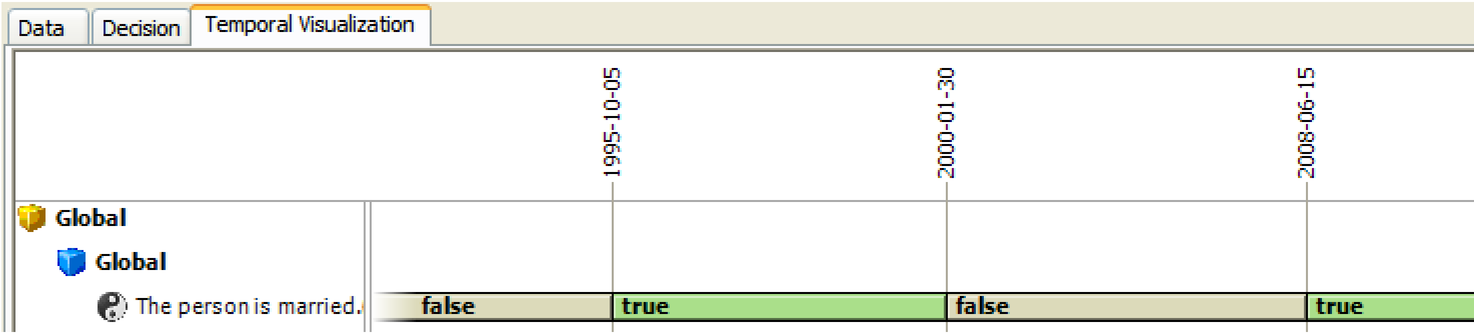
Name	Value	Text
Base Attributes:		
married	{false, true from 1995-10-05...	The person is married.({false, true from 10/5/1995

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.

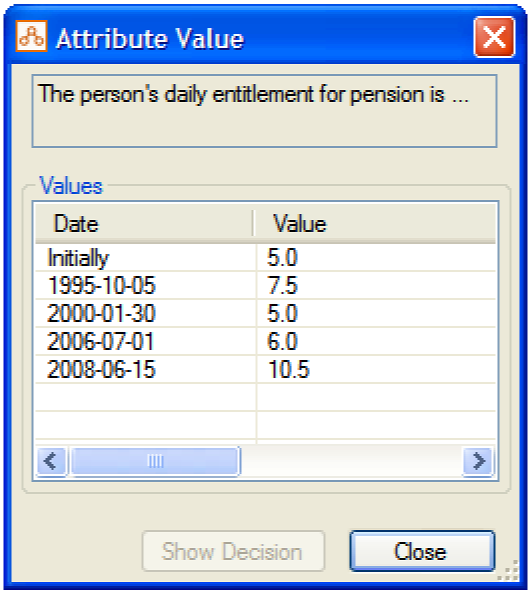
Understand temporal outcomes

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.



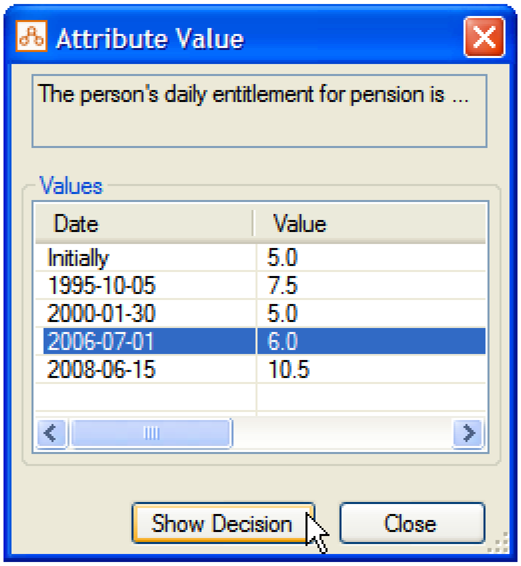
The person's daily entitlement for pension is ...

Date	Value
Initially	5.0
1995-10-05	7.5
2000-01-30	5.0
2006-07-01	6.0
2008-06-15	10.5

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:

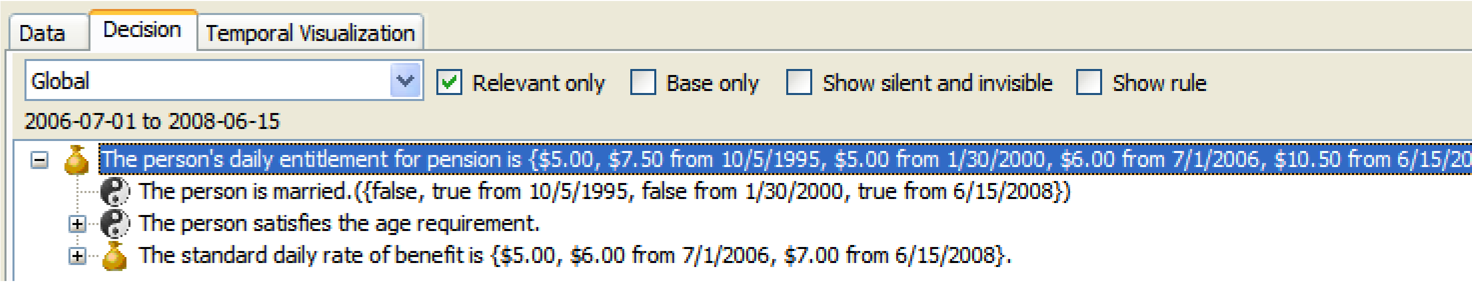
1. Select the date/value in the **Attribute Value** dialog box and then click the **Show Decision** button.



The person's daily entitlement for pension is ...

Date	Value
Initially	5.0
1995-10-05	7.5
2000-01-30	5.0
2006-07-01	6.0
2008-06-15	10.5

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.



☑ Relevant only ☐ Base only ☐ Show silent and invisible ☐ Show rule

2006-07-01 to 2008-06-15

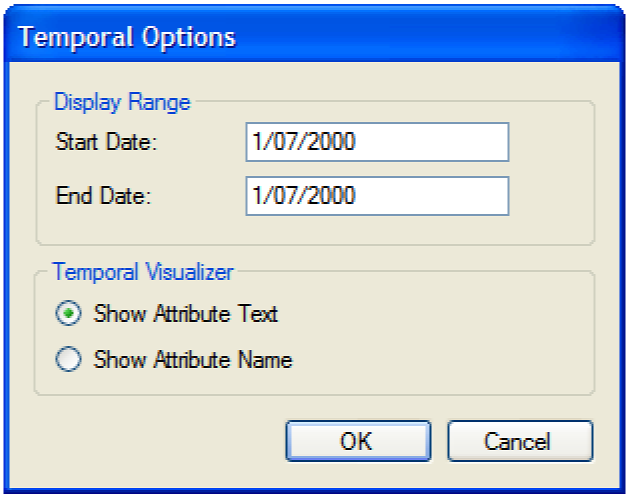
- The person's daily entitlement for pension is {$5.00, $7.50 from 10/5/1995, $5.00 from 1/30/2000, $6.00 from 7/1/2006, $10.50 from 6/15/20
  - The person is married.({false, true from 10/5/1995, false from 1/30/2000, true from 6/15/2008})
  - The person satisfies the age requirement.
  - The standard daily rate of benefit is {$5.00, $6.00 from 7/1/2006, $7.00 from 6/15/2008}.

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.



Start Date: 1/07/2000

End Date: 1/07/2000

● Show Attribute Text

○ Show Attribute Name

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

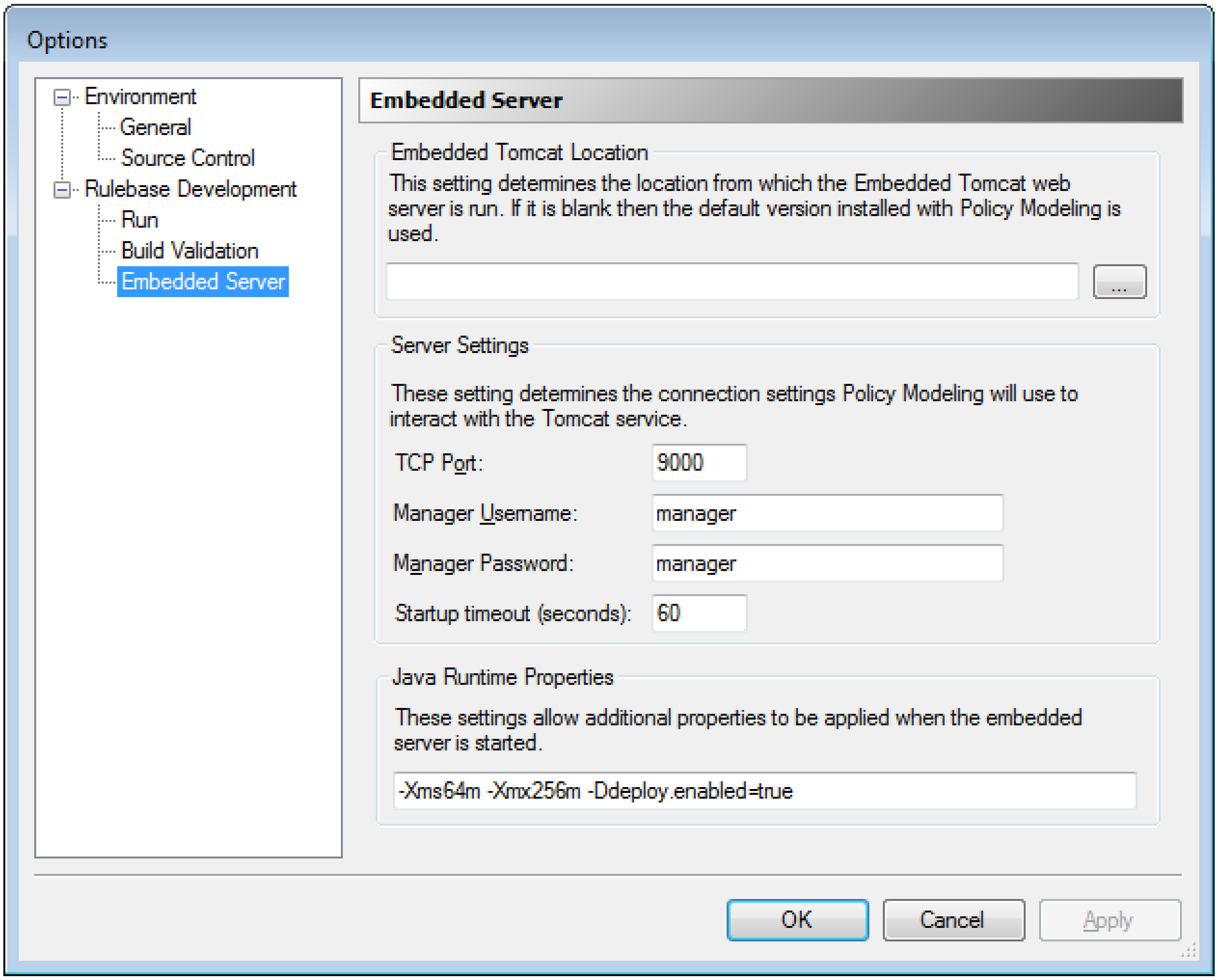
You can deploy a rulebase or interview to Oracle Determinations Server. This is handy for demonstration purposes, as well as a configuration-less testing platform for Determinations Server.

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**.



Embedded Server

Embedded Tomcat Location

This setting determines the location from which the Embedded Tomcat web server is run. If it is blank then the default version installed with Policy Modeling is used.

Server Settings

These setting determines the connection settings Policy Modeling will use to interact with the Tomcat service.

TCP Port: 9000

Manager Username: manager

Manager Password: manager

Startup timeout (seconds): 60

Java Runtime Properties

These settings allow additional properties to be applied when the embedded server is started.

-Xms64m -Xmx256m -Ddeploy.enabled=true

2. (Optional) To specify a location for the Embedded Tomcat web server that is different to the default location, type the file path to the new location (for example, C:\Program Files\apache-tomcat-5.5.28-embed) into the **Tomcat Path** field in the **Embedded Tomcat Location** area of the tab
3. (Optional) To specify a different TCP port for the server from the default port, type the new port number into the **TCP Port** field in the **Server Settings** area of the tab.
4. (Optional) To specify a different username and / or password for the server manager, type the new username into the **Manager Username** field and the new password into the **Manager Password** field in the **Server Settings** area of the tab.
5. (Optional) To specify a different startup timeout, type the new timeout period (in seconds) into the **Startup timeout (seconds)** field in the **Server Settings** area of the tab.
6. (Optional) To change the amount of memory available to the embedded server, edit the **Xms** and **Xmx** settings in the **Properties** field in the **Server Runtime Properties** area of the tab. For example, to increase the available memory, replace -Xms64m -Xmx256m with -Xms256m -Xmx512m. Increasing the available memory will improve server performance for projects with a large number of inferred entity instances, relationships, or aliases, or with complex

document generation. Tip: Once you change this setting, the results are saved locally. This means you will not need to change the memory size again when you install a later version of Policy Modeling. For a new user who has not previously run Policy Modeling 10.4 on their machine, the default setting is -Xms64m -Xmx256m.

7. When you have finished editing the settings, 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\<extension name>. A file called extensions.xml contains the definition of the available custom functions, and the custom function itself is located within a \lib subfolder.

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\<extension name> project folder. In addition:

- 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[1]
- Common rules that are used by several rulebases read more[2]
- Core rules that are deployed in different ways for distinct audiences read more[3]
- Different combinations of the same rules used by multiple rulebases read more[4]

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

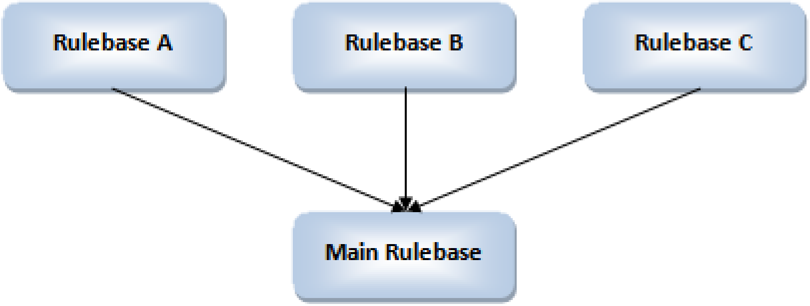
Add a link to a module

To create a link to a module from your rulebase:

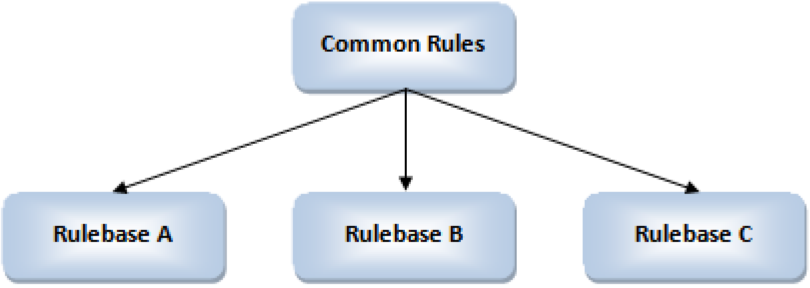
1. In the Project Explorer in Oracle Policy Modeling, select the folder that you would like the module link to be placed in (eg the Modules folder), right-click and select **Add Module Link...**

---

[1] Modules allow teams to work independently on sub-components of a ruleset and combine their work into a single rulebase.

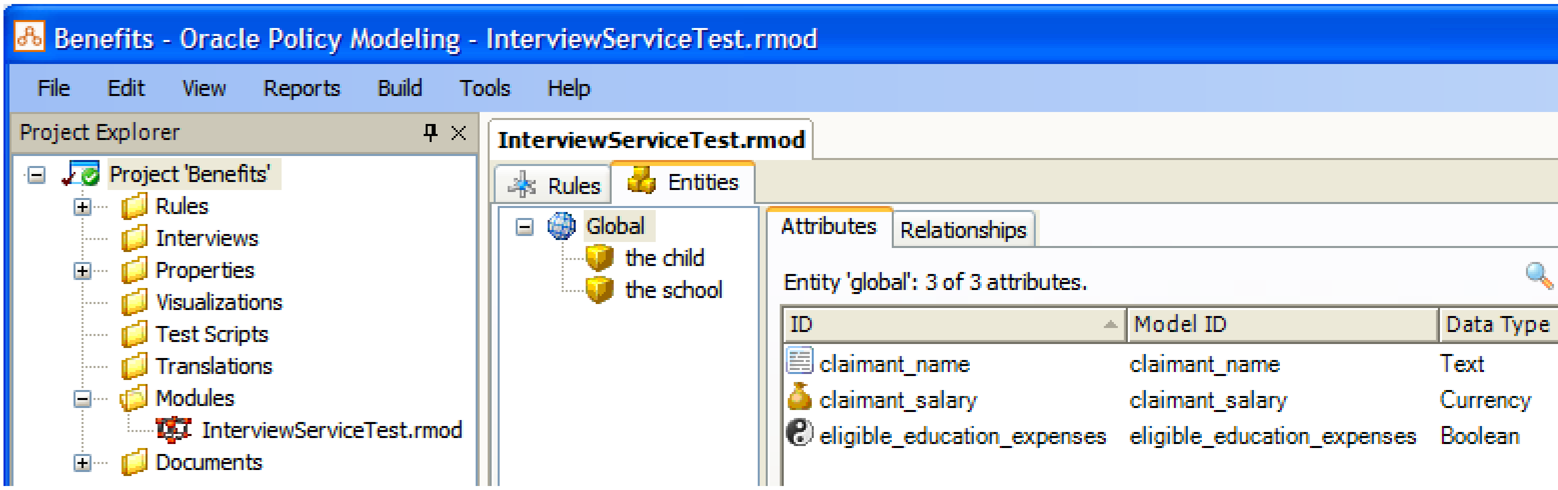


[2] Modules allow rule developers to import a common set of rules and screens into other rulebases, avoiding the need to create and maintain common questions in multiple places.



2. 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.



Entity 'global': 3 of 3 attributes.

ID	Model ID	Data Type
claimant_name	claimant_name	Text
claimant_salary	claimant_salary	Currency
eligible_education_expenses	eligible_education_expenses	Boolean

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.

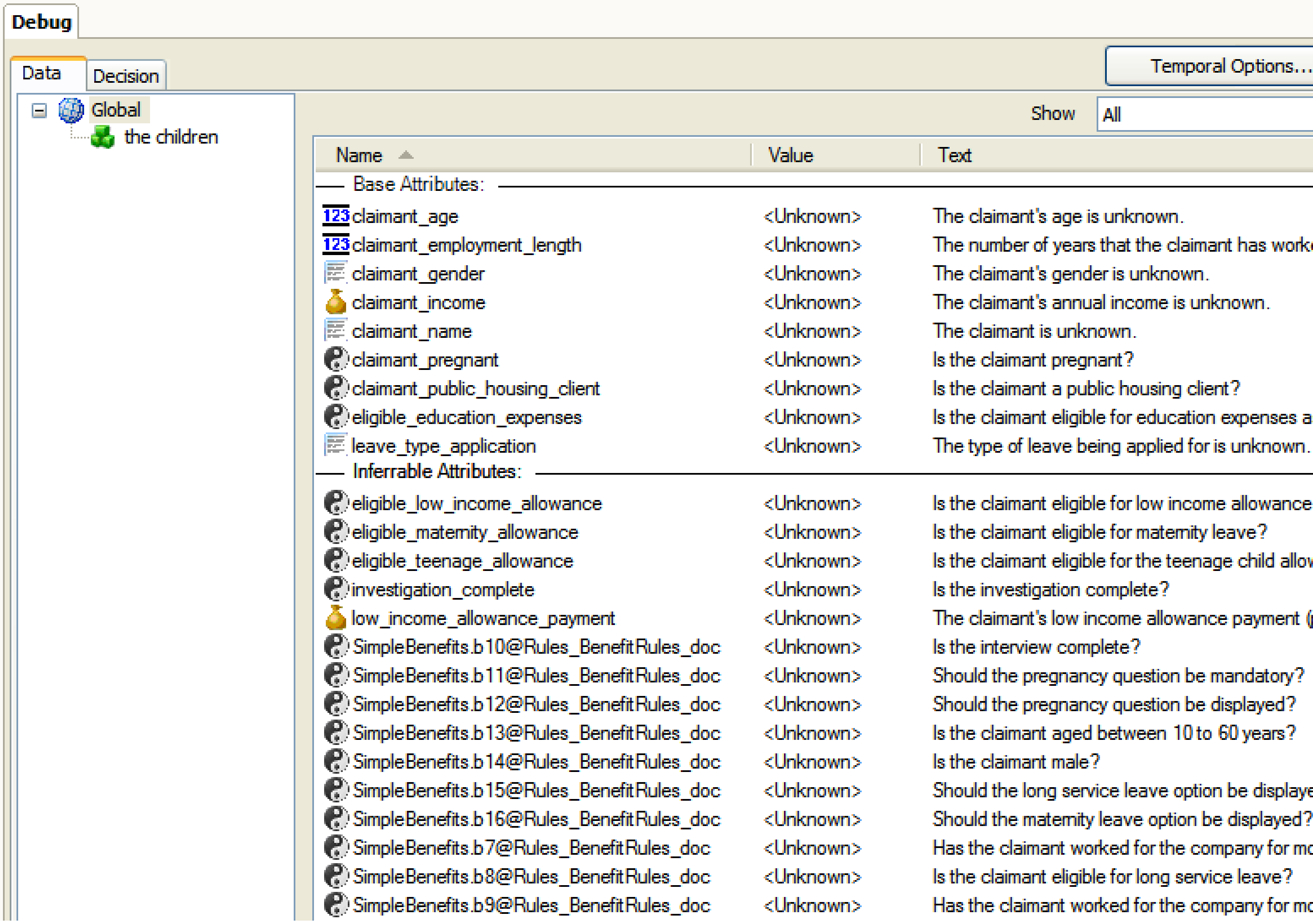
Redeployment

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
Base Attributes:		
claimant_age	<Unknown>	The claimant's age is unknown.
claimant_employment_length	<Unknown>	The number of years that the claimant has work
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 a
leave_type_application	<Unknown>	The type of leave being applied for is unknown.
Inferrable Attributes:		
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 allow
investigation_complete	<Unknown>	Is the investigation complete?
low_income_allowance_payment	<Unknown>	The claimant's low income allowance payment (
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 displaye
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 mo
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 mo

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:

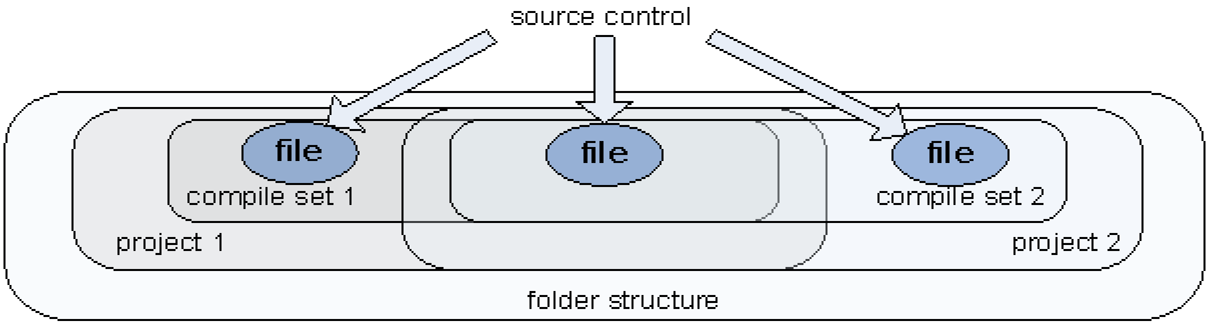
- have multiple Oracle Policy Modeling projects in one folder structure;
- share the files in source control; or
- duplicate the files.

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.

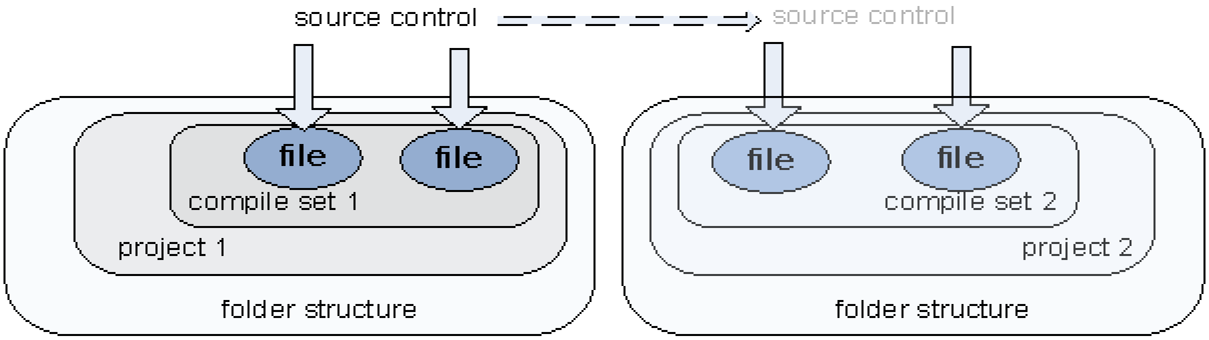
To be in the compile set, a file must be in the project, and to be in the project, a file must be in the folder structure. Source control operates as usual for a single application.



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

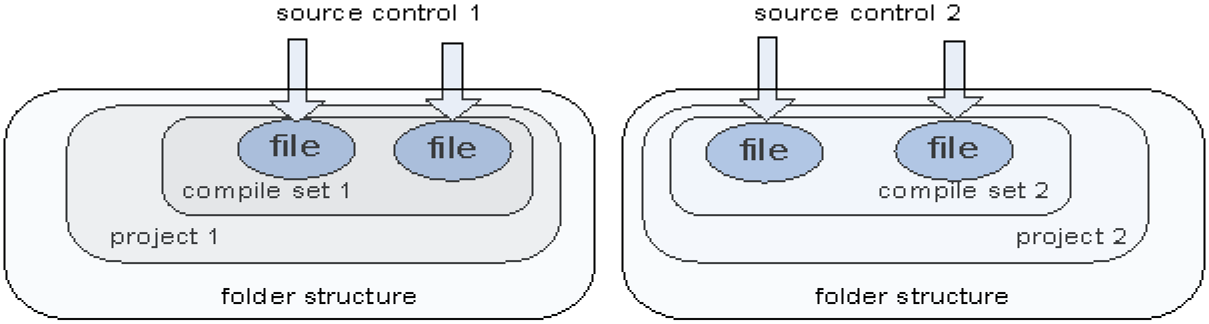
Another method of applying the rules and/or xsrc files in a Oracle Policy Modeling project is to duplicate the files in the file system but synchronize them in source control.



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[1].

---

[1] Metadata here refers to custom properties (ie those defined in the custom properties tab of an attribute, entity or relatiomship) and intrinsic properties (ie non-custom properties that do not relate to the item's text, type or name)

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**.

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

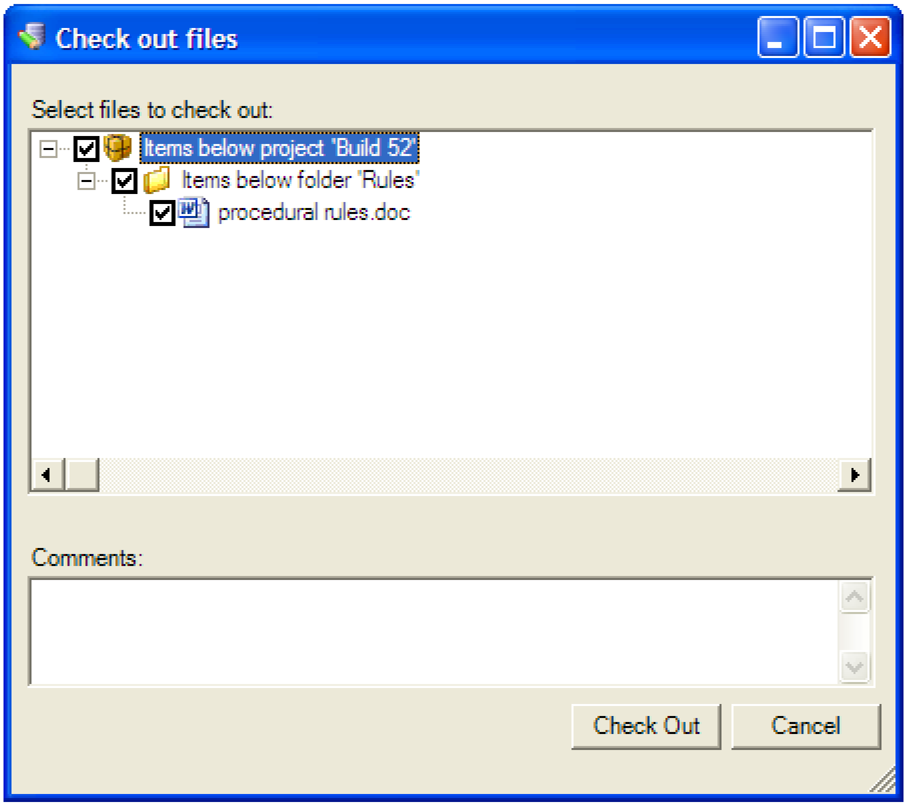
Ensure all documents are checked in when the project is closed

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.

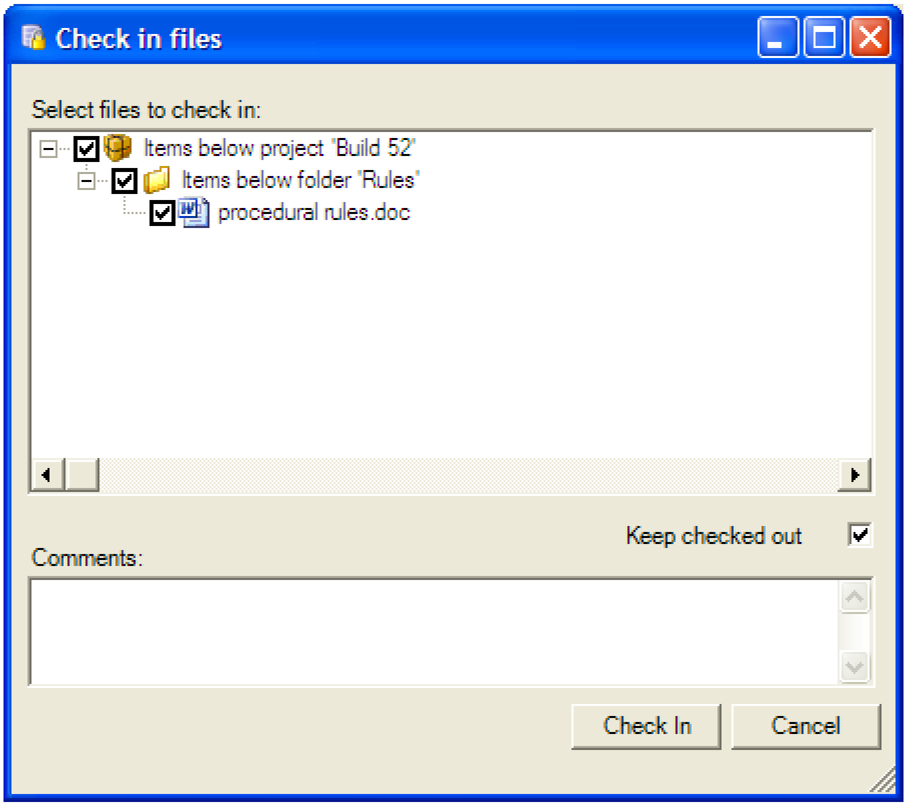
NOTE: Attempting to edit checked-in screen (*.xint) and properties files (*.xsrc) will automatically prompt you to check out the file.

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.



Select files to check in:

- ☑ Items below project 'Build 52'
  - ☑ Items below folder 'Rules'
    - ☑ procedural rules.doc

Keep checked out ☑

Comments:

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.

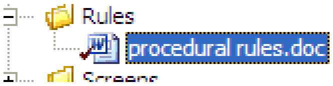
See whether a document is checked in or out

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.

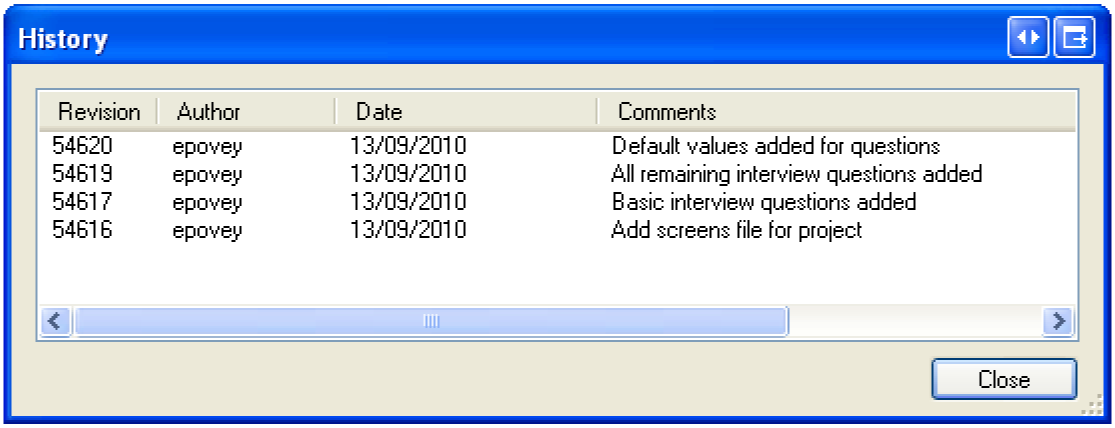
A tick on a project file icon shows that a file is checked out.



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.



Revision	Author	Date	Comments
54620	epovey	13/09/2010	Default values added for questions
54619	epovey	13/09/2010	All remaining interview questions added
54617	epovey	13/09/2010	Basic interview questions added
54616	epovey	13/09/2010	Add screens file for project

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

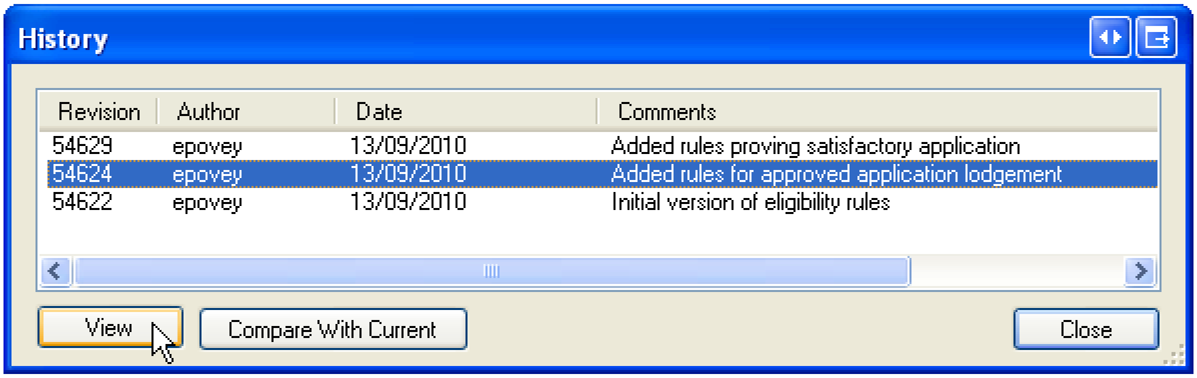
Compare versions of Word rule documents

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.



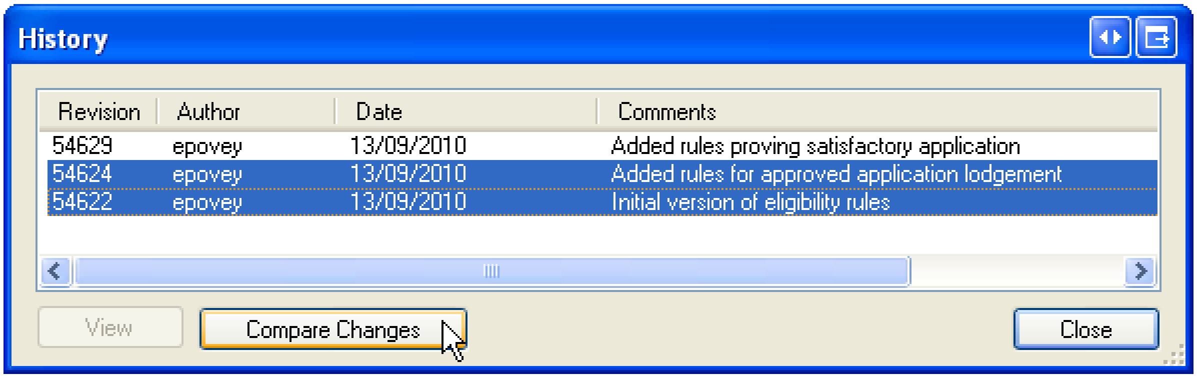
Revision	Author	Date	Comments
54629	epovey	13/09/2010	Added rules proving satisfactory application
54624	epovey	13/09/2010	Added rules for approved application lodgement
54622	epovey	13/09/2010	Initial version of eligibility rules

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

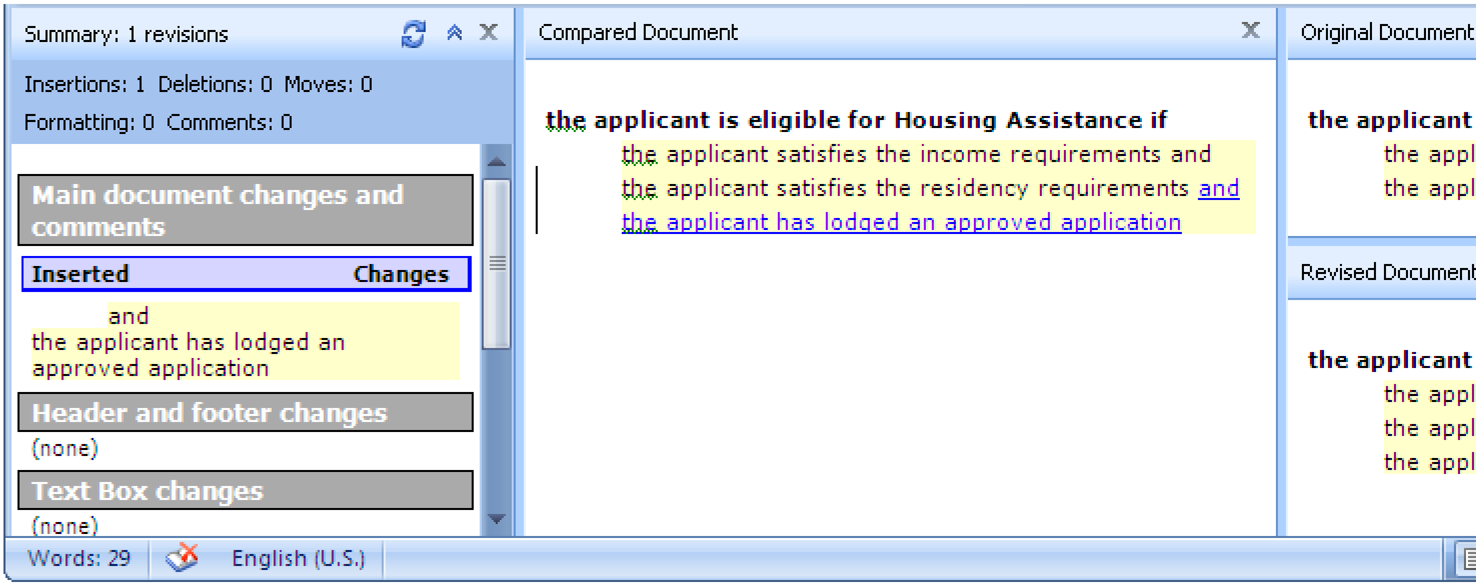
If you are using Subversion for your project source control, you can also compare any historical version of a rule document with the current document, or compare two historical versions.

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.



Revision	Author	Date	Comments
54629	epovey	13/09/2010	Added rules proving satisfactory application
54624	epovey	13/09/2010	Added rules for approved application lodgement
54622	epovey	13/09/2010	Initial version of eligibility rules

3. Word is opened showing the two document versions and highlighting the changes between the two.



Inserted and the applicant has lodged an approved application

the applicant is eligible for Housing Assistance if
the applicant satisfies the income requirements and
the applicant satisfies the residency requirements and
the applicant has lodged an approved application

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

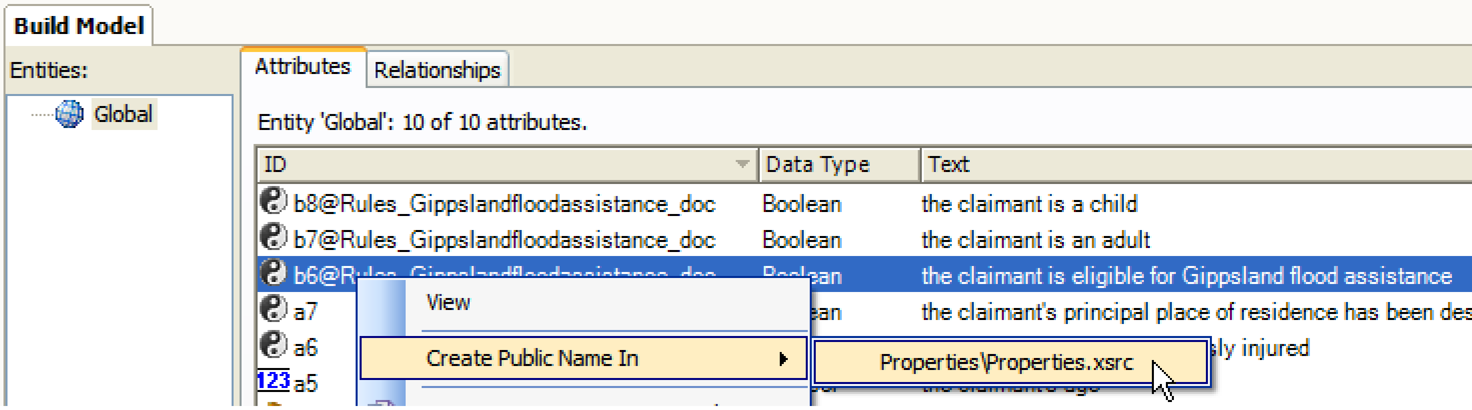
Set public identifiers for entities and attributes

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.

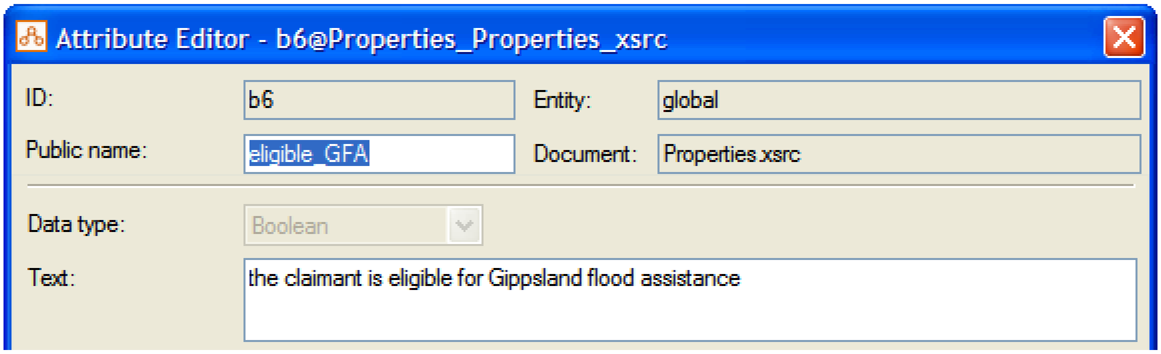


Entity 'Global': 10 of 10 attributes.

ID	Data Type	Text
b8@Rules_Gippslandfloodassistance_doc	Boolean	the claimant is a child
b7@Rules_Gippslandfloodassistance_doc	Boolean	the claimant is an adult
b6@R		the claimant is eligible for Gippsland flood assistance
a7		the claimant's principal place of residence has been des
a6		ly injured
a5		

   The properties file should open to the selected attribute in the **Attribute Editor** dialog.

2. Define a public name for the attribute.



ID: b6

Entity: global

Public name: eligible_GFA

Document: Properties.xsrc

Data type: Boolean

Text: the claimant is eligible for Gippsland flood assistance

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 a 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

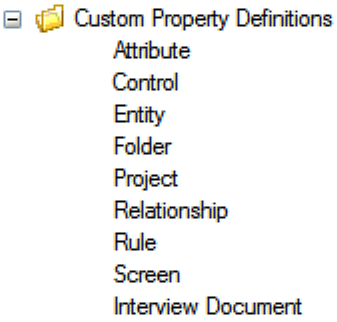
Assign a custom property to a folder
Assign a custom property to the project
Implement a custom property using application support
Generate a report of custom properties in a project

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.

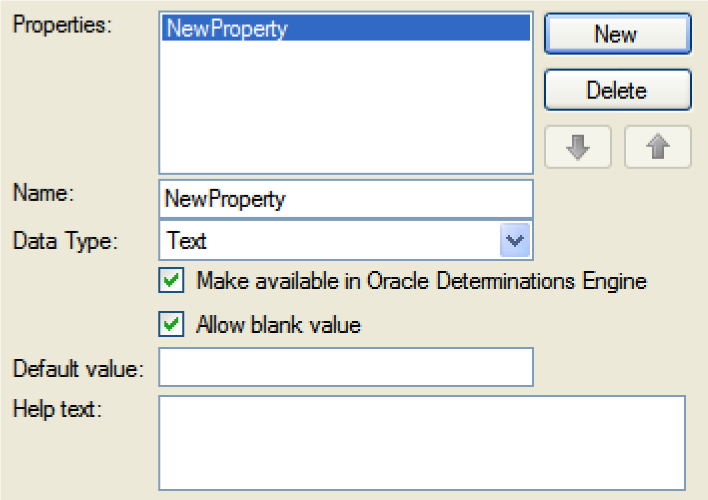
To specify a custom property definition:

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:



Custom Property Definitions
- Attribute
- Control
- Entity
- Folder
- Project
- Relationship
- Rule
- Screen
- Interview Document

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.



Properties: NewProperty

Name: NewProperty

Data Type: Text

☑ Make available in Oracle Determinations Engine

☑ Allow blank value

Default value:

Help text:

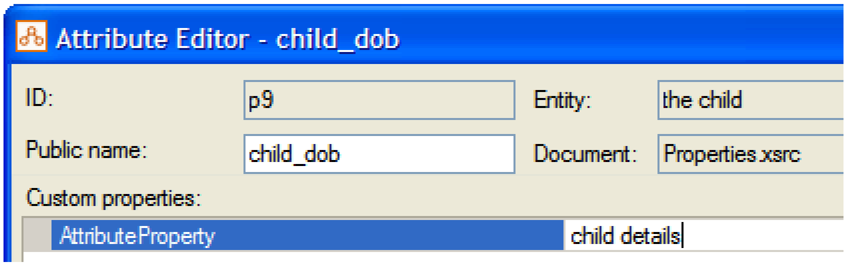
4. Specify a **Name** for the custom property.
5. Select the **Data Type** from the drop-down list (Text, Boolean, Number or List).
6. If you don't want the custom property to be included at runtime, unselect the **Make available in Oracle Determinations Engine** check box.

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.



Attribute Editor - child_dob

ID: p9
Entity: the child
Public name: child_dob
Document: Properties.xsrc
Custom properties:
AttributeProperty child details

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

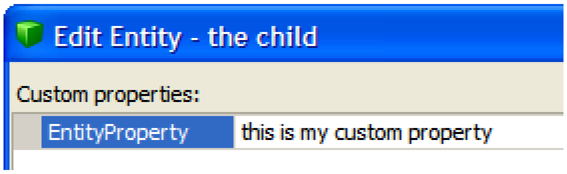
NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work.

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.



Edit Entity - the child

Custom properties:

EntityProperty: this is my custom property

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.

Relationship Editor - the child's school

Source: the child

Target: the school

Custom properties:

RelationshipProperty: this is my custom property

5. Click **OK** to apply the change.

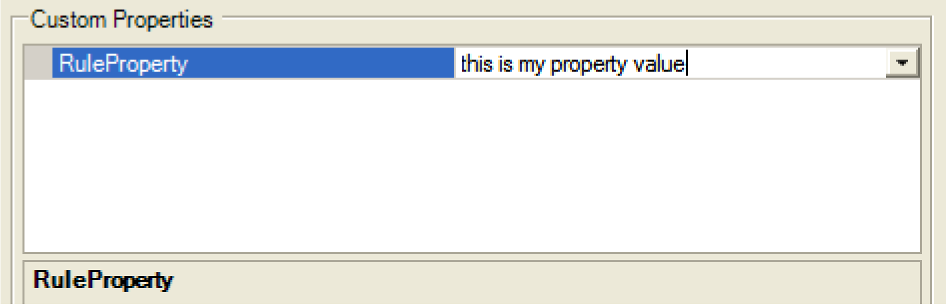
NOTE: Custom property names are case sensitive. If the case does not match the case in the definition then the custom property will not work. Also, in order for a custom property to be compiled out for a relationship, the relationship must have a public name.

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.



this is my property value

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:

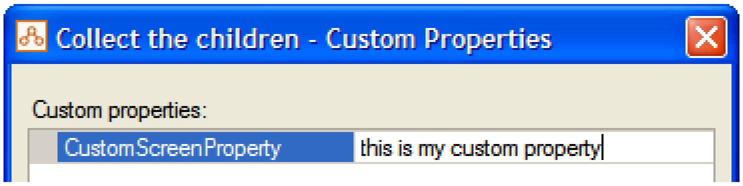
- identifying the owner of rules (eg rule_property[Maintenance:Business Rule Team])
- identifying the date the rule was last amended (eg rule_property[Updated:2006-06-17])

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.



this is my custom property

4. Click **OK** to apply the change and save your screen document.

An example of how custom properties can be used on screens is:

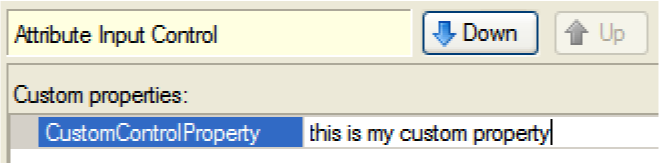
- to make a whole screen display differently. For example, to have a different watermark or size of font from other screens.

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.



Attribute Input Control

Custom properties:

CustomControlProperty: this is my custom property

4. Click **OK** to apply the change and save your screen document.

Some examples of how custom properties can be used on controls are:

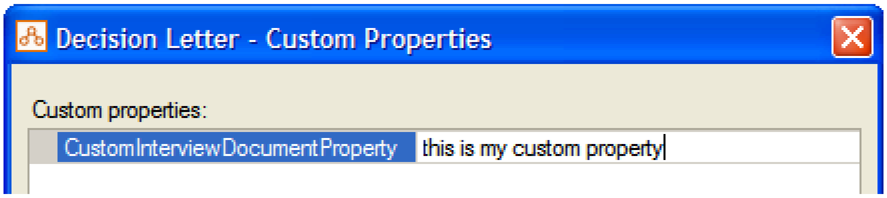
- to control the appearance of inferred attributes on screens
- to make radio buttons appear down the page (rather than the default position of across the page)
- to change commentary based on answers to earlier questions. For example, to substitute the claimant's name in the commentary for a particular 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.



Decision Letter - Custom Properties

Custom properties:

CustomInterviewDocumentProperty: this is my custom property

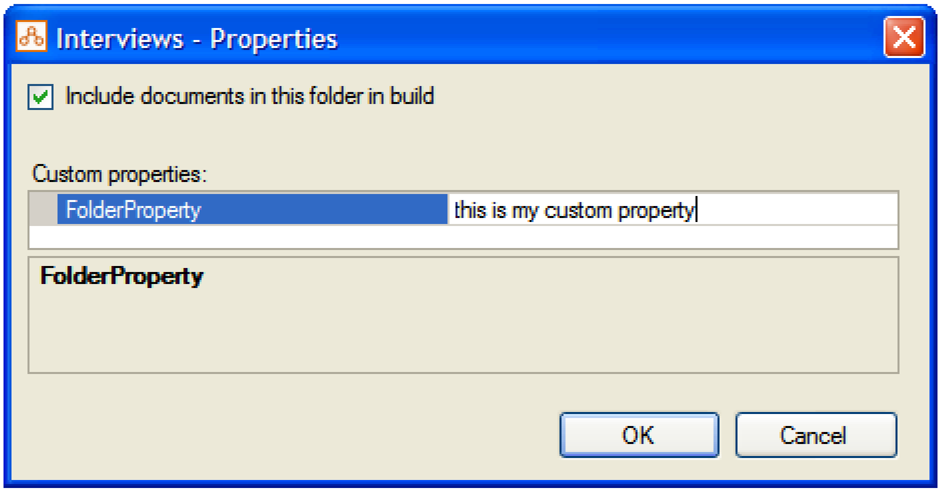
4. Click **OK** to apply the change and save your screen document.

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.



FolderProperty: this is my custom property

FolderProperty

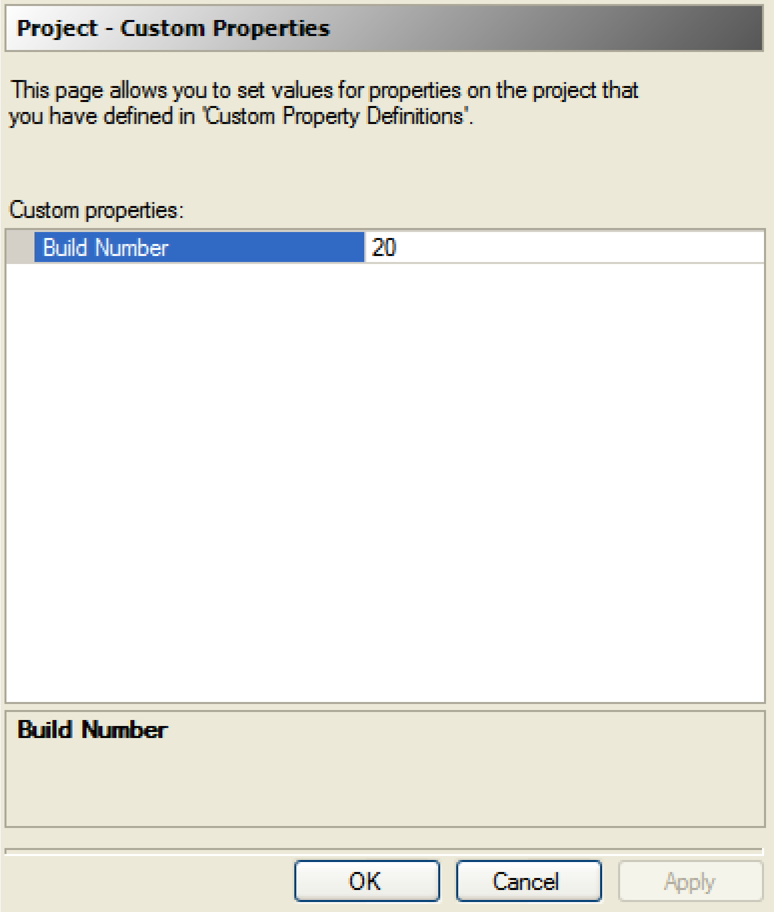
4. Click **OK** to apply the change.

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.



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

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').

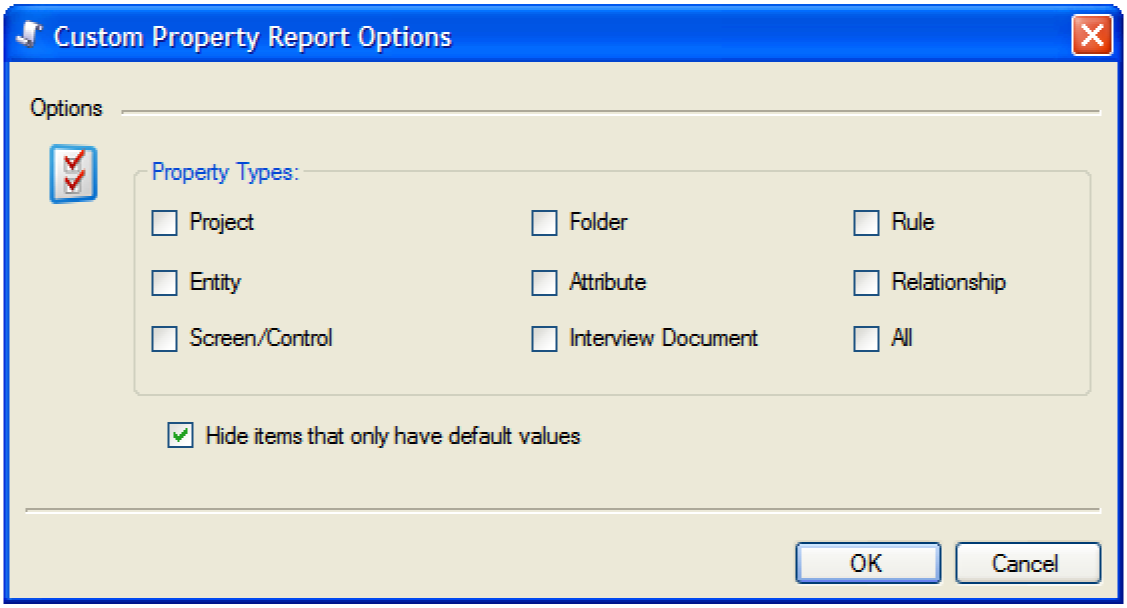
TIP: It does not make sense to use custom properties when rules alone can do the work or when they would require replication of logic in the rulebase. Also, it is not advisable to use custom properties when a change is required globally and there is no likelihood in the future of it being changed again.

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.



Custom Property Report Options

Options

Property Types:

☐ Project ☐ Folder ☐ Rule

☐ Entity ☐ Attribute ☐ Relationship

☐ Screen/Control ☐ Interview Document ☐ All

☑ Hide items that only have default values

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
Build Options	
-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
Validation Options	
-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
Report Options	
-cd	Analyzes a *.coverage file and produces a document-oriented report (.xml)

Parameter	Description
-cg	Analyzes a *.coverage file and produces a goal-oriented report(.xml)
Upgrade Options	
-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 *-upgrade* flag. When set, write-protection will be removed for read-only project files. When not set, read-only project files will still be copied to the upgraded project directory but won't be processed.
Help Options	
-h	Prints the help message
--diagnostics	Generates diagnostic information. The project path and other parameters are ignored.

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.

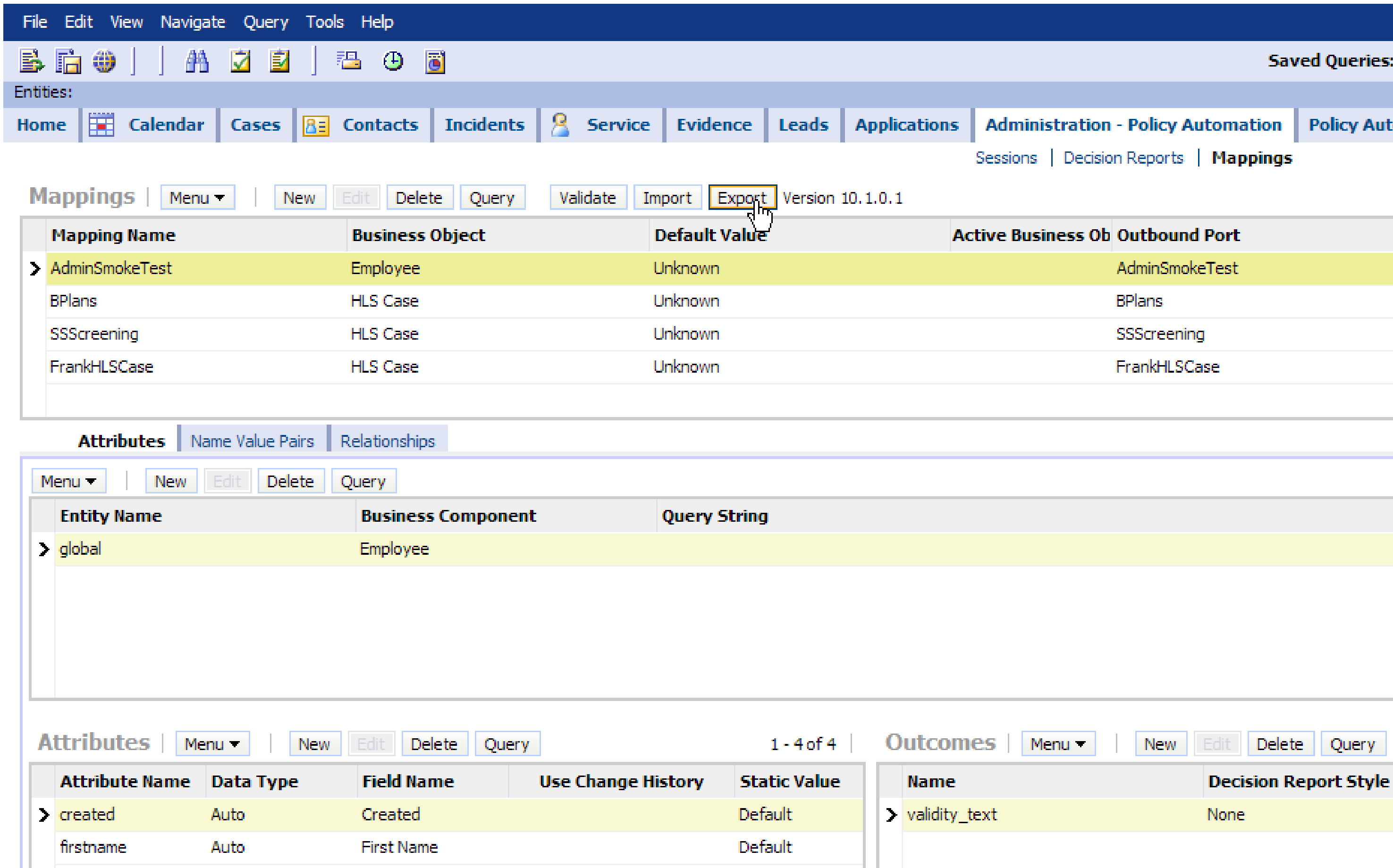
Import a Data Mapping from Siebel

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.



Version 10.1.0.1

Mapping Name	Business Object	Default Value	Active Business Ob	Outbound Port
AdminSmokeTest	Employee	Unknown		AdminSmokeTest
BPlans	HLS Case	Unknown		BPlans
SSScreening	HLS Case	Unknown		SSScreening
FrankHLSCase	HLS Case	Unknown		FrankHLSCase

Entity Name	Business Component	Query String
global	Employee	

Attribute Name	Data Type	Field Name	Use Change History	Static Value
created	Auto	Created		Default
firstname	Auto	First Name		Default

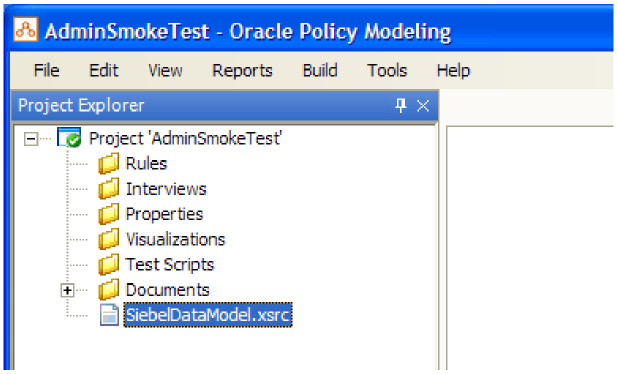
Name	Decision Report Style
validity_text	None

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 .

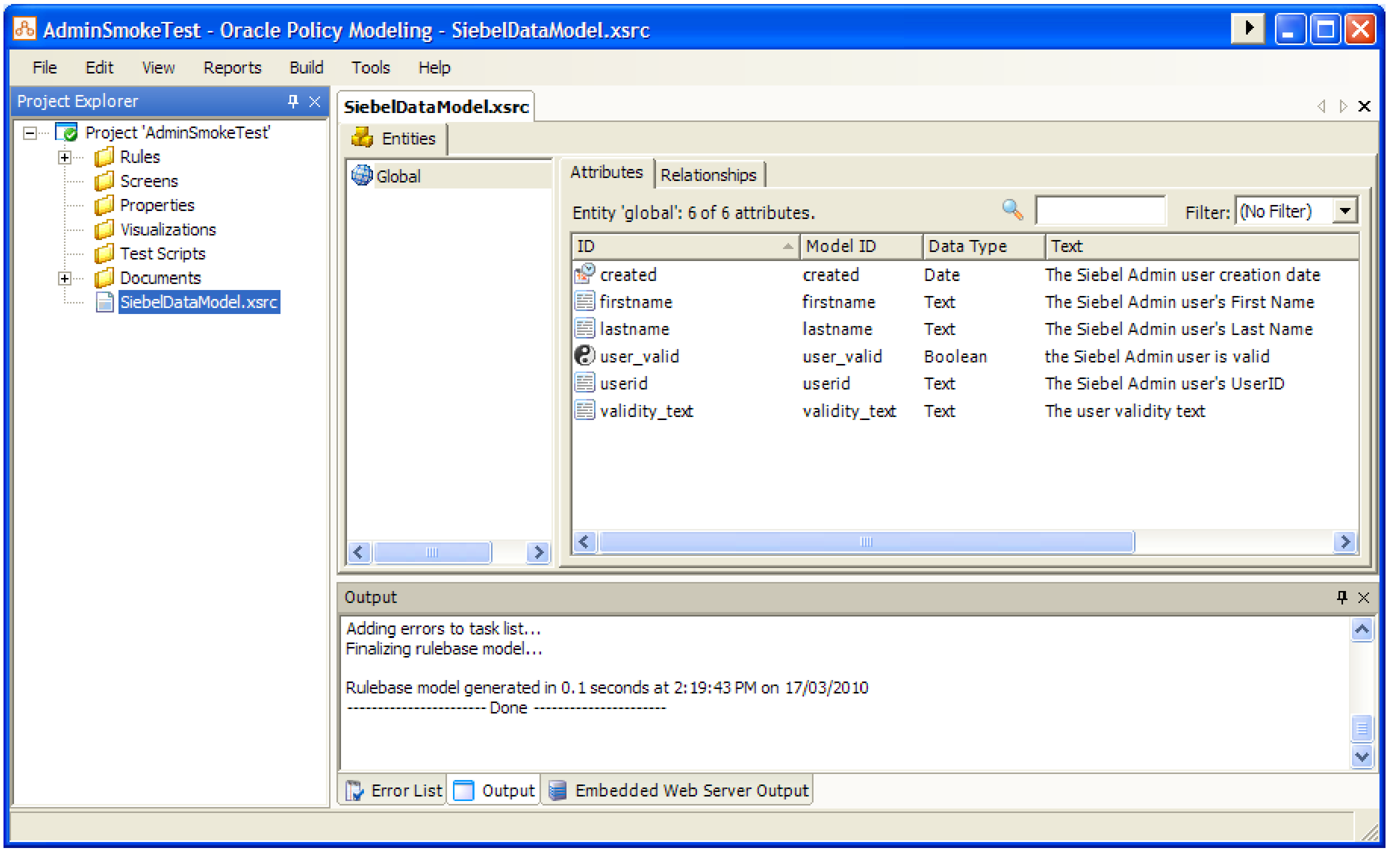
Import the Data Mapping into an Oracle Policy Modeling Project:

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:



Entity 'global': 6 of 6 attributes.

ID	Model ID	Data Type	Text
created	created	Date	The Siebel Admin user creation date
firstname	firstname	Text	The Siebel Admin user's First Name
lastname	lastname	Text	The Siebel Admin user's Last Name
user_valid	user_valid	Boolean	the Siebel Admin user is valid
userid	userid	Text	The Siebel Admin user's UserID
validity_text	validity_text	Text	The user validity text

```
Adding errors to task list...
Finalizing rulebase model...

Rulebase model generated in 0.1 seconds at 2:19:43 PM on 17/03/2010
---------------------- Done ----------------------
```

Accessibility

Topics in "Accessibility"

- Keyboard shortcuts for Oracle Policy Modeling
- Modify the appearance or layout of Oracle Policy Modeling
- Modify the appearance of rules in Word
- 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

Shortcut Key	Function/Navigation
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
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

Shortcut key	Style/Function
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

Shortcut key	Style/Function
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
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

Shortcut key	Style/Function
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

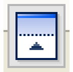
Change the color scheme

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.

Modify the appearance of rules in Word

High-contrast mode is a Windows feature that chooses colors and fonts to maximize the clarity of text and images. When using one of the "dark" high-contrast themes, Policy Modeling rule documents in Microsoft Word become very hard to read.

You can modify the Word template file (*.dotm), however, so that the background colors used for rules are more readable when used with a Windows high-contrast theme. When you open a rule document, Word will use your chosen colors, but when another user opens the same document, they will see the default Policy Modeling colors.

To make this change:

1. Enable "Automatically update document styles" in Word
   This is a per-document setting in Word. It is enabled by default for new rule documents in OPM 10.4 Update 6 (or later) but may have to be changed for files created in a previous version. To change or check the setting for a rule document:
   - Word 2007/2010/2013:
     - Click the Office Button (or the **File** tab), select **Word Options**, then **Add-Ins**.
     - In the **Manage** drop-down, select **Templates** and then click **Go**.
     - On the **Templates** tab, select the option **Automatically update document styles**.
   - Word 2003:
     - In the **Tools** menu, select **Templates and Add-Ins**.
     - On the **Templates** tab, select the option **Automatically update document styles**.
2. Locate the Word templates used by Oracle Policy Automation
   The Microsoft Word templates used by Oracle Policy Automation are located in one of two places:
   - C:\Program Files\Oracle\Policy Modeling\Templates, or
   - C:\Program Files (x86)\Oracle\Policy Modeling\Templates

   Depending on the version of Microsoft Word you have installed, you will have to modify a different template:

- Word 2013 or Word 2010, use the Policy Modeling Word 2010 Template.dotm
- Word 2007, use the Policy Modeling Word 2007 Template.dotm
- Word 2003, use the Policy Modeling Word Template.dot

3. Change the background colors in the Word templates
   To do this:
   a. Right click the Word template and select **Open**.
   b. Open the **Styles** window:
      - For Word 2007/2010/2013, on the **Home** tab, click the arrow button in the bottom right hand corner of the **Styles** group.
      - For Word 2003, in the **Format** menu, select **Styles and Formatting**.
   c. Select the **OPM - Level 1** style, click the down-arrow next to the style and select **Modify**.
   d. In the **Modify Style** window, select **Format**, then **Border**.
   e. In the **Borders and Shading** dialog, select the **Shading** tab. The **Fill** color can be changed as follows:
      - Select **No color** to make the background color black in a dark high-contrast theme, or white otherwise.
      - Select **More colors**, then in the **Colors** dialog, select the **Custom** tab to select a darker version of the same background color (something suitable for the light text color used by most high-contrast themes).
   f. Repeat the above process for the following additional styles:
      - **OPM – configuration**
      - **OPM – level 2**
      - **OPM – level 3**
      - **OPM – level 4**
      - **OPM – level 5**
      - **OPM – level 6**

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 contol, 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
if	Optional term that can appear at the end of a conclusion line that has a following proof
and	Logical conjunction between two attributes
or	Logical disjunction between two attributes
either **one of** **any** **at least one of the following is true**	Grouping element used with disjunctions where two or more attributes need to be grouped

Syntax	Description
any of the following are satisfied	
both **all** **all of the following are true** **all of the following are satisfied**	Grouping element used with conjunctions where two or more attributes need to be grouped
otherwise	Term that appears at the end of a table rule to indicate the otherwise clause
is	Term that is used in a legend entry between the abbreviated phrase and the full attribute text

Logical functions

Syntax	Description
it is not true that *<expr>*	Operator used to return true if attribute has a value which is false
<var> **is certain** **it is certain whether** [**or not**]*<expr>*	Operator used to return true if attribute has a value which is not uncertain
<var> **is uncertain** *<var>* **is not certain** **it is uncertain that** *<expr>* **it is uncertain whether** [**or not**]*<expr>* **it is not certain that** *<expr>*	Operator used to return true if attribute value is uncertain
<var> **is known** *<var>* **is currently known** **it is known whether** [**or not**]*<expr>* **it is currently known whether** [**or not**]*<expr>*	Operator used to return true if attribute has any value
<var> **is** [**currently**] **unknown** **it is** [**currently**] **unknown whether** [**or not**]*<expr>*	Operator used to return true if attribute has no value

Logical constants

Syntax	Description
true	Constant true value used for table rules.
false	Constant false value used for table rules.
uncertain	Constant uncertain value used for table rules.

Comparison operators

Syntax	Description
*<x>***<***<y>* *<x>* **is earlier than** *<y>*	Less than ***Note:*** there is no natural language form when this operator is used with numerical and currency values.
<x> **>** *<y>* *<x>* **is later than** *<y>*	Greater than ***Note:*** there is no natural language form when this operator is used with numerical and currency values.
*<x>***<=***<y>* *<x>* **is less than or equal to** *<y>* *<x>* **is on or earlier than** *<y>* *<x>* **is at or earlier than** *<y>*	Less than or equal to
<x> **>=** *<y>* *<x>* **is greater than or equal to** *<y>* *<x>* **is on or later than** *<y>* *<x>* **is at or later than** *<y>*	Greater than or equal to
*<x>***=***<y>* *<x>* **is equal to** *<y>* *<x>* **equals** *<y>*	Equals
<x> **is not equal to** *<y>* *<x>* **<>** *<y>*	Not equal

Numerical functions

Syntax	Description
Number(*<numText>*)	Convert the specified string into a ***number*** value
<x> **+** *<y>*	Mathematical addition
<x> **-** *<y>*	Mathematical subtraction
<x> * *<y>*	Mathematical multiplication
<x> **/** *<y>*	Mathematical division
<x> **** *<y>*	Integer division
<x> **modulo** *<y>*	Remainder after integer division
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **the greater of** *<x>* **and** *<y>* **the latest of** *<x>* **and** *<y>*	Returns the greater of two values

Syntax	Description
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **the lesser of** *<x>* **and** *<y>* **the earliest of** *<x>* **and** *<y>*	Returns the lesser of two values
Xy(*<x>*, *<y>*) *<x>* **raised to the power of** *<y>*	x to the power of y
Ex(*<x>*) **e to the power of** *<x>*	Constant e to the power of x
Abs(*<x>*) **the absolute value of** *<x>* **|***<x>***|**	Absolute value of x
Ln(*<x>*) **the natural logarithm of** *<x>*	Natural logarithm of x
Log(*<x>*) **the logarithm base 10 of** *<x>*	Logarithm base 10 of x
Sqrt(*<x>*) **the square root of** *<x>*	Square root of x
Round(*<x>*, *<n>*) *<x>* **rounded to** *<n>* **decimal place** *<x>* **rounded to** *<n>* **decimal places**	Rounds x to n decimal places
Trunc(*<x>*, *<n>*) *<x>* **truncated to** *<n>* **decimal place** *<x>* **truncated to** *<n>* **decimal places**	x truncated to n decimal places
Sin(*<x>*)	Sine of x
Cos(*<x>*)	Cosine of x
Tan(*<x>*)	Tangent of x
Asin(*<x>*)	Arcsine of x
Acos(*<x>*)	Arccosine of x
Atan(*<x>*)	Arctangent of x

Date functions

Syntax	Description
CurrentDate()	Returns the current date at the start of the session.

Syntax	Description
the current date	
Date(*<text>*)	Converts the specified string into a date value
MakeDate(*<year>*, *<month>*, *<day>*)	Returns a date formed from the specified year, month, and day.
ExtractDay(*<date/datetime>*)	Returns the day component of a date/datetime attribute.
ExtractMonth(*<date/datetime>*)	Returns the month component of a date/datetime attribute.
ExtractYear(*<date/datetime>*)	Returns the year component of a date/datetime attribute.
NextDayOfTheWeek(*<date/datetime>*, *<day>*) **the next Monday on or after** *<from-date>* **the Monday on or before** *<from-date>* **the next Tuesday on or after** *<from-date>* **the Tuesday on or before** *<from-date>* **the next Wednesday on or after** *<from-date>* **the Wednesday on or before** *<from-date>* **the next Thursday on or after** *<from-date>* **the Thursday on or before** *<from-date>* **the next Friday on or after** *<from-date>* **the Friday on or before** *<from-date>* **the next Saturday on or after** *<from-date>* **the Saturday on or before** *<from-date>* **the next Sunday on or after** *<from-date>* **the Sunday on or before** *<from-date>*	Returns the date of the next weekday on or before/after a date (depending on the syntax used).
NextDate(*<date>*, *<day>*,	Returns the next instance of the given day and month after a date.

Syntax	Description
<month>) **the previous UK tax year start date on or before** *<from-date>* **the next UK tax year end date on or after** *<from-date>*	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.
AddDays(*<date/datetime>*, *<num_days>*) **the date** *<num_days>* **days after** *<datetime>* **the date** *<num_days>* **days before** *<datetime>* **the date** *<num_days>* **day after** *<datetime>* **the date** *<num_days>* **day before** *<datetime>* **the time** *<num_days>* **days after** *<datetime>* **the time** *<num_days>* **days before** *<datetime>* **the time** *<num_days>* **day after** *<datetime>* **the time** *<num_days>* **day before** *<datetime>*	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.
AddWeeks(*<date/datetime>*, *<num_weeks>*) **the date** *<num_weeks>* **weeks after** *<datetime>* **the date** *<num_weeks>* **weeks before** *<datetime>* **the date** *<num_weeks>* **week after** *<datetime>* **the date** *<num_weeks>* **week before** *<datetime>* **the time** *<num_weeks>* **weeks after** *<datetime>* **the time** *<num_weeks>* **weeks before** *<datetime>* **the time** *<num_weeks>* **week after** *<datetime>* **the time** *<num_weeks>* **week before** *<datetime>*	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.

Syntax	Description
AddMonths(*<date/datetime>*, *<num_months>*) **the date** *<num_months>* **months after** *<datetime>* **the date** *<num_months>* **months before** *<datetime>* **the date** *<num_months>* **month after** *<datetime>* **the date** *<num_months>* **month before** *<datetime>* **the time** *<num_months>* **months after** *<datetime>* **the time** *<num_months>* **months before** *<datetime>* **the time** *<num_months>* **month after** *<datetime>* **the time** *<num_months>* **month before** *<datetime>*	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.
AddYears(*<date/datetime>*, *<num_years>*) **the date** *<num_years>* **years after** *<datetime>* **the date** *<num_years>* **years before** *<datetime>* **the date** *<num_years>* **year after** *<datetime>* **the date** *<num_years>* **year before** *<datetime>* **the time** *<num_years>* **years after** *<datetime>* **the time** *<num_years>* **years before** *<datetime>* **the time** *<num_years>* **year after** *<datetime>* **the time** *<num_years>* **year before** *<datetime>*	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.
WeekdayCount(*<date1>*, *<date2>*) **the number of weekdays (inclusive) between** *<date1>* **and** *<date2>*	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.
YearStart(*<date/datetime>*)	Returns the first date in the year in which a date falls.

Syntax	Description
the first day of the year in which *<from-date>* **falls**	
YearEnd(*<date/datetime>*) **the last day of the year in which** *<from-date>* **falls**	Returns the last date in the year in which a date falls.
DayDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of days from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole days between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
DayDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of days (inclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole days (inclusive) between date/datetime1 and date/datetime2. 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.
DayDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of days (exclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole days (exclusive) between date/datetime1 and date/datetime2. 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.
WeekDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of weeks from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole elapsed weeks between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
WeekDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of weeks (inclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the inclusive number of whole elapsed weeks between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
WeekDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of weeks (exclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the exclusive number of whole elapsed weeks between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.

Syntax	Description
MonthDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of months from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole elapsed months between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
MonthDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of months (inclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole inclusive elapsed months between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
MonthDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of months (exclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole exclusive elapsed months between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
YearDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of whole years which** *<date/datetime2>* **is after** *<date/datetime1>* **the number of years between** *<date/datetime1>* **and** *<date/datetime2>*	Returns the number of years between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
YearDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of years (inclusive) between** *<date/datetime1>* **and** *<date/datetime2>*	Returns the inclusive number of years between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.
YearDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of years (exclusive) between** *<date/datetime1>* **and** *<date/datetime2>*	Returns the exclusive number of years between date/datetime1 and date/datetime2. The order of the two dates does not affect the result.

Time of day functions

Syntax	Description
TimeOfDay(*<text>*)	Converts the given string into a time of day
ExtractSecond(*<time/datetime>*)	Returns the second component of a timeofday/datetime attribute.
ExtractMinute(*<time/datetime>*)	Returns the minute component of a timeofday/datetime attribute.
ExtractHour(*<time/datetime>*)	Returns the hour component of a timeofday/datetime attribute.

Date and time functions

Syntax	Description
CurrentDateTime() **the current date time**	Returns the current date and time at the start of the session.
DateTime(*<text>*)	Converts the specified string into a datetime value
ConcatenateDateTime(*<date>*, *<time>*) *<date>* **at** *<time-of-day>* *<time-of-day>* **on** *<date>*	Sets the date time by joining the date and time of day together.
SecondDifference(*<datetime1>*, *<datetime2>*) **SecondDifference** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds from** *<datetime1>* **to** *<datetime2>*	Returns the number of seconds between datetime1 and datetime2.
SecondDifferenceInclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds (inclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the inclusive number of seconds between datetime1 and datetime2.
SecondDifferenceExclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds (exclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the exclusive number of seconds between datetime1 and datetime2.
MinuteDifference(*<datetime1>*, *<datetime2>*)	Returns the number of minutes between datetime1 and datetime2.

Syntax	Description
MinuteDifference (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes from** *<datetime1>* **to** *<datetime2>*	
MinuteDifferenceInclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes (inclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the inclusive number of minutes between datetime1 and datetime2.
MinuteDifferenceExclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes (exclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the exclusive number of minutes between datetime1 and datetime2.
HourDifference(*<datetime1>*, *<datetime2>*) **HourDifference**(*<timeOfDay1>*, *<timeOfDay2>*) **the number of hours from** *<datetime1>* **to** *<datetime2>*	Returns the number of hours between datetime1 and datetime2.
HourDifferenceInclusive(*<datetime1>*, *<datetime2>*) **HourDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of hours (inclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the inclusive number of hours between datetime1 and datetime2.
HourDifferenceExclusive(*<datetime1>*, *<datetime2>*) **HourDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of hours (exclusive) from** *<datetime1>* **to** *<datetime2>*	Returns the exclusive number of hours between datetime1 and datetime2.
ExtractDate(*<datetime>*)	Extracts the date from a datetime attribute.
ExtractTimeOfDay(*<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.
AddHours(*<datetime>*, *<num_hours>*) **AddHours**(*<timeOfDay>*, *<num_hours>*) **the time** *<num_hours>* **hours after** *<datetime>* **the time** *<num_hours>* **hours before** *<datetime>* **the time** *<num_hours>* **hour after** *<datetime>* **the time** *<num_hours>* **hour before** *<datetime>*	Adds a number of hours to a date time.
AddMinutes(*<datetime>*, *<num_minutes>*) **AddMinutes**(*<timeOfDay>*, *<num_minutes>*) **the time** *<num_minutes>* **minutes after** *<datetime>* **the time** *<num_minutes>* **minutes before** *<datetime>* **the time** *<num_minutes>* **minute after** *<datetime>* **the time** *<num_minutes>* **minute before** *<datetime>*	Adds a number of minutes to a date time.
AddSeconds(*<datetime>*, *<num_seconds>*) **AddSeconds**(*<timeOfDay>*, *<num_seconds>*) **the time** *<num_seconds>* **seconds after** *<datetime>* **the time** *<num_seconds>* **seconds before** *<datetime>* **the time** *<num_seconds>* **second after** *<datetime>* **the time** *<num_seconds>* **second before** *<datetime>*	Adds a number of seconds to a date time.

Text functions

Syntax	Description
<text1> **&** *<text2>*	Combines text1 with text2 and so on to form a single text value. ***Note:*** that you can use variables of any type. Values are formatted using the formatter that is installed in the rule session.
the concatenation of *<text1>* **&** *<text2>*	Combines text1 with text2 and so on to form a single text value. ***Note:*** that you can use variables of any type. Values are formatted using the formatter that is installed in the rule session.
Contains(*<text>*, *<sub-string>*) *<text>* **contains** *<sub-string>*	Returns a boolean value indicating whether the given text value contains the given text sub-string. The text comparison is case-insensitive.
EndsWith(*<text>*, *<sub-string>*) *<text>* **ends with** *<sub-string>*	Returns a boolean value indicating whether the given text value ends with the given text sub-string. The text comparison is case-insensitive.
IsNumber(*<text>*) *<text>* **is a number**	Returns a boolean value indicating whether the given text value represents a valid number.
Length(*<text>*) **the length of** *<text>*	Returns the character length of the given text value.
StartsWith(*<text>*, *<substring>*) *<text>* **starts with** *<sub-string>*	Returns a boolean value indicating whether the given text value starts with the given text sub-string. The text comparison is case-insensitive.
Substring(*<text>*, *<off-set>*, *<length>*)	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.
Text(*<number>*) **Text**(*<date>*) **Text**(*<datetime>*) **Text**(*<timeOfDay>*)	Convert the specified number or date attribute into a text value.

Entity and relationship functions

Syntax	Description
For(*<relationship>*, *<Exp>*) **in the case of** *<relationship>***,** *<attr>* *<val>***, in the case of** *<relationship>*	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.
ForScope(*<relationship>*, *<alias>*) **ForScope**(*<relationship>*)	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
in the case of *<relationship>* **in the case of** *<relationship>* **(**<alias>**)**	
ForAll(*<relationship>*, *<Exp>*) **each of** *<relationship-attr>* **for each of** *<relationship>***,** *<attr>* **for all of** *<relationship>***,** *<attr>*	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. This form is used when there is only one condition in the rule.
ForAllScope(*<relationship>*) **ForAllScope**(*<relationship>*, *<alias>*) **for all of** *<relationship>* **each of** *<relationship>* **for each of** *<relationship>* **for all of** *<relationship>* **(**<alias>**)** **each of** *<relationship>* **(**<alias>**)** **for each of** *<relationship>* **(**<alias>**)**	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. This form is used when there are one or more conditions in the rule.
Exists(*<relationship>*, *<Exp>*) **at least one of** *<relationship-attr>* **for at least one of** *<relationship>***,** *<attr>*	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. This form is used when there is only one condition in the rule.
ExistsScope(*<relationship>*) **ExistsScope**(*<relationship>*, *<alias>*) **at least one of** *<relationship>* **for at least one of** *<relationship>* **at least one of** *<relationship>* **(**<alias>**)** **for at least one of** *<relationship>* **(**<alias>**)**	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. This form is used when there are one or more conditions in the rule.
IsMemberOf(*<target>*, *<relationship>*) **IsMemberOf**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **is a member of** *<relationship>* <ent-target> **(**<alias>**) is a member of** *<relationship>*	Used as a conclusion to infer that an entity instance is a member of a relationship. 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.
IsNotMemberOf(*<target>*, *<relationship>*) <ent-target> **is not a member of** *<relationship>*	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.
InstanceCount(*<relationship>*) **the number of** *<relationship>*	Counts the number of instances that exist for an entity.
InstanceCountIf(*<relationship>*,	Counts the number of instances there are of an entity for which a particular entity-

Syntax	Description
<Exp>) **the number of** *<relationship>* **for which it is the case that** *<condition>*	level attribute has a particular value.
InstanceMaximum(*<relationship>*, *<number-attr>*) **InstanceMaximum**(*<relationship>*, *<date-attr>*) **InstanceMaximum**(*<relationship>*, *<datetime-attr>*) **InstanceMaximum**(*<relationship>*, *<time-attr>*) *<date-attr>* **which is the latest for all** [**of**]*<relationship>* *<max-attr>* **which is the greatest for all** [**of**]*<relationship>* **the latest of all** *<relationship-attr>* **the latest of all** *<attr>* **for** *<relationship>* **the greatest of** [**all**]*<relationship-attr>* **the greatest of** [**all**]*<attr>* **for** [**all**] [**of**]*<relationship>*	Obtains the highest/most recent value of an entity-level variable for all instances of the entity.
InstanceMaximumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **which is the latest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<max-attr>* **which is the greatest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the latest of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the greatest of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the greatest of** *<attr>* **for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>*	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.

Syntax	Description
InstanceMinimum(*<relationship>*, *<number-attr>*) **InstanceMinimum**(*<relationship>*, *<date-attr>*) **InstanceMinimum**(*<relationship>*, *<datetime-attr>*) **InstanceMinimum**(*<relationship>*, *<time-attr>*) *<date-attr>* **which is the earliest for all** [**of**]*<relationship>* *<attr>* **which is the least for all** [**of**]*<relationship>* **the earliest of all** *<relationship-attr>* **the earliest of all** *<attr>* **for** *<relationship>* **the least of** [**all**]*<relationship-attr>* **the least of** [**all**]*<attr>* **for** [**all**] [**of**]*<relationship>*	Obtains the lowest/least recent value of an entity-level variable for all instances of the entity.
InstanceMinimumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **which is the earliest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<num-attr>* **which is the least for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the least of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the least of all** *<attr>* **for** *<relationship>* **for which it is the case that** *<ent-test>* **the earliest of all** *<attr>* **for** *<relationship>* **for which it is the case that** *<ent-test>*	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.
InstanceSum(*<relationship>*, *<number-attr>*) *<num-attr>*(**totaled** | **totalled**) **for all**	Obtains the sum of all instances of an entity-level variable.

Syntax	Description
[**of**]*<relationship>* **the total amount of** [**all**]*<relationship-attr>* **the total for all** *<relationship-attr>* **total for all** *<relationship>***,** *<attr>*	
InstanceSumIf(*<relationship>*, *<number-attr>*, *<condition>*) *<num-attr>* **totalled for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<num-attr>* **totaled for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the total amount of all** *<relationship-attr>* **only where** *<condition>* **the total amount of** [**all**]*<relationship-attr>* **for which it is the case that** *<condition>* **total for all** *<relationship>***,** *<attr>* **only where** *<condition>*	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.
InstanceValueIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<time-attr>*, *<condition>*)	Obtains 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 the value is Uncertain.
InstanceEquals(*<instance1>*, *<instance2>*) <ent-target> **is** <ent-target>	Determines if two instances of an entity are the same instance.
InstanceNotEquals(*<instance1>*, *<instance2>*) <ent-target> **is not** <ent-target>	Determines if two instances of an entity are not the same instance.
InferInstance(*<relationship>*, *<identity>*) *<rel>***(***<identity>***) exists**	Used as a conclusion to infer that an entity instance exists and is a member of a relationship.

Temporal reasoning functions

Syntax	Description
IntervalCountDistinct (*<start-date>*, *<end-date>*, *<variable>*) **IntervalCountDistinct** (*<start-date>*, *<end-date>*, *<condition>*)	Counts the number of known distinct values for the variable, in the interval from the start date (inclusive) to the end date (exclusive).
IntervalCountDistinctIf (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	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.
IntervalDailySum(*<start-date>*, *<end-date>*, *<number-attr>*)	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.
IntervalDailySumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	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.
IntervalMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Selects the maximum value of a variable in the interval from a start date (inclusive) to an end date (exclusive).
IntervalMaximumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	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.
IntervalMinimum(*<start-*	Selects the minimum value of a variable in the interval from a start date (inclusive) to an end

Syntax	Description
date>, *<end-date>*, *<number-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	date (exclusive).
IntervalMinimumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	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.
IntervalWeightedAverage (*<start-date>*, *<end-date>*, *<number-attribute>*)	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.
IntervalWeightedAverageIf (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	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).
IntervalAlways(*<start-date>*, *<end-date>*, *<condition>*)	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).
IntervalAtLeastDays(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	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).
IntervalConsecutiveDays (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	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).
IntervalSometimes(*<start-*	Returns true if and only if a boolean condition is ever true in the interval from the start date

Syntax	Description
date>, *<end-date>*, *<condition>*)	(inclusive) to the end date (exclusive).
ValueAt(*<date>*, *<value>*)	Returns the value of the given attribute at the specified date.
WhenLast(*<date>*, *<condition>*)	Returns the date on which a boolean condition was last true, looking backwards from (and including) a specified date.
WhenNext(*<date>*, *<condition>*)	Returns the date on which a boolean condition will next be true, looking forwards from (and including) a specified date.
Latest()	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.
Earliest()	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.
TemporalDaysSince(*<date>*, *<end-date>*)	Returns a number variable that varies every day and is the number of full days since the date.
TemporalWeeksSince (*<date>*, *<end-date>*)	Returns a number variable that varies every week and is the number of full weeks since the date.
TemporalMonthsSince (*<date>*, *<end-date>*)	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.
TemporalYearsSince (*<date>*, *<end-date>*)	Returns a number variable that varies every year and is the number of full years since the date.
TemporalAlwaysDays (*<days>*, *<condition>*)	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.
TemporalConsecutiveDays (*<minDays>*, *<days>*, *<condition>*)	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.
TemporalSometimesDays (*<days>*, *<condition>*)	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.
TemporalAfter(*<date>*)	Returns a boolean attribute that varies over time and is true after a date and false on and before.
TemporalBefore(*<date>*)	Returns a boolean attribute that varies over time and is true before a date and false on and afterwards.
TemporalOn(*<date>*)	Returns a boolean attribute that varies over time and is true on a date and false before and afterwards.

Syntax	Description
TemporalOnOrAfter (*<date>*)	Returns a boolean attribute that varies over time and is true on or after a date and false before.
TemporalOnOrBefore (*<date>*)	Returns a boolean attribute that varies over time and is true on and before a date and false afterwards.
TemporalFromStartDate (*<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.
TemporalFromEndDate (*<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.
TemporalFromRange(*<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.
TemporalIsWeekday (*<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.
TemporalOncePerMonth (*<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
Error (*<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.
Warning (*<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
CallCustomFunction (*<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

الروابط المنطقية(English)

الصياغة	الوصف
إذا **اذا**	شرط اختياري قد يظهر في نهاية سطر استنتاج له برهان تالٍ
و	الربط المنطقي بين قيمتين من ***attributes***
أو **او**	أداة إلغاء ربط منطقي بين اثنتين من ***attributes***

الوصف	الصياغة
عنصر تجميع مُستخدم مع أدوات إلغاء الربط، عندما يلزم تجميع قيمتي ***attributes*** أو أكثر	**احد الامرين** **إما** **واحد من** **أي** **واحد على الأقل مما يلي صحيح** **أيٍ مما يلي راضٍ**
عنصر تجميع مُستخدم مع أدوات الربط، عندما يلزم تجميع قيمتي ***attributes*** أو أكثر	**الاثنان معا** **كل** **كلٌ من** **الكل** **جميع ما يلي صحيح** **جميع ما يلي راضٍ**
شرط يظهر في نهاية قاعدة جدول للإشارة إلى عبارة 'غير ذلك'	**غير ذلك**
شرط يُستخدم في إدخال وسيلة إيضاح بين العبارة المختصرة و ***attribute text*** الكامل	**عبارة عن**

الدوال المنطقية(English)

الوصف	الصياغة
معامل يُستخدم لإرجاع القيمة صحيح إذا كانت قيمة ***attribute*** هي خطأ	**من غير الصحيح أن** *<expr>*
معامل يُستخدم لإرجاع القيمة صحيح إذا كانت قيمة ***attribute*** لا تساوي ***uncertain***	*<var>* **مؤكد** **من المؤكد ما إذا** *<expr>*
معامل يُستخدم لإرجاع القيمة صحيح إذا كانت قيمة ***attribute*** تساوي ***uncertain***	*<var>* **غير مؤكد** **من غير المؤكد أن** *<expr>* **من غير المؤكد ما إذا** *<expr>* **من غير المؤكد أن** *<expr>* **غير مؤكد**
معامل يُستخدم لإرجاع القيمة صحيح إذا كان هناك أية قيمة للسمة ***attribute***	*<var>* **معروف** **من المعروف ما إذا** *<expr>*
معامل يُستخدم لإرجاع القيمة صحيح إذا كانت ***attribute*** بدون أية قيمة.	*<var>* **غير معروف** **من غير المعروف ما إذا** *<expr>* **غير معروف**

الثوابت المنطقية(English)

الوصف	الصياغة
قيمة الثابت صحيح المستخدمة لقواعد الجدول.	**صحيح**
قيمة الثابت خطأ المستخدمة لقواعد الجدول.	**خطأ**

الوصف	الصياغة
قيمة الثابت ***uncertain*** المستخدمة لقواعد الجدول.	**غير مؤكد**

معاملات المقارنة(English)

الوصف	الصياغة
أقل من ملحوظة: لا يوجد نموذج لغة طبيعية عندما يتم استخدام هذا المعامل مع القيم الرقمية وقيم العملات.	*<x>***<***<y>*
أكبر من ملحوظة: لا يوجد نموذج لغة طبيعية عندما يتم استخدام هذا المعامل مع القيم الرقمية وقيم العملات.	*<x>* **>** *<y>*
أقل من أو يساوي	*<x>***<=***<y>*
أكبر من أو يساوي	*<x>* **>=** *<y>*
يساوي	*<x>***=***<y>*
لا يساوي	*<x>* **<>** *<y>*

الدوال الرقمية(English)

الوصف	الصياغة
تحويل السلسلة المحددة إلى قيمة رقمية	**عدد**(*<numText>*)
وظيفة الجمع في الرياضيات	*<x>* **+** *<y>*
وظيفة الطرح في الرياضيات	*<x>* **-** *<y>*
وظيفة الضرب في الرياضيات	*<x>* * *<y>*
وظيفة القسمة في الرياضيات	*<x>* **/** *<y>*
قسمة عدد صحيح	*<x>* **** *<y>*
المتبقي بعد قسمة العدد الصحيح	*<x>* **modulo** *<y>*
إرجاع القيمة الأكبر من قيمتين	**الحد الأقصى**(*<x>*, *<y>*) **الحد الأقصى**(*<date/time/datetime1>*, *<date/time/datetime2>*)
إرجاع القيمة الأقل من قيمتين	**الحد الأدنى**(*<x>*, *<y>*) **الحد الأدنى**(*<date/time/datetime1>*, *<date/time/datetime2>*)
x مرفوعة إلى قوة y	**Xy**(*<x>*, *<y>*)
قيمة الثابت e مرفوعة إلى قوة X	**Ex**(*<x>*)
القيمة المطلقة لـ x	**القيمة المطلقة**(*<x>*) **|***<val>***|**

الوصف	الصياغة
اللوغاريتم الطبيعي لـ x	**لوغاريتم طبيعي**(*<x>*)
القاعدة اللوغاريتمية 10 لـ x	**لوغاريتم**(*<x>*)
الجذر التربيعي لـ x	**جذر تربيعي**(*<x>*)
تقريب x إلى n من الخانات العشرية	**تقريب**(*<x>*, *<n>*)
قيمة x مقتطعة إلى n من الخانات العشرية	**اقتطاع**(*<x>*, *<n>*)
جيب الزاوية لـ x	**جيب الزاوية**(*<x>*)
جيب تمام x	**جيب التمام**(*<x>*)
ظل زاوية لـ x	**ظل الزاوية**(*<x>*)
الدالة العكسية لدالة جيب x	**الدالة العكسية لدالة الجيب**(*<x>*)
الدالة العكسية لجيب تمام x	**الدالة العكسية لجيب التمام**(*<x>*)
الدالة العكسية لدالة ظل x	**الدالة العكسية لدالة الظل**(*<x>*)

دوال التاريخ(English)

الوصف	الصياغة
إرجاع ***date*** الحالي عند بدء الجلسة.	**التاريخ الحالي**()
تحويل السلسلة المحددة إلى قيمة ***date***	**التاريخ**(*<text>*)
إرجاع ***date*** مكون من السنة والشهر واليوم المحددين.	**تكوين تاريخ**(*<year>*, *<month>*, *<day>*)
إرجاع مكون اليوم من ***date/datetime attribute***.	**استخراج اليوم**(*<date/datetime>*)
إرجاع مكون الشهر من ***date/datetime attribute***.	**استخراج الشهر**(*<date/datetime>*)
إرجاع مكون السنة من ***date/datetime attribute***.	**استخراج السنة**(*<date/datetime>*)
إرجاع ***date*** يوم الأسبوع التالي في ***date*** المستخدمة (على أساس الصياغة أو قبله/بعده).	**اليوم التالي من الأسبوع**(*<date/datetime>*, *<day>*)
إرجاع المثيل التالي لليوم والشهر المدخل بعد ***date***.	**التاريخ التالي**(*<date>*, *<day>*, *<month>*)
إضافة/طرح عدد من الأيام إلى/من ***date*** المدخلة. في حالة استخدام نموذج لغوي مختصر، يجب أن يكون العدد عبارة عن عدد صحيح موجب حتى تتم إضافة الأيام إلى قيمة ***date*** المدخلة أو عدد سالب لكي يتم طرح الأيام من قيمة ***date*** المدخلة.	**إضافة أيام**(*<date/datetime>*, *<num_days>*)

الوصف	الصياغة
في حالة استخدام نموذج لغوي مختصر، يجب أن يكون الرقم عبارة عن ***date***. إضافة عدد من الأسابيع إلى المدخلة ***date*** عدد صحيح موجب حتى تتم إضافة الأسابيع إلى قيمة.	**إضافة أسابيع**(*<date/datetime>*, *<num_weeks>*)
في حالة استخدام نموذج لغوي مختصر، يجب أن يكون الرقم عبارة عن ***date***. إضافة عدد من الشهور إلى المدخلة ***date*** عدد صحيح موجب حتى تتم إضافة الشهور إلى قيمة.	**إضافة شهور**(*<date/datetime>*, *<num_months>*)
في حالة استخدام نموذج لغوي مختصر، يجب أن يكون الرقم عبارة عن ***date***. إضافة عدد من السنوات إلى المدخلة ***date*** عدد صحيح موجب حتى تتم إضافة السنوات إلى قيمة.	**إضافة سنوات**(*<date/datetime>*, *<num_years>*)
؛ أي عدد الأيام الواقع بين يومي الاثنين والجمعة.***date***2 و***date***1 حساب عدد أيام الأسبوع بين مستبعد ***date*** مُضمن والتاريخ اللاحق ***date*** ملاحظة: التاريخ السابق.	**عدد أيام الأسبوع**(*<date1>*, *<date2>*)
إرجاع أول قيمة ***date*** من السنة تقع فيها قيمة ***date***.	**بداية العام**(*<date/datetime>*)
إرجاع آخر قيمة ***date*** من السنة تقع فيها قيمة ***date***.	**نهاية العام**(*<date/datetime>*)
إرجاع عدد الأيام الكاملة بين ***date/datetime1*** و***date/datetime2***. لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالأيام**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد الأيام الكاملة بين ***date/datetime1*** و***date/datetime2*** (بما في ذلك هذين التاريخين). يتم تضمين نقطتي النهاية في الاحتساب. إذا كانت قيمتا التاريخ متماثلتين، تكون النتيجة 1. لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالأيام-تضمين**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد الأيام الكاملة بين ***date/datetime1*** و***date/datetime2*** (مع استبعاد هذين التاريخين). يتم استبعاد نقطتي النهاية من الاحتساب. إذا كانت قيمتا التاريخ متماثلتين، تكون النتيجة هي صفر. لا يؤثر ترتيب التواريخ على النتيجة.	**الفرق بالأيام-استثناء**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد كل الأسابيع المنقضية بين ***date/datetime1*** و***date/datetime2***. لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالأسابيع**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد كل الأسابيع المنقضية بين ***date/datetime1*** و***date/datetime2*** (بما في ذلك هذين التاريخين). لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالأسابيع-تضمين**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد كل الأسابيع المنقضية بين ***date/datetime1*** و***date/datetime2*** (مع استبعاد هذين التاريخين). لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالأسابيع-استثناء**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد كل الشهور المنقضية بين ***date/datetime1*** و***date/datetime2***. لا يؤثر ترتيب التاريخين على الناتج.	**الفرق بالشهور**(*<date/datetime1>*, *<date/datetime2>*)
إرجاع عدد كل الشهور المنقضية بين ***date/datetime1*** و***date/datetime2*** (بما في ذلك هذين التاريخين). لا يؤثر ترتيب التاريخين على النتيجة.	**الفرق بالشهور-تضمين**(*<date/datetime1>*, *<date/datetime2>*)

الوصف	الصياغة
مع استبعاد هذين (***date/datetime2*** و ***date/datetime1*** إرجاع عدد كل الشهور المنقضية بين التاريخين). لا يؤثر ترتيب التاريخين على الناتج.	**الفرق بالشهور-استثناء** (*<date/datetime1>*, *<date/datetime2>*)
لا يؤثر ترتيب قيمتي التاريخ على ***date/datetime2*** و ***date/datetime1*** إرجاع عدد السنوات بين النتيجة.	**الفرق بالسنوات**(*<date/datetime1>*, *<date/datetime2>*)
بما في ذلك هذين التاريخين). لا يؤثر (***date/datetime2*** و ***date/datetime1*** إرجاع عدد السنوات بين ترتيب قيمتي التاريخ على النتيجة.	**الفرق بالسنوات-تضمين** (*<date/datetime1>*, *<date/datetime2>*)
مع استبعاد هذين التاريخين). لا يؤثر (***date/datetime2*** و ***date/datetime1*** إرجاع عدد السنوات بين ترتيب قيمتي التاريخ على النتيجة.	**الفرق بالسنوات-استثناء** (*<date/datetime1>*, *<date/datetime2>*)

دوال الوقت من اليوم(English)

الوصف	الصياغة
تحويل السلسلة المحددة إلى وقت من اليوم	**الوقت من اليوم**(*<text>*)
إرجاع مكون الثانية من ***timeofday/datetime attribute***.	**استخراج الثانية**(*<time/datetime>*)
إرجاع مكون الدقيقة من ***timeofday/datetime attribute***.	**استخراج الدقيقة**(*<time/datetime>*)
إرجاع مكون الساعة من ***timeofday/datetime attribute***.	**استخراج الساعة**(*<time/datetime>*)

دوال التاريخ والوقت(English)

الوصف	الصياغة
إرجاع ***date*** والوقت الحاليين عند بدء الجلسة.	**التاريخ/الوقت الحالي**()
تحويل السلسلة المحددة إلى قيمة ***datetime***	**التاريخ/الوقت**(*<text>*)
إعداد وقت ***date*** بربط ***date*** والوقت من اليوم معًا.	**ربط التاريخ والوقت** (*<date>*, *<time>*)
إرجاع عدد الثواني بين ***datetime1*** و ***datetime2***.	**الفرق بالثواني**(*<datetime1>*, *<datetime2>*) **الفرق بالثواني** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الدقائق بين ***datetime1*** و ***datetime2*** بما يشتمل على القيمتين.	**الفرق بالثواني-تضمين** (*<datetime1>*,

الوصف	الصياغة
	<datetime2>) **الفرق بالثواني-تضمين** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الثواني بين ***datetime1*** و***datetime2*** مع استبعاد القيمتين.	**الفرق بالثواني-استثناء** (*<datetime1>*, *<datetime2>*) **الفرق بالثواني-استثناء** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الدقائق بين ***datetime1*** و***datetime2***.	**الفرق بالدقائق**(*<datetime1>*, *<datetime2>*) **الفرق بالدقائق** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الدقائق بين ***datetime1*** و***datetime2*** بما يشتمل على القيمتين.	**الفرق بالدقائق-تضمين** (*<datetime1>*, *<datetime2>*) **الفرق بالدقائق-تضمين** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الدقائق بين ***datetime1*** و***datetime2*** مع استبعاد القيمتين.	**الفرق بالدقائق-استثناء** (*<datetime1>*, *<datetime2>*) **الفرق بالدقائق-استثناء** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الساعات بين ***datetime1*** و***datetime2***.	**الفرق بالساعات**(*<datetime1>*, *<datetime2>*) **الفرق بالساعات** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الساعات بين ***datetime1*** و***datetime2*** بما يشتمل على القيمتين.	**الفرق بالساعات-تضمين** (*<datetime1>*, *<datetime2>*) **الفرق بالساعات-تضمين** (*<timeOfDay1>*, *<timeOfDay2>*)
إرجاع عدد الساعات بين ***datetime1*** و***datetime2*** مع استبعاد القيمتين.	**الفرق بالساعات-استثناء** (*<datetime1>*,

الوصف	الصياغة
	<datetime2>) **الفرق بالساعات-استثناء**(<timeOfDay1>, <timeOfDay2>)
استخراج **date** من **datetime attribute**.	**استخراج التاريخ**(<datetime>)
استخراج الوقت من اليوم من **datetime attribute**. يمكن استخدام هذه الدالة لإعداد قيمة **timeofday attribute** على الوقت الذي يتم فيه تنفيذ القاعدة، عن طريق استخراج الوقت من الوقت و **date** الحاليين.	**استخراج الوقت من اليوم**(<datetime>)
إضافة عدد من الساعات إلى وقت **date**.	**إضافة ساعات**(<datetime>, <num_hours>) **إضافة ساعات**(<timeOfDay>, <num_hours>)
إضافة عدد من الدقائق إلى وقت **date**.	**إضافة دقائق**(<datetime>, <num_minutes>) **إضافة دقائق**(<timeOfDay>, <num_minutes>)
إضافة عدد من الثواني إلى وقت **date**.	**إضافة ثوانٍ**(<datetime>, <num_seconds>) **إضافة ثوانٍ**(<timeOfDay>, <num_seconds>)

دوال النص(English)

الوصف	الصياغة
جمع القيمة **text1** والقيمة **text2** لتكوين قيمة **text** مفردة. ملاحظة: يمكن استخدام أي نوع من المتغيرات. تتم صياغة القيم باستخدام أداة الصياغة المثبتة في جلسة القاعدة.	<text1> **&** <text2>
جمع القيمة **text1** والقيمة **text2** لتكوين قيمة **text** مفردة. ملاحظة: يمكن استخدام أي نوع من المتغيرات. تتم صياغة القيم باستخدام أداة الصياغة المثبتة في جلسة القاعدة.	
إرجاع قيمة منطقية تشير إلى ما إذا كانت قيمة **text** المحددة تحتوي على سلسلة **text** الفرعية المحددة أم لا. مقارنة **text** غير حساسة لحالة الأحرف.	**تحتوي على**(<text>, <substring>)
إرجاع قيمة منطقية تشير إلى ما إذا كانت قيمة **text** المحددة تنتهي بسلسلة **text** الفرعية المحددة أم لا. مقارنة **text** غير حساسة لحالة الأحرف.	**تنتهي بـ**(<text>, <substring>)

الوصف	الصياغة
إرجاع قيمة منطقية تشير إلى ما إذا كانت قيمة ***text*** المحددة تمثل عددًا صالحًا.	**عدد**(*<text>*)
إرجاع طول الأحرف لقيمة ***text*** المحددة.	**الطول**(*<text>*)
إرجاع قيمة منطقية تشير إلى ما إذا كانت قيمة ***text*** المحددة تبدأ بسلسلة ***text*** الفرعية المحددة أم لا. مقارنة ***text*** غير حساسة لحالة الأحرف.	**تبدأ بـ**(*<text>*, *<substring>*)
إرجاع سلسلة ***text*** الفرعية التي تبدأ عند الإزاحة المحددة، والتي تمثل الطول المحدد بالحروف. يتم إرجاع حروف أقل إذا تم الوصول إلى نهاية السلسلة.	**سلسلة فرعية**(*<text>*, *<offset>*, *<length>*)
تحويل العدد المحدد أو ***date attribute*** إلى قيمة ***text***.	**نص**(*<number>*) **نص**(*<date>*) **نص**(*<datetime>*) **نص**(*<timeOfDay>*)

دوال الكيان والعلاقة(English)

الوصف	الصياغة
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى واحد" أو "متعدد إلى واحد" أو "متعدد إلى متعدد"، في حالة وجود شرط واحد فقط.	**ل**(*<relationship>*, *<Exp>*)
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى واحد" أو "متعدد إلى واحد" أو "متعدد إلى متعدد"، في حالة وجود شرط واحد أو أكثر.	**للمجال**(*<relationship>*, *<alias>*) **للمجال**(*<relationship>*)
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى متعدد" أو "متعدد إلى متعدد"، عند الحاجة لتحديد ما إذا كان يجب استيفاء كل الأعضاء في مجموعة ***entity*** الهدف للقاعدة. يُستخدم هذا النموذج في حالة وجود شرط واحد فقط في القاعدة.	**للكل**(*<relationship>*, *<Exp>*)
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى متعدد" أو "متعدد إلى متعدد"، عند الحاجة لتحديد ما إذا كان يجب استيفاء كل الأعضاء في مجموعة ***entity*** الهدف للقاعدة. يُستخدم هذا النموذج في حالة وجود شرط واحد أو أكثر في القاعدة.	**للمجال ككل**(*<relationship>*) **للمجال ككل**(*<relationship>*, *<alias>*)
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى متعدد" أو "متعدد إلى متعدد"، عند الحاجة لتحديد ما إذا كان يجب استيفاء أي أعضاء في مجموعة ***entity*** الهدف للقاعدة. يُستخدم هذا النموذج في حالة وجود شرط واحد فقط في القاعدة.	**موجود**(*<relationship>*, *<Exp>*)
يُستخدم للإشارة من ***entity*** إلى ***entity*** آخر في ***relationship*** بالنوع "واحد إلى متعدد" أو "متعدد إلى متعدد"، عند الحاجة لتحديد ما إذا كان يجب استيفاء أي أعضاء في مجموعة ***entity*** الهدف للقاعدة. يُستخدم هذا النموذج في حالة وجود شرط واحد أو أكثر في القاعدة.	**المجال الموجود**(*<relationship>*) **المجال الموجود**(*<relationship>*, *<alias>*)
تُستخدم كاستنتاج للإشارة إلى أن مثيل ***entity*** هو أحد أعضاء ***relationship***. تُستخدم كشرط للتأكد من أن مثيل ***entity*** هو هدف للعلاقة ***relationship*** حيث مثيل ***entity*** الثاني هو المصدر.	**عضو في**(*<target>*, *<relationship>*) **عضو في**(*<target>*, *<alias>*,

الوصف	الصياغة
	<relationship>)
تُستخدم كشرط لاختبار ما إذا كان مثيل ***entity*** ليس هدفًا للعلاقة ***relationship*** حيث مثيل ***entity*** الثاني هو المصدر.	**ليس عضوًا في**(*<target>*, *<relationship>*)
حساب عدد المثيلات الموجودة للكيان ***entity***.	**عدد المثيلات**(*<relationship>*)
حساب عدد المثيلات الموجودة لكيان ***entity*** المخصص له سمة ***entity-level attribute*** معينة بقيمة محددة.	**عدد المثيلات إذا**(*<relationship>*, *<Exp>*)
الحصول على أعلى/أحدث قيمة لمتغير ***entity-level*** لكل مثيلات ***entity***.	**الحد الأقصى للمثيل**(*<relationship>*, *<number-attr>*) **الحد الأقصى للمثيل**(*<relationship>*, *<date-attr>*) **الحد الأقصى للمثيل**(*<relationship>*, *<datetime-attr>*) **الحد الأقصى للمثيل**(*<relationship>*, *<time-attr>*)
الحصول على أعلى/أحدث قيمة لمتغير ***entity-level*** لكل مثيلات الكيان ***entity*** المخصص له سمة ***entity-level attribute*** معينة بقيمة محددة.	**الحد الأقصى للمثيل إذا**(*<relationship>*, *<number-attr>*, *<condition>*) **الحد الأقصى للمثيل إذا**(*<relationship>*, *<date-attr>*, *<condition>*) **الحد الأقصى للمثيل إذا**(*<relationship>*, *<datetime-attr>*, *<condition>*) **الحد الأقصى للمثيل إذا**(*<relationship>*, *<time-attr>*, *<condition>*)
الحصول على أقل/أقدم قيمة لمتغير ***entity-level*** لكل مثيلات ***entity***.	**الحد الأدنى للمثيل**(*<relationship>*, *<number-attr>*) **الحد الأدنى للمثيل**(*<relationship>*, *<date-attr>*) **الحد الأدنى للمثيل**(*<relationship>*, *<datetime-attr>*) **الحد الأدنى للمثيل**(*<relationship>*, *<time-attr>*)
الحصول على أقل/أقدم قيمة لمتغير ***entity-level*** لكل مثيلات الكيان ***entity*** المخصص له سمة ***entity-level attribute*** معينة بقيمة محددة.	**الحد الأدنى للمثيل إذا**(*<relationship>*, *<number-attr>*, *<condition>*) **الحد الأدنى للمثيل إذا**(*<relationship>*, *<date-attr>*, *<con-*

الصياغة	الوصف
dition>) **الحد الأدنى للمثيل إذا**(*<relationship>*, *<datetime-attr>*, *<condition>*) **الحد الأدنى للمثيل إذا**(*<relationship>*, *<time-attr>*, *<condition>*)	
مجموع المثيلات(*<relationship>*, *<number-attr>*)	الحصول على مجموع كل المثيلات لمتغير ***entity-level***.
مجموع المثيلات إذا(*<relationship>*, *<number-attr>*, *<condition>*)	الحصول على مجموع كل مثيلات متغير ***entity-level***، التي تكون فيها قيمة ***attribute*** المنطقية لـ ***entity-level*** المحدد في ***entity*** بالقيمة صحيح.
قيمة المثيل إذا(*<relationship>*, *<number-attr>*, *<condition>*) **قيمة المثيل إذا**(*<relationship>*, *<text-attr>*, *<condition>*) **قيمة المثيل إذا**(*<relationship>*, *<date-attr>*, *<condition>*) **قيمة المثيل إذا**(*<relationship>*, *<datetime-attr>*, *<condition>*) **قيمة المثيل إذا**(*<relationship>*, *<time-attr>*, *<condition>*)	الحصول على قيمة من مثيل ***entity*** فريد، يتم تحديدها من مثيلات ***entity*** الهدف لـ ***relationship*** بواسطة شرط. • إذا كان الشرط يحدد مثيل ***entity*** هدف واحد، تكون القيمة هي القيمة التي يتم حسابها مقابل مثيل ***entity***. • إذا كان هناك أكثر من مثيل هدف واحد يستوفي الشرط، يتم إرجاع ***uncertain***. • إذا لم تكن هناك أية مثيلات هدف تستوفي الشرط وكانت قيمة ***relationship*** معروفة، تكون القيمة هي ***uncertain***.
مثيلات متساوية(*<instance1>*, *<instance2>*)	تحديد ما إذا كان هناك مثيلان متماثلان للكيان ***entity***.
مثيلات غير متساوية(*<instance1>*, *<instance2>*)	تحديد ما إذا كان هناك مثيلان غير متماثلتين للكيان ***entity***.
استنتاج مثيل(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** **موجود**	يٌستخدم كاستنتاج لإثبات وجود مثيل كيان ***entity*** وأنه أحد أعضاء ***relationship***.

الدوال التحليلية المؤقتة(English)

الصياغة	الوصف
عدد القيم المختلفة بالفاصل الزمني(*<start-date>*, *<end-date>*, *<variable>*) **عدد القيم المختلفة بالفاصل الزمني**	حساب عدد القيم المختلفة المعروفة للمتغير، في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ).

الوصف	الصياغة
	(*<start-date>*, *<end-date>*, *<condition>*)
بما في ذلك هذا) ***date*** حساب عدد القيم المختلفة المعروفة للمتغير في الفاصل الزمني من تاريخ البداية مع استبعاد هذا التاريخ)، ويتم فقط تضمين الأوقات التي يكون فيها عامل ***date*** التاريخ) إلى تاريخ النهاية التصفية المنطقي بالقيمة صحيح.	**عدد القيم المختلفة بالفاصل الزمني إذا** (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)
بما في ذلك هذا التاريخ) إلى تاريخ) ***date*** حساب مجموع متغير رقم أو عملة في الفاصل الزمني ما بين البداية النهاية ***date*** مع استبعاد هذا التاريخ). يفترض أن تكون السمة ***attribute*** كمية يومية.	**المجموع اليومي للفاصل الزمني**(*<start-date>*, *<end-date>*, *<number-attr>*)
بما في ذلك) ***date*** حساب مجموع كل القيم اليومية لمتغير رقم أو عملة في الفاصل الزمني من تاريخ البداية مع استبعاد هذا التاريخ)، ويتم فقط تضمين الأوقات التي يكون فيها الشرط) ***date*** هذا التاريخ) إلى تاريخ النهاية بالقيمة صحيح.	**المجموع اليومي للفاصل الزمني إذا** (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)
بما في ذلك هذا التاريخ) إلى تاريخ) ***date*** تحديد قيمة الحد الأقصى لمتغير في الفاصل الزمني من تاريخ البداية النهاية ***date*** (مع استبعاد هذا التاريخ).	**الحد الأقصى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<number-attr>*) **الحد الأقصى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<date-attr>*) **الحد الأقصى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **الحد الأقصى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<time-attr>*)
بما في ذلك هذا التاريخ) إلى تاريخ) ***date*** تحديد قيمة الحد الأقصى لمتغير في الفاصل الزمني من تاريخ البداية النهاية ***date*** (مع استبعاد هذا التاريخ)، ويتم فقط تضمين الأوقات التي يكون فيها الشرط بالقيمة صحيح.	**الحد الأقصى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **الحد الأقصى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*,

الوصف	الصياغة
	<date-attr>, *<condition>*) **الحد الأقصى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **الحد الأقصى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)
تحديد قيمة الحد الأدنى لمتغير في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ).	**الحد الأدنى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<number-attr>*) **الحد الأدنى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<date-attr>*) **الحد الأدنى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **الحد الأدنى للفاصل الزمني**(*<start-date>*, *<end-date>*, *<time-attr>*)
تحديد قيمة الحد الأدنى لمتغير في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ)، ويتم فقط تضمين الأوقات التي يكون فيها الشرط بالقيمة صحيح.	**الحد الأدنى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **الحد الأدنى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **الحد الأدنى للفاصل الزمني إذا**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*)

الوصف	الصياغة
	الحد الأدنى للفاصل الزمني إذا(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)
حساب متوسط قيمة متغير رقم أو عملة في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ)، مقدرًا بالفترة الزمنية التي تنطبق عليها هاتان القيمتان.	**المتوسط المرجح في الفاصل الزمني**(*<start-date>*, *<end-date>*, *<number-attribute>*)
حساب متوسط قيمة متغير رقم أو عملة في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ)، ويتم فقط تضمين الأوقات التي يكون فيها شرط القيمة المنطقية بالقيمة صحيح (ويتم تقديره من خلال الفترة الزمنية التي يتم تطبيق أية قيمة من القيمتين وعندما يكون عامل التصفية بالقيمة صحيح).	**المتوسط المرجح في الفاصل الزمني إذا** (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)
إرجاع القيمة صحيح فقط إذا كان شرط القيمة المنطقية 'صحيح' في كل الأوقات في الفاصل الزمني ما بين البداية ***date*** (بما في ذلك هذا التاريخ) والنهاية ***date*** (مع استبعاد هذا التاريخ).	**دائمًا بالفاصل الزمني** (*<start-date>*, *<end-date>*, *<condition>*)
إرجاع القيمة صحيح فقط إذا كان شرط القيمة المنطقية 'صحيح' على الأقل لعدد الأيام المحدد (ليس بالضرورة أن تكون متتالية) في الفاصل الزمني من تاريخ البداية ***date*** بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ).	**الأيام المحددة على الأقل بالفاصل الزمني** (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)
إرجاع القيمة صحيح فقط إذا كان شرط القيمة المنطقية 'صحيح' على الأقل لعدد محدد من الأيام المتتالية في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ).	**الأيام المتعاقبة بالفاصل الزمني** (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)
إرجاع القيمة صحيح فقط إذا كان شرط القيمة المنطقية هو 'صحيح دائمًا' في الفاصل الزمني من تاريخ البداية ***date*** (بما في ذلك هذا التاريخ) إلى تاريخ النهاية ***date*** (مع استبعاد هذا التاريخ).	**أحيانًا في الفاصل الزمني**(*<start-date>*, *<end-date>*, *<condition>*)
إرجاع قيمة سمة ***attribute*** المحددة في التاريخ المحدد ***date***.	**قيمة في**(*<date>*, *<value>*)
إرجاع آخر تاريخ ***date*** كان فيه شرط القيمة المنطقية بالقيمة 'صحيح'، بالرجوع للخلف اعتبارًا من التاريخ المحدد ***date*** (بما في ذلك هذا التاريخ).	**تاريخ آخر شرط صحيح** (*<date>*, *<condition>*)
إرجاع التاريخ ***date*** الذي يكون عنده شرط القيمة المنطقية بالقيمة 'صحيح' في المرة القادمة، بالتقدم للأمام اعتبارًا من التاريخ المحدد ***date*** (بما في ذلك هذا التاريخ).	**تاريخ الشرط الصحيح التالي**(*<date>*, *<condition>*)

الوصف	الصياغة
إرجاع قيمة ***date*** مساوية لقيمة أحدث تاريخ محتمل ***date*** - أي أنه مضمون أن قيمة ***date*** لاحقة لأية قيمة ***date*** أخرى يمكن استخدامها من خلال ***date attribute*** أو يمكن تقييم أحد التعبيرات إليها.	**الأحدث**()
إرجاع قيمة ***date*** مساوية لقيمة أقدم تاريخ محتمل ***date*** - أي أنه مضمون أن قيمة ***date*** سابقة لأية قيمة ***date*** أخرى يمكن استخدامها من خلال ***date attribute*** أو يمكن تقييم أحد التعبيرات إليها.	**الأقدم**()
إرجاع متغير رقمي يتغير كل يوم وهو عدد الأيام الكاملة منذ ***date***.	**مؤقت بالأيام منذ** (*<date>*, *<end-date>*)
إرجاع متغير رقمي يتغير كل أسبوع وهو عدد الأسابيع الكاملة منذ ***date***.	**مؤقت بالأسابيع منذ** (*<date>*, *<end-date>*)
إرجاع متغير رقمي يتغير كل شهر ويمثل عدد الشهور الكاملة منذ ***date***. ملاحظة: إذا كانت القيمة المتوفرة ***date*** لاحقة لليوم 28 من الشهر والشهر التالي يحتوي على أيام أقل من الشهر الحالي، سيتم تكوين نقطة التغيير للشهر السنوي في اليوم الأخير من هذا الشهر. على سبيل المثال، إذا كانت القيمة المتوفرة ***date*** هي 28 أو 29 أو 30 أو 31 يناير 2007، ستكون نقطة التغيير الأولى هي 28 فبراير 2007.	**مؤقت بالشهور منذ** (*<date>*, *<end-date>*)
إرجاع متغير رقمي يتغير كل سنة وهو عدد السنوات الكاملة منذ ***date***.	**مؤقت بالسنوات منذ** (*<date>*, *<end-date>*)
إرجاع قيمة ***attribute*** منطقية تتغير على مدار الوقت وتكون صحيحة فقط إذا كان شرط القيمة المنطقية بالقيمة صحيح لكل العدد المحدد من الأيام السابقة باستثناء اليوم الحالي.	**مؤقت دائمًا بالأيام** (*<days>*, *<condition>*)
إرجاع قيمة ***attribute*** منطقية تتغير على مدار الوقت وتكون صحيحة فقط إذا كان شرط القيمة المنطقية بالقيمة صحيح على الأقل للحد الأدنى من عدد الأيام المتتالية في أي وقت خلال عدد أيام المجموعة السابقة باستثناء اليوم الحالي.	**مؤقت للأيام المتعاقبة** (*<minDays>*, *<days>*, *<condition>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة فقط إذا كان شرط القيمة المنطقية بالقيمة صحيح في أي وقت خلال العدد المحدد من الأيام السابقة، باستثناء اليوم الحالي.	**مؤقت أحيانًا بالأيام** (*<days>*, *<condition>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة بعد التاريخ ***date*** وخطأ إذا كانت في هذا التاريخ أو قبله.	**مؤقت بعد**(*<date>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة قبل التاريخ ***date*** وخطأ إذا كانت في هذا التاريخ أو بعده.	**مؤقت قبل**(*<date>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة في التاريخ ***date*** وخطأ إذا كانت قبله أو بعده.	**مؤقت بتاريخ**(*<date>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة إذا كانت مماثلة للتاريخ ***date*** أو بعده وخطأ قبله.	**مؤقت في أو بعد التاريخ**(*<date>*)
إرجاع قيمة ***attribute*** منطقية تختلف على مدار الوقت فتكون صحيحة إذا كانت مماثلة للتاريخ ***date*** أو واقعة قبله وخطأ بعده.	**مؤقت في أو قبل التاريخ**(*<date>*)
إرجاع ***attribute*** مؤقتة واحدة (على مستوى الكيان ***entity*** المصدر) من علاقة ***relationship*** و***attribute*** قيمة في الكيانات، مع القيم التي تبدأ فعاليتها من تاريخ البداية ***date attribute***.	**مؤقت من تاريخ البداية** (*<relationship>*, *<date>*, *<value>*)
إرجاع ***attribute*** مؤقتة واحدة (على مستوى الكيان ***entity*** المصدر) من ***relationship*** و***attribute*** قيمة	**مؤقت حتى تاريخ**

الصياغة	الوصف
النهاية(*<relationship>*, *<date>*, *<value>*)	***date attribute***. في الكيانات، مع القيم التي لها فعالية حتى تاريخ النهاية
مؤقت من النطاق (*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	***attribute*** و ***relationship*** (المصدر) من علاقة ***entity*** مؤقتة واحدة (على مستوى الكيان) ***attribute*** إرجاع بما في ذلك هذا التاريخ)) ***date attribute*** قيمة في الكيانات، مع القيم التي تبدأ فعاليتها من تاريخ البداية في حالة ***uncertain*** مع استبعاد هذا التاريخ). وتكون القيمة هي(***date attribute*** حتى تاريخ النهاية التالي. ***date*** انتهاء الصلاحية قبل تاريخ البداية
مؤقت في اليوم من الأسبوع(*<startdate>*, *<enddate>*)	إرجاع القيمة صحيح للتواريخ التي تمثل أيام الأسبوع والقيمة خطأ للتواريخ التي تمثل نهايات الأسبوع بدءًا من مع استبعاد هذا التاريخ). يتم (***date*** المحدد (بما في ذلك هذا التاريخ) حتى تاريخ النهاية ***date*** تاريخ البداية ***date***. إذا كان يوم الأسبوع خارج نطاق ***uncertain*** إرجاع القيمة
مؤقت مرة واحدة بكل شهر(*<startdate>*, *<enddate>*, *<day-ofmonth>*)	إرجاع القيمة صحيح إذا كان اليوم يساوي معامل اليوم من الشهر وإرجاع القيمة خطأ لكل الأيام الأخرى من مع استبعاد (***date*** بما في ذلك هذا التاريخ) إلى تاريخ النهاية المحدد) ***date*** الشهر من تاريخ البداية المحدد في حالة تجاوز اليوم من الشهر ***date***. إذا كان اليوم خارج نطاق ***uncertain*** هذا التاريخ). ويتم إرجاع القيمة عدد أيام الشهر الحالي، تكون القيمة صحيح لليوم الأخير من هذا الشهر، وبهذا تقوم الدالة بإرجاع القيمة صحيح ليوم واحد فقط من الشهر.

دوال حدث المراجعة(English)

الصياغة	الوصف
خطأ (*<text>*)	حدث خطأ يُستخدم لتمرير رسالة إلى المستخدم ومنعه من الاستمرار في الاستقصاء حتى يتوقف تطبيق الشرط الذي أدى إلى حدوث هذا الخطأ.
تحذير (*<text>*)	حدث تحذير يُستخدم لتمرير رسالة إلى المستخدم، ولكن يسمح له بالاستمرار على الرغم من الشرط الذي أدى إلى هذا التحذير.

الدوال المهملة(English)

الصياغة	الوصف
استدعاء دالة مخصصة(*<A>*, *<B>*)	لنجاح Determinations إظهار نتيجة استدعاء خارجي لمكتبة رموز. يجب توفير مكتبة رموز لمحرك استدعاء الدالة المخصصة.

Conectores lógicos(English)

Sintaxe	Descrição
se	Termo opcional que pode aparecer no final de uma linha de conclusão que tem a seguinte prova
e	Conjunção lógica entre dois ***attributes***
ou	Disjunção lógica entre dois ***attributes***

Sintaxe	Descrição
qualquer um **qualquer** **um de** **algum** **ao menos um destes é verdadeiro** **qualquer um destes está satisfeito**	O elemento de agrupamento usado com disjunções em que dois ou mais ***attributes*** precisam ser agrupados
ambos **todos** **tudo** **todos estes são verdadeiros** **todos estes estão satisfeitos**	O elemento de agrupamento usado com conjunções em que dois ou mais ***attributes*** precisam ser agrupados
caso contrário	Termo que aparece no final de uma regra de tabela para indicar a cláusula caso contrário
é	Termo usado em uma entrada de legenda entre a frase abreviada e o ***attribute text*** completo

Funções lógicas(English)

Sintaxe	Descrição
não é verdade que *<expr>*	Operador usado para retornar verdadeiro se ***attribute*** tiver um valor que seja falso
<var> **é certo** *<var>* **é certa** *<var>* **é certos** *<var>* **é certas** **é certo que** [**ou não**]*<expr>*	Operador usado para retornar verdadeiro se ***attribute*** tiver um valor que não seja ***uncertain***
<var> **é incerto** *<var>* **é incerta** *<var>* **é incertos** *<var>* **é incertas** **é incerto se** *<expr>* **é incerto que** [**ou não**]*<expr>* **não é certo se** *<expr>* **incerto**	Operador usado para retornar verdadeiro se o valor de ***attribute*** for ***uncertain***
<var> **é conhecido** *<var>* **é conhecida** *<var>* **é conhecidos** *<var>* **é conhecidas** **sabe-se que** [**ou não**]*<expr>*	Operador usado para retornar verdadeiro se ***attribute*** tiver qualquer valor

Sintaxe	Descrição
<var> **é desconhecido** *<var>* **é desconhecida** *<var>* **é desconhecidos** *<var>* **é desconhecidas** **não se sabe se** [**ou não**]*<expr>* **desconhecido**	Operador usado para retornar verdadeiro se ***attribute*** não tiver nenhum valor

Constantes lógicas(English)

Sintaxe	Descrição
verdadeiro	Valor verdadeiro constante usado para regras de tabela.
falso	Valor falso constante usado para regras de tabela.
incerto	Valor ***uncertain*** constante usado para regras de tabela.

Operadores de comparação(English)

Sintaxe	Descrição
*<x>***<***<y>*	Menor que Observação: não há forma de linguagem natural quando esse operador é usado com valores numéricos e monetários.
<x> **>** *<y>*	Maior que Observação: não há forma de linguagem natural quando esse operador é usado com valores numéricos e monetários.
*<x>***<=***<y>*	Menor que ou igual a
<x> **>=** *<y>*	Maior que ou igual a
*<x>***=***<y>*	É Igual
<x> **<>** *<y>*	Diferente

Funções numéricas(English)

Sintaxe	Descrição
Número(*<numText>*)	Converta a string especificada em um valor numérico
<x> **+** *<y>*	Adição matemática

Sintaxe	Descrição
<x> **-** *<y>*	Subtração matemática
<x> * *<y>*	Multiplicação matemática
<x> **/** *<y>*	Divisão matemática
<x> \ *<y>*	Divisão do integer
<x> **modulo** *<y>*	Resto da divisão inteira
Máximo(*<x>*, *<y>*) **Máximo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Retorna o maior de dois valores
Mínimo(*<x>*, *<y>*) **Mínimo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Retorna o menor de dois valores
Xy(*<x>*, *<y>*)	x elevado a y
Ex(*<x>*)	Constante e elevada a x
Abs(*<x>*) **|***<val>***|**	Valor absoluto de x
Ln(*<x>*)	Logaritmo natural de x
Log(*<x>*)	Logaritmo base 10 de x
Raiz quadrada(*<x>*)	Raiz quadrada de x
Arredond(*<x>*, *<n>*)	Arredonda x para n casas decimais
Trunc(*<x>*, *<n>*)	x truncado para n casas decimais
Sen(*<x>*)	Seno de x
Cos(*<x>*)	Cosseno de x
Tan(*<x>*)	Tangente de x
Asen(*<x>*)	Arco seno de x
Acos(*<x>*)	Arco seno de x
Atan(*<x>*)	Arco tangente de x

Funções de data(English)

Sintaxe	Descrição
DataAtual() **a data atual**	Retorna a ***date*** atual ao início da sessão.

Sintaxe	Descrição
Data(*<text>*)	Converte a string especificada em um valor ***date***
DataDeCriação(*<year>*, *<month>*, *<day>*)	Retorna uma ***date*** formada pelo ano, mês e dia especificados.
ExtrairDia(*<date/datetime>*)	Retorna o componente dia de um atributo ***date/datetime attribute***.
ExtrairMês(*<date/datetime>*)	Retorna o componente mês de um atributo ***date/datetime attribute***.
ExtrairAno(*<date/datetime>*)	Retorna o componente ano de um atributo ***date/datetime attribute***.
PróximoDiaDaSemana (*<date/datetime>*, *<day>*)	Retorna a ***date*** do próximo dia da semana em ou depois de ***date*** (dependendo da sintaxe usada).
PróximaData(*<date>*, *<day>*, *<month>*)	Retorna a próxima instância do dia e do mês especificados depois de uma ***date***.
AdicionarDias (*<date/datetime>*, *<num_days>*)	Adiciona/subtrai um número de dias a/de uma ***date***. Quando se usa a forma sintática concisa, o número deve ser um inteiro positivo para adicionar dias à entrada ***date***, ou um número negativo para subtrair dias da entrada ***date***.
AdicionarSemanas(*<date/datetime>*, *<num_weeks>*)	Adiciona um número de semanas a uma ***date***. Ao usar a forma sintática concisa, o número deve ser um inteiro positivo para adicionar semanas à entrada ***date***.
AdicionarMeses(*<date/datetime>*, *<num_months>*)	Adiciona um número de meses a uma ***date***. Ao usar a forma sintática concisa, o número deve ser um inteiro positivo para adicionar meses à entrada ***date***.
AdicionarAnos(*<date/datetime>*, *<num_years>*)	Adiciona um número de anos a uma ***date***. Ao usar a forma sintática concisa, o número deve ser um inteiro positivo para adicionar anos à entrada ***date***.
ContagemDosDiasDaSemana (*<date1>*, *<date2>*)	Conta o número de dias da semana entre ***date***1 e ***date***2. Ou seja, o número de dias entre segunda e sexta-feira. Observação: A ***date*** anterior é incluída e a ***date*** posterior é excluída.
InícioDoAno(*<date/datetime>*)	Retorna a primeira ***date*** do ano na qual a ***date*** cai.
FimDoAno(*<date/datetime>*)	Retorna a última ***date*** do ano na qual a ***date*** cai.
DiferençaDeDia(*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de dias completos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeDiaInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de dias completos (inclusive) entre ***date/datetime1*** e ***date/datetime2***. Esse cálculo inclui ambas as extremidades. Onde as datas são as mesmas, o resultado é 1. A ordem das duas datas não afeta o resultado.
DiferençaDeDiaExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de dias completos (exclusivos) entre ***date/datetime1*** e ***date/datetime2***. Esse cálculo exclui ambas as extremidades. Onde as datas são as mesmas, o resultado é 0. A ordem das duas datas não afeta o resultado.
DiferençaDeSemana(*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de semanas transcorridas completas entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.

Sintaxe	Descrição
DiferençaDeSemanaInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número inclusivo de semanas transcorridas completas entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeSemanaExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número exclusivo de semanas transcorridas completas entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMês(*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de meses transcorridos completos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMêsInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de meses transcorridos completos inclusivos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMêsExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de meses transcorridos completos exclusivos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAno(*<date/datetime1>*, *<date/datetime2>*)	Retorna o número de anos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAnoInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número inclusivo de anos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAnoExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Retorna o número exclusivo de anos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.

Funções de hora do dia(English)

Sintaxe	Descrição
HoraDoDia(*<text>*)	Converte a string dada em uma hora do dia
ExtrairSegundo(*<time/datetime>*)	Retorna o componente segundo de um atributo ***timeofday/datetime attribute***.
ExtrairMinuto(*<time/datetime>*)	Retorna o componente minuto de um atributo ***timeofday/datetime attribute***.
ExtrairHora(*<time/datetime>*)	Retorna o componente hora de um atributo ***timeofday/datetime attribute***.

Funções de data e hora(English)

Sintaxe	Descrição
DataHoraAtual()	Retorna a ***date*** e a hora atuais ao início da sessão.

Sintaxe	Descrição
DataHora(*<text>*)	Converte a string especificada em um valor ***datetime***
ConcatenarDataHora(*<date>*, *<time>*)	Define a hora ***date*** unindo a ***date*** e a hora do dia.
DiferençaDeSegundo(*<datetime1>*, *<datetime2>*) **DiferençaDeSegundo** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número de segundos entre ***datetime1*** e ***datetime2***.
DiferençaDeSegundoInclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeSegundoInclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número inclusivo de segundos entre ***datetime1*** e ***datetime2***.
DiferençaDeSegundoExclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeSegundoExclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número exclusivo de segundos entre ***datetime1*** e ***datetime2***.
DiferençaDeMinuto(*<datetime1>*, *<datetime2>*) **DiferençaDeMinuto** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número de minutos entre ***datetime1*** e ***datetime2***.
DiferençaDeMinutoInclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeMinutoInclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número inclusivo de minutos entre ***datetime1*** e ***datetime2***.
DiferençaDeMinutoExclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeMinutoExclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número exclusivo de minutos entre ***datetime1*** e ***datetime2***.
DiferençaDeHora(*<datetime1>*, *<datetime2>*) **DiferençaDeHora** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número de horas entre ***datetime1*** e ***datetime2***.
DiferençaDeHoraInclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeHoraInclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Retorna o número inclusivo de horas entre ***datetime1*** e ***datetime2***.
DiferençaDeHoraExclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeHoraExclusiva**	Retorna o número exclusivo de horas entre ***datetime1*** e ***datetime2***.

Sintaxe	Descrição
(*<timeOfDay1>*, *<timeOfDay2>*)	
ExtrairData(*<datetime>*)	Extrai a ***date*** de um ***datetime attribute***.
ExtrairHoraDoDia(*<datetime>*)	Extrai a hora do dia de um atributo ***datetime attribute***. Pode ser usado para definir o valor de um atributo ***timeofday attribute*** como a hora em que a regra é executada extraindo a hora da ***date*** e da hora atuais.
AdicionarHoras(*<datetime>*, *<num_hours>*) **AdicionarHoras**(*<timeOfDay>*, *<num_hours>*)	Adiciona um número de horas a uma hora ***date***.
AdicionarMinutos(*<datetime>*, *<num_minutes>*) **AdicionarMinutos** (*<timeOfDay>*, *<num_minutes>*)	Adiciona um número de minutos a uma hora ***date***.
AdicionarSegundos (*<datetime>*, *<num_seconds>*) **AdicionarSegundos** (*<timeOfDay>*, *<num_seconds>*)	Adiciona um número de segundos a uma hora ***date***.

Funções de texto(English)

Sintaxe	Descrição
<text1> **&** *<text2>*	Combina ***text1*** com ***text2***, e assim por diante, para formar um único valor de ***text***. Observação: você pode usar variáveis de qualquer tipo. Os valores são formatados com o formatador instalado na sessão de regras.
	Combina ***text1*** com ***text2***, e assim por diante, para formar um único valor de ***text***. Observação: você pode usar variáveis de qualquer tipo. Os valores são formatados com o formatador instalado na sessão de regras.
Contém(*<text>*, *<substring>*)	Retorna um valor booliano que indica se o valor de ***text*** fornecido contém a substring ***text*** dada. A comparação ***text*** é sensível a maiúsculas e minúsculas.
TerminaCom(*<text>*, *<substring>*)	Retorna um valor booliano que indica se o valor de ***text*** fornecido termina com a substring ***text*** dada. A comparação ***text*** é sensível a maiúsculas e minúsculas.
ÉNúmero(*<text>*)	Retorna um valor booliano que indica se o valor de ***text*** dado representa um número válido.
Comprimento(*<text>*)	Retorna o comprimento do caractere do valor ***text*** dado.
ComeçaCom(*<text>*, *<substring>*)	Retorna um valor booliano que indica se o valor de ***text*** fornecido inicia com a substring ***text*** dada. A comparação ***text*** é sensível a maiúsculas e minúsculas.

Sintaxe	Descrição
Subsequência(*<text>*, *<offset>*, *<length>*)	Retorna a substring de ***text*** que começa no recuo dado, que é o comprimento especificado em caracteres. Menos caracteres são retornados se o final da string é alcançado.
Texto(*<number>*) **Texto**(*<date>*) **Texto**(*<datetime>*) **Texto**(*<timeOfDay>*)	Converta o número especificado ou ***date attribute*** em um valor ***text***.

Funções de entidade e relação(English)

Sintaxe	Descrição
Para(*<relationship>*, *<Exp>*) **no caso** (**da** | **das** | **do** | **dos**)*<relationship>***,** *<attr>* *<val>***, no caso** (**da** | **das** | **do** | **dos**)*<relationship>*	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Um", "Muitos para Um" ou "Muitos para Muitos" ***relationship***, em que há somente uma condição.
ParaEscopo(*<relationship>*, *<alias>*) **ParaEscopo**(*<relationship>*) **no caso** (**da** | **das** | **do** | **dos**)*<relationship>* **no caso** (**da** | **das** | **do** | **dos**)*<relationship>* **(**<alias>**)**	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Um", "Muitos para Um" ou "Muitos para Muitos" ***relationship***, em que há uma ou mais condições.
ParaTudo(*<relationship>*, *<Exp>*)	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Muitos" ou "Muitos para Muitos" ***relationship***, quando é necessário determinar se todos os membros do grupo ***entity*** de destino precisam satisfazer a regra. Esse formulário é usado quando há somente uma condição na regra.
ParaTodosEscopos(*<relationship>*) **ParaTodosEscopos**(*<relationship>*, *<alias>*)	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Muitos" ou "Muitos para Muitos" ***relationship***, quando é necessário determinar se todos os membros do grupo ***entity*** de destino precisam satisfazer a regra. Esse formulário é usado quando há uma ou mais condições na regra.
Existe(*<relationship>*, *<Exp>*)	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Muitos" ou "Muitos para Muitos" ***relationship***, quando é necessário determinar se os membros do grupo ***entity*** de destino precisam satisfazer a regra. Esse formulário é usado quando há somente uma condição na regra.
ExisteEscopo(*<relationship>*) **ExisteEscopo**(*<relationship>*, *<alias>*)	Usado para fazer referência de um ***entity*** a outro ***entity*** em uma relação "Um para Muitos" ou "Muitos para Muitos" ***relationship***, quando é necessário determinar se os membros do grupo ***entity*** de destino precisam satisfazer a regra. Esse formulário é usado quando há uma ou mais condições na regra.

Sintaxe	Descrição
ÉMembroDe(*<target>*, *<relationship>*) **ÉMembroDe**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **é membro** (**dos** | **de**)*<relationship>* <ent-target> **(**<alias>**) é membro** (**dos** | **de**)*<relationship>*	Usado como conclusão para deduzir que uma instância de ***entity*** é membro de uma ***relationship***. Usado como uma condição para testar se uma instância de ***entity*** é o destino de uma ***relationship*** para a qual uma segunda instância de ***entity*** é a origem.
NãoÉMembroDe(*<target>*, *<relationship>*)	Usado como uma condição para testar se uma instância de ***entity*** não é um destino de uma ***relationship*** para a qual uma segunda instância de ***entity*** é a origem.
ContagemDeInstâncias (*<relationship>*)	Conta o número de instâncias que existem para uma ***entity***.
ContagemDeInstânciasSe (*<relationship>*, *<Exp>*)	Conta o número de instâncias existentes de uma ***entity*** para a qual um ***entity-level attribute*** específico tem um valor específico.
MáximoDeInstâncias(*<relationship>*, *<number-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<date-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<datetime-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<time-attr>*)	Obtém o valor mais alto/recente de uma variável ***entity-level*** para todas as instâncias da ***entity***.
MáximoDeInstânciasSe (*<relationship>*, *<number-attr>*, *<condition>*) **MáximoDeInstânciasSe** (*<relationship>*, *<date-attr>*, *<condition>*) **MáximoDeInstânciasSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MáximoDeInstânciasSe** (*<relationship>*, *<time-attr>*, *<condition>*)	Obtém o valor mais alto/recente de uma variável ***entity-level*** para todas as instâncias da ***entity*** para as quais um ***entity-level attribute*** específico tem um valor específico.
MínimoDeInstâncias(*<relationship>*, *<number-attr>*) **MínimoDeInstâncias**(*<relationship>*, *<date-attr>*) **MínimoDeInstâncias**(*<rela-*	Obtém o valor mais baixo/menos recente de uma variável ***entity-level*** para todas as instâncias da ***entity***.

Sintaxe	Descrição
tionship>, *<datetime-attr>*) **MínimoDeInstâncias**(*<relationship>*, *<time-attr>*)	
MínimoDeInstânciasSe (*<relationship>*, *<number-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<date-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<time-attr>*, *<condition>*)	Obtém o valor mais baixo/menos recente de uma variável ***entity-level*** para todas as instâncias da ***entity*** para as quais um ***entity-level attribute*** específico tem um valor específico.
SomaDeInstâncias(*<relationship>*, *<number-attr>*)	Obtém a soma de todas as instâncias de uma variável ***entity-level***.
SomaDeInstânciasSe(*<relationship>*, *<number-attr>*, *<condition>*)	Obtém a soma de todas as instâncias de uma variável ***entity-level*** para a qual é verdadeiro de ***entity*** que um ***attribute*** Booliano ***entity-level*** é verdadeiro.
ValorIfInstância(*<relationship>*, *<number-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<text-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<date-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<time-attr>*, *<condition>*)	Obtém um valor de uma instância ***entity*** exclusiva, identificada das instâncias ***entity*** de destino de um ***relationship*** por uma condição. • Se a condição identifica uma única instância ***entity*** de destino, o valor é o calculado em relação a essa instância ***entity***. • Se mais de uma instância de destino satisfaz a condição, ***uncertain*** é retornado. • Se nenhuma instância de destino satisfaz a condição e ***relationship*** é conhecido, o valor é ***uncertain***.
IgualdadesDeInstância (*<instance1>*, *<instance2>*)	Determina se duas instâncias de uma ***entity*** são iguais.
SemIgualdadesDeInstância (*<instance1>*, *<instance2>*)	Determina se duas instâncias de uma ***entity*** não são iguais.
InferirInstância(*<rela-*	Usado como conclusão para inferir que uma instância de ***entity*** existe e é membro de uma

Sintaxe	Descrição
tionship>, <*identity*>) <*rel*>**(**<*identity*>**) existe**	***relationship***.

Funções de argumentos temporais(English)

Sintaxe	Descrição
ContagemDeIntervalosDistintos (<*start-date*>, <*end-date*>, <*variable*>) **ContagemDeIntervalosDistintos** (<*start-date*>, <*end-date*>, <*condition*>)	Conta o número de valores distintos conhecidos da variável no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
ContagemDeIntervalosDistintosSe (<*start-date*>, <*end-date*>, <*variable*>, <*condition*>)	Conta o número de valores distintos conhecidos da variável no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), e inclui hora somente quando um filtro booliano é verdadeiro.
SomaDiáriaDeIntervalos(<*start-date*>, <*end-date*>, <*number-attr*>)	Calcula a soma de uma variável monetária ou numérica no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive). Presume-se que ***attribute*** seja uma quantidade diária.
SomaDiáriaDeIntervalosSe(<*start-date*>, <*end-date*>, <*number-attr*>, <*condition*>)	Calcula a soma de todos os valores diários de uma variável monetária ou numérica no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), e inclui hora somente quando uma condição é verdadeira.
MáximoDeIntervalos(<*start-date*>, <*end-date*>, <*number-attr*>) **MáximoDeIntervalos**(<*start-date*>, <*end-date*>, <*date-attr*>) **MáximoDeIntervalos**(<*start-date*>, <*end-date*>, <*datetime-attr*>) **MáximoDeIntervalos**(<*start-date*>, <*end-date*>, <*time-attr*>)	Seleciona o valor máximo de uma variável no intervalo ***date*** inicial (inclusive) até ***date*** inicial (exclusive).
MáximoDeIntervalosSe(<*start-date*>, <*end-date*>, <*number-attr*>, <*condition*>) **MáximoDeIntervalosSe**(<*start-date*>, <*end-date*>, <*date-attr*>, <*condition*>) **MáximoDeIntervalosSe**(<*start-date*>, <*end-date*>, <*datetime-attr*>, <*condition*>) **MáximoDeIntervalosSe**(<*start-date*>, <*end-date*>, <*time-attr*>, <*con-*	Seleciona o valor máximo de uma variável no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), e inclui hora somente quando uma condição é verdadeira.

Sintaxe	Descrição
dition>)	
MínimoDeIntervalos(*<start-date>*, *<end-date>*, *<number-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<date-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<time-attr>*)	Seleciona o valor mínimo de uma variável no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
MínimoDeIntervalosSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Seleciona o valor mínimo de uma variável no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), e inclui hora somente quando uma condição é verdadeira.
MédiaPonderadaDeIntervalos (*<start-date>*, *<end-date>*, *<number-attribute>*)	Calcula o valor médio de uma variável monetária ou numérica no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), ponderado pelo tempo ao qual cada valor se aplica.
MédiaPonderadaDeIntervalosSe (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Calcula o valor médio de uma variável monetária ou numérica no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive), e inclui hora somente quando uma condição booliana é verdadeira (ponderado pelo tempo ao qual cada valor se aplica e onde o filtro é verdadeiro).
IntervaloSempre(*<start-date>*, *<end-date>*, *<condition>*)	Retorna verdadeiro somente se uma condição booliana é verdadeira sempre no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
IntervalNoMínimo(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Retorna verdadeiro somente se uma condição booliana é verdadeira para pelo menos o número de dias especificado (não necessariamente consecutivos) no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
IntervaloDiasConsecutivos(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Retorna verdadeiro somente se uma condição booliana é verdadeira para pelo menos um número de dias consecutivos especificado no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
IntervaloAlgumasVezes(*<start-date>*, *<end-date>*, *<condition>*)	Retorna verdadeiro somente se uma condição booliana é verdadeira no intervalo ***date*** inicial (inclusive) até ***date*** final (exclusive).
ValorEm(*<date>*, *<value>*)	Retorna o valor do ***attribute*** dado na ***date*** especificada.

Sintaxe	Descrição
QuandoÚltimo(*<date>*, *<condition>*)	Retorna a ***date*** na qual uma condição booliana foi verdadeira pela última vez, desde (inclusive) uma ***date*** especificada.
QuandoPróximo(*<date>*, *<condition>*)	Retorna a próxima ***date*** na qual uma condição booliana será verdadeira a partir de (inclusive) uma ***date*** especificada.
Último()	Retorna um valor ***date*** equivalente à ***date*** mais recente possível, isto é, uma ***date*** que seja seguramente posterior a qualquer outra ***date*** que uma ***date attribute*** possa usar ou para a qual uma expressão possa ser avaliada.
Primeiro()	Retorna um valor ***date*** equivalente à ***date*** mais antiga possível, isto é, uma ***date*** que seja seguramente anterior a qualquer outra ***date*** que uma ***date attribute*** possa usar ou para a qual uma expressão possa ser avaliada.
TemporalDiasDesde(*<date>*, *<end-date>*)	Retorna uma variável numérica que varia a cada dia e é o número de dias completos desde ***date***.
TemporalSemanasDesde(*<date>*, *<end-date>*)	Retorna uma variável numérica que varia a cada semana e é o número de semanas completas desde ***date***.
TemporalMesesDesde(*<date>*, *<end-date>*)	Retorna uma variável numérica que varia a cada mês e é o número de meses completos desde ***date***. Observação: onde a ***date*** fornecida for depois do 28º dia do mês e um mês subsequente tiver menos dias do que o mês fornecido, o ponto de alteração do mês de aniversário será criado no último dia desse mês. Por exemplo, se a ***date*** fornecida for 28, 29, 30 ou 31 de janeiro de 2007, o primeiro ponto de alteração será 28 de fevereiro de 2007.
TemporalAnosDesde(*<date>*, *<end-date>*)	Retorna uma variável numérica que varia a cada ano e é o número de anos completos desde ***date***.
TemporalSempreDias(*<days>*, *<condition>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro somente se uma condição booliana for verdadeira para todos os dias anteriores, e não inclui o dia atual.
TemporalDiasConsecutivos (*<minDays>*, *<days>*, *<condition>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro somente se uma condição booliana é verdadeira para pelo menos um número mínimo de dias consecutivos a qualquer momento no número de dias anteriores definidos, e não inclui o dia atual.
TemporalAlgunsDias(*<days>*, *<condition>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro somente se uma condição booliana é sempre verdadeira em um número especificado de dias anteriores, e não inclui o dia atual.
TemporalApós(*<date>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro depois de uma ***date*** e falso nessa data e antes dela.
TemporalAntes(*<date>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro antes de uma ***date*** e falso nessa data e depois dela.

Sintaxe	Descrição
TemporalEm(*<date>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro em uma ***date*** e falso antes e depois dela.
TemporalEmOuApós(*<date>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro em ***date*** e depois dela e falso antes dessa data.
TemporalEmOuAntes(*<date>*)	Retorna um ***attribute*** booliano que varia ao longo do tempo e é verdadeiro em ***date*** e antes dela e falso depois dessa data.
TemporalDaDataDeInício(*<relationship>*, *<date>*, *<value>*)	Retorna um único ***attribute*** temporal (no nível ***entity*** de origem) de uma ***relationship*** e um valor ***attribute*** nas entidades, com valores em vigor desde a ***date attribute*** inicial.
TemporalDaDataDeTérmino(*<relationship>*, *<date>*, *<value>*)	Retorna um único ***attribute*** temporal (no nível ***entity*** de origem) de uma ***relationship*** e um valor ***attribute*** nas entidades, com valores em vigor até uma ***date attribute*** final.
TemporalDoIntervalo(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Retorna um único ***attribute*** temporal (no nível ***entity*** de origem) de uma ***relationship*** e um valor ***attribute*** nas entidades, com valores em vigor desde o ***date attribute*** inicial (inclusive) até ***date attribute*** final (exclusive). O valor é ***uncertain*** se expira antes do próximo ***date*** inicial.
TemporalDiaDaSemana (*<startdate>*, *<enddate>*)	Retorna verdadeiro em datas que são dias de semana e falso em datas que são finais de semana, desde o ***date*** inicial especificado (inclusive) até ***date*** final (exclusive). Retorna ***uncertain*** fora do intervalo ***date***.
TemporalUmaVezPorMês (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Retorna verdadeiro se o dia é igual ao parâmetro dia-do-mês e falso em todos os demais dias do mês ***date*** inicial especificado (inclusive) até ***date*** final (exclusive). Retorna ***uncertain*** fora do intervalo ***date***. Quando o parâmetro dia-do-mês excede o número de dias do mês atual, o valor é verdadeiro no último dia desse mês para que a função retorne um valor que seja verdadeiro exatamente um dia por mês.

Funções de evento de validação(English)

Sintaxe	Descrição
Erro (*<text>*)	Um evento de erro é usado para encaminhar uma mensagem ao usuário e evitar que ele continue uma investigação até que a condição que ativou o erro não seja mais aplicável.
Aviso (*<text>*)	Um evento de aviso é usado para encaminhar uma mensagem ao usuário, mas permite que ele continue apesar da condição que ativou o aviso.

Funções em remoção gradual(English)

Sintaxe	Descrição
ChamarFunçãoPersonalizada	Retorna o resultado de uma chamada externa para uma biblioteca de códigos. A biblioteca

Sintaxe	Descrição
(<*A*>, <*B*>)	de códigos deve ser fornecida para o mecanismo de determinações para que a chamada personalizada seja bem-sucedida.

逻辑连接词(English)

语法	说明
如果	可选术语可出现在具有以下证明的结论行的末尾
并且 **以及**	两个 ***attributes*** 之间的逻辑与运算
或者	两个 ***attributes*** 之间的逻辑或运算
任一 **一个** **任何** **满足一个条件** **满足任意条件**	需要对两个或两个以上 ***attributes*** 分组时，请对逻辑或运算中使用的要素进行分组。
两者 **所有** **所有条件均成立** **满足以下所有条件**	需要对两个或两个以上 ***attributes*** 分组时，请对逻辑与运算中使用的要素进行分组
其他	出现在表格规则末尾表示 otherwise 子句的术语
是	在简短语句与完整 ***attribute text*** 之间的图例项中使用的术语

逻辑函数(English)

语法	说明
这不是真的：<*expr*>	***attribute*** 具有假值时用来返回真值的运算符
<*var*> **是确定的** **这是确定的：**<*expr*>	当 ***attribute*** 的值不是 ***uncertain*** 时，用来返回真值的运算符
<*var*> **是未确定的** <*var*> **是不确定的** **这是不确定的：**<*expr*> **这是没确定的：**<*expr*> **这是未确定的：**<*expr*>	***attribute*** 值 ***uncertain*** 时用来返回真值的运算符
<*var*> **是已知的** **这是已知的：**<*expr*>	***attribute*** 具有任意值时用来返回真值的运算符

语法	说明
<var> 是未知的 这是未知的：*<expr>*	***attribute*** 没有值时用来返回真值的运算符

逻辑常数(English)

语法	说明
真的	用于表规则的常数真值。
假的	用于表规则的常数假值。
不确定的	用于表规则的常数 ***uncertain*** 值。

比较运算符(English)

语法	说明
*<lhs>***<***<rhs>* *<lhs>* 小于 *<rhs>* *<lhs>* **小于或者等于** *<rhs>* *<lhs>* 早于 *<rhs>*	小于 注：此运算符与数字值和货币值一起使用时没有任何自然语言形式。
<lhs> **>** *<rhs>* *<lhs>*大于*<rhs>* *<lhs>* **大于或者等于** *<rhs>* *<lhs>* 晚于 *<rhs>*	大于 注：此运算符与数字值和货币值一起使用时没有任何自然语言形式。
*<lhs>***<=***<rhs>*	小于或等于
<lhs> **>=** *<rhs>*	大于或等于
*<lhs>***=***<rhs>* *<lhs>*等于*<rhs>* *<lhs>* 等同 *<rhs>*	等于
<lhs> 不等于 *<rhs>* *<lhs>* **<>** *<rhs>*	不等于

数值函数(English)

语法	说明
数字(*<numText>*)	将指定字符串转换为数字值
<x> **+** *<y>*	数学加法
<x> **-** *<y>*	数学减法

语法	说明
<lhs> ***** *<rhs>*	数学乘法
<lhs> **/** *<rhs>*	数学除法
<lhs> **** *<rhs>*	整除
<lhs> **modulo** *<rhs>*	整除后的余数
最大值(*<x>*, *<y>*) **最大值**(*<date/time/datetime1>*, *<date/time/datetime2>*)	返回两个值中的较大者
最小值(*<x>*, *<y>*) **最小值**(*<date/time/datetime1>*, *<date/time/datetime2>*)	返回两个值中的较小者
取指数(*<x>*, *<y>*)	x 的 y 次幂
e指数(*<x>*)	常量 e 的 x 次幂
绝对值(*<x>*) **|***<val>***|**	x 的绝对值
自然对数(*<x>*)	x 的自然对数值
常用对数(*<x>*)	以 10 为底数的 x 的对数值
平方根(*<x>*)	x 的平方根
四舍五入(*<x>*, *<n>*)	将 x 精确到 n 位小数
截取值(*<x>*, *<n>*)	截取 x 到 n 位小数
取正弦函数值(*<x>*)	x 的正弦值
取余弦函数值(*<x>*)	x 的余弦值
取正切函数值(*<x>*)	x 的正切值
取反正弦函数值(*<x>*)	x 的反正弦值
取反余弦函数值(*<x>*)	x 的反余弦值
取反正切函数值(*<x>*)	x 的反正切值

日期函数(English)

语法	说明
当前日期()	在会话的开头返回当前日期 ***date***。
日期(*<text>*)	将指定字符串转换为 ***date*** 值
构建日期(*<year>*, *<month>*,	从指定年、月、日中返回 ***date***。

语法	说明
<day>)	
取具体日子 (*<date/datetime>*)	返回 ***date/datetime attribute*** 的日组成部分。
取月份(*<date/datetime>*)	返回 ***date/datetime attribute*** 的月组成部分。
取年份(*<date/datetime>*)	返回 ***date/datetime attribute*** 的年组成部分。
特定星期的另一天(*<date/datetime>*, *<day>*) **下个星期一(** *<from-date>* **)** **上个星期一(** *<from-date>* **)** **下个星期二(** *<from-date>* **)** **上个星期二(** *<from-date>* **)** **下个星期三(** *<from-date>* **)** **上个星期三(** *<from-date>* **)** **下个星期四(** *<from-date>* **)** **上个星期四(** *<from-date>* **)** **下个星期五(** *<from-date>* **)** **上个星期五(** *<from-date>* **)** **下个星期六(** *<from-date>* **)** **上个星期六(** *<from-date>* **)** **下个星期日(** *<from-date>* **)** **上个星期日(** *<from-date>* **)**	返回某一 ***date*** 或之前/之后下一工作日的 ***date***(取决于所使用的语法)。
另一天(*<date>*, *<day>*, *<month>*) **上个UK税收年(** *<from-date>* **)** **下个UK税收年(** *<from-date>* **)**	返回某一 ***date*** 之后给定日和月的下一实例。
增加天数(*<date/datetime>*, *<num_days>*) **减少天数(***<date>* **,** *<num_days>* **)**	向 ***date*** 中添加或从中减去天数。使用简洁句法形式时，数字必须为正整数，以便向输入 ***date*** 中添加天数，或者必须为负数，以便从输入 ***date*** 中减去天数。
增加星期数 (*<date/datetime>*, *<num_weeks>*) **减少星期数(***<date>* **,** *<num_weeks>* **)**	向 ***date*** 中添加周数。使用简洁句法形式时，数字必须为正整数，以便向输入 ***date*** 中添加周数。
增加月份数 (*<date/datetime>*, *<num_months>*)	向 ***date*** 中添加月数。使用简洁句法形式时，数字必须为正整数，以便向输入 ***date*** 中添加月数。

语法	说明
减少月份数**(**<*date*> **,** <*num_months*> **)**	
增加年数(<*date/datetime*>, <*num_years*>) 减少年数**(**<*date*> **,** <*num_years*> **)**	向 ***date*** 中添加年数。使用简洁句法形式时，数字必须为正整数，以便向输入 ***date*** 中添加年数。
相隔工作日(<*date1*>, <*date2*>)	计算 ***date***1 与 ***date***2 之间的工作日数。即星期一到星期五之间的天数。 注：包含较早的 ***date***，但是不包含较晚的 ***date***。
年初(<*date/datetime*>)	返回某一 ***date*** 所在年中的第一个 ***date***。
年末(<*date/datetime*>)	返回某一 ***date*** 所在年中的最后一个 ***date***。
相隔天数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的完整天数。两个日期的顺序不会影响结果。
含两边的相隔天数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的完整天数(包含首尾时间)。计算结果包含首尾时间。首尾日期相同时，返回结果为 1。两个日期的顺序不会影响结果。
不含两边的相隔天数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的完整天数(不包含首尾时间)。计算结果不包含首尾时间。首尾日期相同时，返回结果为 0。两个日期的顺序不会影响结果。
相隔星期数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整周数。两个日期的顺序不会影响结果。
含两边的相隔星期数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整周数(包含首尾时间)。两个日期的顺序不会影响结果。
不含两边的相隔星期数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整周数(不包含首尾时间)。两个日期的顺序不会影响结果。
相隔月数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整月数。两个日期的顺序不会影响结果。
含两边的相隔月数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整月数(包含首尾时间)。两个日期的顺序不会影响结果。
不含两边的相隔月数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的已用整月数(不包含首尾时间)。两个日期的顺序不会影响结果。
相隔整年数(<*date/datetime1*>, <*date/datetime2*>)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的年数。两个日期的顺序不会影响结果。

语法	说明
晚的整年数(*<date2>***,** *<date1>***)**	
含两边的相隔年数(*<date/datetime1>*, *<date/datetime2>*)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的年数(包含首尾时间)。两个日期的顺序不会影响结果。
不含两边的相隔年数(*<date/datetime1>*, *<date/datetime2>*)	返回 ***date/datetime1*** 与 ***date/datetime2*** 之间的年数(不包含首尾时间)。两个日期的顺序不会影响结果。

当天的时间函数(English)

语法	说明
当前的实际时间(*<text>*)	将给定字符串转换为当天的时间
取秒(*<time/datetime>*)	返回 ***timeofday/datetime attribute*** 的秒钟组成部分。
取分钟(*<time/datetime>*)	返回 ***timeofday/datetime attribute*** 的分钟组成部分。
取小时(*<time/datetime>*)	返回 ***timeofday/datetime attribute*** 的小时组成部分。

日期和时间函数(English)

语法	说明
当前日期时间()	在会话的开头返回当前日期 ***date*** 和时间。
日期时间(*<text>*)	将指定字符串转换为 ***datetime*** 值
连接日期时间(*<date>*, *<time>*)	通过连接 ***date*** 与当天的时间，设置 ***date*** 时间。
相隔秒数(*<datetime1>*, *<datetime2>*) **相隔秒数**(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的秒钟数。
含两边的相隔秒数(*<datetime1>*, *<datetime2>*) **含两边的相隔秒数**(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的秒钟数(包含首尾时间)。
不含两边的相隔秒数(*<datetime1>*, *<datetime2>*) **不含两边的相隔秒数**(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的秒钟数(不包含首尾时间)。
相隔分钟数(*<datetime1>*, *<dat-*	返回 ***datetime1*** 与 ***datetime2*** 之间的分钟数。

语法	说明
etime2>) 相隔分钟数(*<timeOfDay1>*, *<timeOfDay2>*)	
含两边的相隔分钟数(*<datetime1>*, *<datetime2>*) 含两边的相隔分钟数(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的分钟数(包含首尾时间)。
不含两边的相隔分钟数(*<datetime1>*, *<datetime2>*) 不含两边的相隔分钟数(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的分钟数(不包含首尾时间)。
相隔小时数(*<datetime1>*, *<datetime2>*) 相隔小时数(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的小时数。
含两边的相隔小时数(*<datetime1>*, *<datetime2>*) 含两边的相隔小时数(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的小时数(包含首尾时间)。
不含两边的相隔小时数(*<datetime1>*, *<datetime2>*) 不含两边的相隔小时数(*<timeOfDay1>*, *<timeOfDay2>*)	返回 ***datetime1*** 与 ***datetime2*** 之间的小时数(不包含首尾时间)。
取日期(*<datetime>*)	从 ***datetime attribute*** 中取 ***date***。
取当前的实际时间(*<datetime>*)	从 ***datetime attribute*** 中取当天的时间。可用于将 ***timeofday attribute*** 的值设置为从当前 ***date*** 和时间中取时间时执行规则的时间。
增加小时数(*<datetime>*, *<num_hours>*) 增加小时数(*<timeOfDay>*, *<num_hours>*)	向 ***date*** 时间中添加小时数。
增加分钟数(*<datetime>*, *<num_minutes>*) 增加分钟数(*<timeOfDay>*, *<num_minutes>*)	向 ***date*** 时间中添加分钟数。
增加秒数(*<datetime>*, *<num_seconds>*) 增加秒数(*<timeOfDay>*, *<num_seconds>*)	向 ***date*** 时间中添加秒数。

文本函数(English)

语法	说明
<text1> **&** *<text2>*	结合使用 ***text1*** 与 ***text2*** 等以形成单个 ***text*** 值。 注：您可以使用任意类型的变量。这些值将使用规则会话中安装的格式化程序进行格式化。
	结合使用 ***text1*** 与 ***text2*** 等以形成单个 ***text*** 值。 注：您可以使用任意类型的变量。这些值将使用规则会话中安装的格式化程序进行格式化。
包含(*<text>*, *<substring>*)	返回布尔值，表示给定的 ***text*** 值是否包含给定的 ***text*** 子字符串。***text*** 比较不区分大小写。
结束日期(*<text>*, *<substring>*)	返回布尔值，表示给定的 ***text*** 值是否以给定的 ***text*** 子字符串结尾。***text*** 比较不区分大小写。
是数字(*<text>*)	返回布尔值，表示给定 ***text*** 值是否代表有效数字。
长度(*<text>*)	返回给定 ***text*** 值的字符长度。
开头(*<text>*, *<substring>*)	返回布尔值，表示给定的 ***text*** 值是否以给定的 ***text*** 子字符串开始。***text*** 比较不区分大小写。
子字符串(*<text>*, *<offset>*, *<length>*)	返回以给定偏移量(指定字符长度)开始的 ***text*** 的子字符串。如果到达字符串末尾，则返回较少字符。
文本(*<number>*) **文本**(*<date>*) **文本**(*<datetime>*) **文本**(*<timeOfDay>*)	将指定数字或 ***date attribute*** 转换为 ***text*** 值。

实体和关系函数(English)

语法	说明
满足条件(*<relationship>*, *<Exp>*)	只有一个条件时，用来在"一对一"、"多对一"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。
范围满足条件(*<relationship>*, *<alias>*) **范围满足条件**(*<relationship>*)	有一个或多个条件时，用来在"一对一"、"多对一"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。
所有都满足条件(*<relationship>*, *<Exp>*)	当需要确定目标 ***entity*** 组中的所有成员是否需要满足规则时，用于在"一对多"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。 当规则中只有一个条件时使用这种形式。
范围内所有都满足条件(*<relationship>*)	当需要确定目标 ***entity*** 组中的所有成员是否需要满足规则时，用于在"一对多"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。

语法	说明
范围内所有都满足条件(*<relationship>*, *<alias>*)	当规则中有一个或多个条件时使用这种形式。
存在一个满足条件(*<relationship>*, *<Exp>*)	当需要确定目标 ***entity*** 组中的任何成员是否需要满足规则时，用于在"一对多"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。 当规则中只有一个条件时使用这种形式。
范围内存在一个满足条件(*<relationship>*) **范围内存在一个满足条件**(*<relationship>*, *<alias>*)	当需要确定目标 ***entity*** 组中的任何成员是否需要满足规则时，用于在"一对多"或"多对多"***relationship*** 中从一个 ***entity*** 指向另一个 ***entity***。 当规则中有一个或多个条件时使用这种形式。
是集合的成员(*<target>*, *<relationship>*) **是集合的成员**(*<target>*, *<alias>*, *<relationship>*)	用作结论来推断一个 ***entity*** 实例是 ***relationship*** 的成员。用作条件来测试 ***entity*** 实例是 ***relationship***(其中另一个 ***entity*** 实例是源实例)的目标。
不是集合的成员(*<target>*, *<relationship>*)	用作条件来测试一个 ***entity*** 实例不是 ***relationship***(其中另一个 ***entity*** 实例是源实例)的目标。
实例总数(*<relationship>*)	计算 ***entity*** 的实例数量。
满足条件的实例总数(*<relationship>*, *<Exp>*)	计算特定 ***entity-level attribute*** 具有特定值的 ***entity*** 的实例数量。
实例最大值(*<relationship>*, *<number-attr>*) **实例最大值**(*<relationship>*, *<date-attr>*) **实例最大值**(*<relationship>*, *<datetime-attr>*) **实例最大值**(*<relationship>*, *<time-attr>*)	获取该 ***entity*** 的所有实例中 ***entity-level*** 变量的最高/最新值。
满足条件的实例最大值(*<relationship>*, *<number-attr>*, *<condition>*) **满足条件的实例最大值**(*<relationship>*, *<date-attr>*, *<condition>*) **满足条件的实例最大值**(*<relationship>*, *<datetime-attr>*, *<condition>*) **满足条件的实例最大值**(*<relationship>*, *<time-attr>*, *<condition>*)	获取特定 ***entity-level attribute*** 具有特定值的 ***entity*** 的所有实例中 ***entity-level*** 变量的最高/最新值。
实例最小值(*<relationship>*, *<number-attr>*)	获取该 ***entity*** 的所有实例中 ***entity-level*** 变量的最低/最旧值。

语法	说明
实例最小值(*<relationship>*, *<date-attr>*) 实例最小值(*<relationship>*, *<datetime-attr>*) 实例最小值(*<relationship>*, *<time-attr>*)	
满足条件的实例最小值(*<relationship>*, *<number-attr>*, *<condition>*) 满足条件的实例最小值(*<relationship>*, *<date-attr>*, *<condition>*) 满足条件的实例最小值(*<relationship>*, *<datetime-attr>*, *<condition>*) 满足条件的实例最小值(*<relationship>*, *<time-attr>*, *<condition>*)	获取特定 ***entity-level attribute*** 具有特定值的 ***entity*** 的所有实例中 ***entity-level*** 变量的最低/最旧值。
实例总和(*<relationship>*, *<number-attr>*)	获取 ***entity-level*** 变量的所有实例总数。
满足条件的实例总和(*<relationship>*, *<number-attr>*, *<condition>*)	获取 ***entity-level*** 变量(特定 ***entity-level*** 布尔 ***attribute*** 为真的 ***entity*** 为真)的所有实例总数。
满足条件的实例值(*<relationship>*, *<number-attr>*, *<condition>*) 满足条件的实例值(*<relationship>*, *<text-attr>*, *<condition>*) 满足条件的实例值(*<relationship>*, *<date-attr>*, *<condition>*) 满足条件的实例值(*<relationship>*, *<datetime-attr>*, *<condition>*) 满足条件的实例值(*<relationship>*, *<time-attr>*, *<condition>*)	按照条件在 ***relationship*** 的目标 ***entity*** 实例中查找，获取唯一 ***entity*** 实例的值。 • 如果按条件找到单个目标 ***entity*** 实例，则值根据该 ***entity*** 实例进行计算。 • 如果多个目标实例均符合条件，则返回 ***uncertain***。 • 如果没有目标实例符合条件，则 ***relationship*** 已知，且值是 ***uncertain***。
实例等于(*<instance1>*, *<instance2>*)	确定 ***entity*** 的两个实例是否为同一实例。
实例不等于(*<instance1>*, *<instance2>*)	确定 ***entity*** 的两个实例是否是同一实例。
推断实例(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)**存在	用作结论来推断一个 ***entity*** 实例存在且是 ***relationship*** 的成员。

时间推理函数(English)

语法	说明
时间间隔内变量的唯一值总数(*<start-date>*, *<end-date>*, *<variable>*) **时间间隔内变量的唯一值总数**(*<start-date>*, *<end-date>*, *<condition>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间变量的已知不同值的数量。
满足条件的时间间隔内变量的唯一值总数(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间且布尔筛选条件为真时变量的已知不同值的数量。
时间间隔内变量值的每日总和(*<start-date>*, *<end-date>*, *<number-attr>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间货币或数值变量的总和。***attribute*** 假定为每日数量。
满足条件的时间间隔内变量值的每日总和(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间且仅在条件为真时，货币或数值变量的所有每日值的总和。
时间间隔内变量的最大值(*<start-date>*, *<end-date>*, *<number-attr>*) **时间间隔内变量的最大值**(*<start-date>*, *<end-date>*, *<date-attr>*) **时间间隔内变量的最大值**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **时间间隔内变量的最大值**(*<start-date>*, *<end-date>*, *<time-attr>*)	选择在开始 ***date***(包含)与结束 ***date***(不包含)之间变量的最大值。
满足条件的时间间隔内变量的最大值(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **满足条件的时间间隔内变量的最大值**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **满足条件的时间间隔内变量的最大值**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*)	选择在开始 ***date***(包含)与结束 ***date***(不包含)之间且条件为真时变量的最大值。

语法	说明
满足条件的时间间隔内变量的最大值(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	
时间间隔内变量的最小值(*<start-date>*, *<end-date>*, *<number-attr>*) **时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<date-attr>*) **时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<time-attr>*)	选择在开始 ***date***(包含)与结束 ***date***(不包含)之间变量的最小值。
满足条件的时间间隔内变量的最小值(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **满足条件的时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **满足条件的时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **满足条件的时间间隔内变量的最小值**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	选择在开始 ***date***(包含)与结束 ***date***(不包含)之间且条件为真时变量的最小值。
时间间隔内变量值的加权平均数(*<start-date>*, *<end-date>*, *<number-attribute>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间，按照各个值所对应的时间范围进行加权的货币或数值变量的平均值。
满足条件的时间间隔内变量值的加权平均数(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	计算在开始 ***date***(包含)与结束 ***date***(不包含)之间且布尔条件为真(按照各个值所对应的时间范围进行加权并且筛选条件为真)时，货币或数值变量的平均值。
时间间隔内始终满足条件(*<start-date>*, *<end-date>*,	当且仅当布尔条件在开始 ***date***(包含)与结束 ***date***(不包含)之间始终为真时，返回真值。

语法	说明
<condition>)	
时间间隔内至少数天满足条件(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	当且仅当布尔条件在开始 ***date***(包含)与结束 ***date***(不包含)之间至少指定的天数(不必连续)内为真时，返回真值。
时间间隔内连续数天满足条件(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	当且仅当布尔条件在开始 ***date***(包含)与结束 ***date***(不包含)之间至少给定的连续天数内为真时，返回真值。
时间间隔内有时满足条件(*<start-date>*, *<end-date>*, *<condition>*)	当且仅当布尔条件在开始 ***date***(包含)与结束 ***date***(不包含)之间始终为真时，返回真值。
取特定日期的值(*<date>*, *<value>*)	返回给定 ***attribute*** 在指定 ***date*** 的值。
在此之前满足条件的日期(*<date>*, *<condition>*)	返回从指定 ***date***(包含该日期)起倒推，布尔条件为真的最后一个 ***date***。
在此之后满足条件的日期(*<date>*, *<condition>*)	返回从指定 ***date***(包含该日期)起前推，布尔条件将会为真的下一个 ***date***。
最晚日期()	返回最晚可能 ***date*** 对应的 ***date*** 值 - 即保证晚于 ***date attribute*** 可能具有或表达式求值结果中可能包含的其它任何 ***date*** 的一个 ***date***。
最早日期()	返回最早可能 ***date*** 对应的 ***date*** 值 - 即保证早于 ***date attribute*** 可能具有或表达式求值结果中可能包含的其它任何 ***date*** 的一个 ***date***。
间隔累计天数(*<date>*, *<end-date>*)	返回每天各异的数值变量，该变量代表自该 ***date*** 起的完整天数。
间隔累计星期数(*<date>*, *<end-date>*)	返回每周各异的数值变量，该变量代表自该 ***date*** 起的完整周数。
间隔累计月数(*<date>*, *<end-date>*)	返回每月各异的数值变量，该变量代表自该 ***date*** 起的完整月数。注：如果提供的 ***date*** 在当月 28 日之后，并且下个月的天数比该月天数少，则将会在下个月的最后一天创建周年月份变更点。例如，如果提供的 ***date*** 为 2007 年 1 月 28、29、30 或 31 日，则第一个变更点即为 2007 年 2 月 28 日。
间隔累计年数(*<date>*, *<end-date>*)	返回每年各异的数值变量，该变量代表自该 ***date*** 起的完整年数。
连续数天始终满足条件(*<days>*, *<condition>*)	返回随时间变化，当且仅当布尔条件在之前指定的天数(不包括当天)内始终为真时才为真的布尔 ***attribute***。
至少连续数天满足条件(*<minDays>*, *<days>*, *<condition>*)	返回随时间变化，当且仅当布尔条件在之前设定天数(不包括当前天)中任意时间段内连续为真的天数至少达到最少天数时才为真的布尔 ***attribute***。

语法	说明
累计数天满足条件(*<days>*, *<condition>*)	返回随时间变化，当且仅当布尔条件在之前指定的天数(不包含当天)内始终为真时才为真的布尔 ***attribute***。
本日期以后(*<date>*)	返回随时间变化且在某一 ***date*** 之后为真，当天及之前为假的布尔 ***attribute***。
本日期以前(*<date>*)	返回随时间变化且在某一 ***date*** 之前为真，当天及之后为假的布尔 ***attribute***。
本日期当天(*<date>*)	返回随时间变化且在某一 ***date*** 为真，之前和之后均为假的布尔 ***attribute***。
本日期当天或者以后(*<date>*)	返回随时间变化且在某一 ***date*** 或之后为真，之前为假的布尔 ***attribute***。
本日期当天或者以前(*<date>*)	返回随时间变化且在某一 ***date*** 或之前为真，之后为假的布尔 ***attribute***。
依据开始日期取值(*<relationship>*, *<date>*, *<value>*)	从其值从开始 ***date attribute*** 起有效的实体的 ***relationship*** 和值 ***attribute*** 中返回单一临时 ***attribute***(在源 ***entity*** 层中)。
依据结束日期取值(*<relationship>*, *<date>*, *<value>*)	从其值在结束 ***date attribute*** 之前均有效的实体的 ***relationship*** 和值 ***attribute*** 中返回单一临时 ***attribute***(在源 ***entity*** 层中)。
依据日期区间取值(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	从其值在开始 ***date attribute***(包含)和结束 ***date attribute***(不包含)之间均有效的实体的 ***relationship*** 和值 ***attribute*** 中返回单一临时 ***attribute***(在源 ***entity*** 层中)。如果值在下一开始 ***date*** 前过期，则该值是 ***uncertain***。
临时工作日(*<startdate>*, *<enddate>*)	日期是指定开始 ***date***(包含)与结束 ***date***(不包含)之间的工作日时返回真值，日期是周末时返回假值。返回 ***date*** 范围外的 ***uncertain***。
临时每月一次(*<startdate>*, *<enddate>*, *<dayofmonth>*)	日等于月份日期参数时返回真值，为从指定开始 ***date***(包含)与结束 ***date***(不包含)之间的月份所有其它日时返回假值。返回 ***date*** 范围外的 ***uncertain***。月份日期超出当月天数时，该值在该月的最后一天为真值，因此函数返回的值恰好是每月有一天为真。

验证事件功能(English)

语法	说明
错误(*<text>*)	使用错误事件将消息传递给用户，防止继续调查到触发该错误的条件不再适用为止。
警告(*<text>*)	使用警告事件将消息传递给用户，但允许继续进行，而不管触发该警告的条件。

已过时的函数(English)

语法	说明
调用客户化函数(*<A>*, *<B>*)	返回代码库的外部调用结果。必须向 Determinations 引擎提供代码库，自定义函数调用才会成功。

邏輯連接器(English)

語法	描述
如果	可在結論行結尾(具有下列證明)顯示的選擇性詞彙
和	兩個 ***attributes*** 之間的邏輯連接詞
或	兩個 ***attributes*** 之間的邏輯反意連接詞
二者之一 其中之一 任何 以下至少一項為真 符合以下任何一項	使用反意連接詞的群組要素，其中兩個或更多的 ***attributes*** 需要加以群組
二者皆是 所有皆是 以下所有項目皆真 已符合以下所有項目	使用連接詞的群組要素，其中兩個或更多的 ***attributes*** 需要加以群組
否則	在表格規則結尾顯示的詞彙，表示「否則」子句
是	圖例項目中使用的詞彙，介於縮短的片語與完整 ***attribute text*** 之間

邏輯函數(English)

語法	描述
非真 *<expr>*	如果 ***attribute*** 的值為 false，用來傳回 true 的運算子
<var> 確定 已確定是否 *<expr>*	如果 ***attribute*** 的值不是 ***uncertain***，用來傳回 true 的運算子
<var> 不確定 不確定 *<expr>* 不確定是否 *<expr>* 不確定 *<expr>* 不確定	如果 ***attribute*** 值為 ***uncertain***，用來傳回 true 的運算子
<var> 已知 已知是否 *<expr>*	如果 ***attribute*** 有任何值，用來傳回 true 的運算子
<var> 不明 不知道是否 *<expr>* 不知道	如果 ***attribute*** 沒有值，用來傳回 true 的運算子

邏輯常數(English)

語法	描述
真	用於表格規則的常數 true 值。
假	用於表格規則的常數 false 值。
不確定	用於表格規則的常數 ***uncertain*** 值。

比較運算子(English)

語法	描述
*<x>***<***<y>*	小於 備註:此運算子與數值及幣別值一起使用時,則不具有自然語言形式。
<x> **>** *<y>*	大於 備註:此運算子與數值及幣別值一起使用時,則不具有自然語言形式。
*<x>***<=***<y>*	小於或等於
<x> **>=** *<y>*	大於或等於
*<x>***=***<y>*	等於
<x> **<>** *<y>*	不等於

數值函數(English)

語法	描述
數字(*<numText>*)	將指定的字串轉換成數值
<x> **+** *<y>*	數學加法
<x> **-** *<y>*	數學減法
<x> * *<y>*	數學乘法
<x> **/** *<y>*	數學除法
<x> **** *<y>*	整數除法
<x> **modulo** *<y>*	整數除法之後餘數
最大(*<x>*, *<y>*) **最大**(*<date/time/datetime1>*, *<date/time/datetime2>*)	傳回兩個數值的較大值
最小(*<x>*, *<y>*) **最小**(*<date/time/datetime1>*, *<date/time/datetime2>*)	傳回兩個數值的較小值
X 的 Y 次方(*<x>*, *<y>*)	x 至 y 的次方

語法	描述
指數(<*x*>)	常數 e 對 x 的次方
絕對值(<*x*>) |<*val*>|	x 的絕對值
自然對數(<*x*>)	x 的自然對數
對數(<*x*>)	x 的對數底 10
平方根(<*x*>)	x 的平方根
四捨五入(<*x*>, <*n*>)	將 x 四捨五入至 n 個小數位
截取值(<*x*>, <*n*>)	將 x 縮短至 n 個小數位
正弦(<*x*>)	x 的正弦
餘弦(<*x*>)	x 的餘弦
正切(<*x*>)	x 的正切
反正弦(<*x*>)	x 的反正弦
反餘弦(<*x*>)	x 的反餘弦
反正切(<*x*>)	x 的反正切

日期函數(English)

語法	描述
目前日期()	在階段作業開始傳回目前的 ***date***。
日期(<*text*>)	將指定的字串轉換成 ***date*** 值
建立日期(<*year*>, <*month*>, <*day*>)	傳回從指定年度、月份及該日所構成的 ***date***。
擷取日(<*date/datetime*>)	傳回 ***date/datetime attribute*** 的日元件。
擷取月(<*date/datetime*>)	傳回 ***date/datetime attribute*** 的月份元件。
擷取年(<*date/datetime*>)	傳回 ***date/datetime attribute*** 的年份元件。
當週隔日(<*date/datetime*>, <*day*>)	傳回 ***date*** 當日或之前/之後下一個平日的 ***date***(視使用的語法而定)。
隔日(<*date*>, <*day*>, <*month*>)	傳回 ***date*** 之後下一個指定日期與月份的實例。
新增日(<*date/datetime*>, <*num_days*>)	增加/減去 ***date*** 的天數。當使用簡潔的語法形式時，數字必須為正整數才能將天數加到輸入值 ***date*** 中，或為負數才能從輸入值 ***date*** 中減去天數。

語法	描述
新增週(*<date/datetime>*, *<num_weeks>*)	將週數加到 ***date*** 中。當使用簡潔的語法形式時，數字必須為正整數才能將週數加到輸入值 ***date*** 中。
新增月(*<date/datetime>*, *<num_months>*)	將月份數加到 ***date*** 中。當使用簡潔的語法形式時，數字必須為正整數才能將月份數加到輸入值 ***date*** 中。
新增年(*<date/datetime>*, *<num_years>*)	將年數加到 ***date*** 中。當使用簡潔的語法形式時，數字必須為正整數才能將年數加到輸入值 ***date*** 中。
平日計次(*<date1>*, *<date2>*)	計算介於 ***date***1 與 ***date***2 之間的平日天數。亦即，介於星期一與星期五之間的天數。備註：較早的 ***date*** 包含在內，而較晚的 ***date*** 則不含在內。
年初(*<date/datetime>*)	傳回一年中 ***date*** 所在的第一個 ***date***。
年終(*<date/datetime>*)	傳回一年中 ***date*** 所在的最後一個 ***date***。
日差異(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的整日天數。兩個日期的順序不會影響結果。
日差異包含在內(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的整日(含)天數。此計算法會包含兩個端點。當日期相同時，結果為 1。兩個日期的順序不會影響結果。
日差異排除在外(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的整日(不含)天數。此計算法會排除兩個端點。當日期相同時，結果為 0。兩個日期的順序不會影響結果。
週差異(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整週數目。兩個日期的順序不會影響結果。
週差異包含在內(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整週數目(含)。兩個日期的順序不會影響結果。
週差異排除在外(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整週數目(不含)。兩個日期的順序不會影響結果。
月差異(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整月數目。兩個日期的順序不會影響結果。
月差異包含在內(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整月(含)數目。兩個日期的順序不會影響結果。
月差異排除在外(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間經過的整月(不含)數目。兩個日期的順序不會影響結果。
年差異(*<date/datetime1>*,	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的年數。兩個日期的順序不會影

語法	描述
<date/datetime2>)	響結果。
年差異包含在內(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的年數(含)。兩個日期的順序不會影響結果。
年差異排除在外(*<date/datetime1>*, *<date/datetime2>*)	傳回介於 ***date/datetime1*** 與 ***date/datetime2*** 之間的年數(不含)。兩個日期的順序不會影響結果。

當日時間函數(English)

語法	描述
當日時間(*<text>*)	將指定字串轉換成一天的時間
擷取秒(*<time/datetime>*)	傳回 ***timeofday/datetime attribute*** 的秒鐘元件。
擷取分鐘(*<time/datetime>*)	傳回 ***timeofday/datetime attribute*** 的分鐘元件。
擷取小時(*<time/datetime>*)	傳回 ***timeofday/datetime attribute*** 的小時元件。

日期與時間函數(English)

語法	描述
目前日期時間()	在階段作業開始傳回目前的 ***date*** 與時間。
日期時間(*<text>*)	將指定的字串轉換成 ***datetime*** 值
連結日期時間(*<date>*, *<time>*)	透過結合 ***date*** 與當日時間來設定 ***date*** 時間。
秒差異(*<datetime1>*, *<datetime2>*) **秒差異**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的秒數。
秒差異包含在內(*<datetime1>*, *<datetime2>*) **秒差異包含在內**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的秒數(含)。
秒差異排除在外(*<datetime1>*, *<datetime2>*) **秒差異排除在外**	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的秒數(不含)。

語法	描述
(*<timeOfDay1>*, *<timeOfDay2>*)	
分鐘差異(*<datetime1>*, *<datetime2>*) **分鐘差異**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的分鐘數。
分鐘差異包含在內(*<datetime1>*, *<datetime2>*) **分鐘差異包含在內**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的分鐘數(含)。
分鐘差異排除在外(*<datetime1>*, *<datetime2>*) **分鐘差異排除在外**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的分鐘數(不含)。
小時差異(*<datetime1>*, *<datetime2>*) **小時差異**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的小時數。
小時差異包含在內(*<datetime1>*, *<datetime2>*) **小時差異包含在內**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的小時數(含)。
小時差異排除在外(*<datetime1>*, *<datetime2>*) **小時差異排除在外**(*<timeOfDay1>*, *<timeOfDay2>*)	傳回介於 ***datetime1*** 與 ***datetime2*** 之間的小時數(不含)。
擷取日期(*<datetime>*)	從 ***datetime attribute*** 摘錄 ***date***。
擷取當日時間(*<datetime>*)	從 ***datetime attribute*** 摘錄當日時間。可用來將 ***timeofday attribute*** 的值設為規則執行的時間(透過從目前的 ***date*** 與時間摘錄時間資訊)。
新增小時(*<datetime>*, *<num_hours>*) **新增小時**(*<timeOfDay>*, *<num_hours>*)	將小時數加到 ***date*** 時間中。
新增分鐘(*<datetime>*,	將分鐘數加到 ***date*** 時間中。

語法	描述
<num_minutes>) **新增分鐘**(*<timeOfDay>*, *<num_minutes>*)	
新增秒(*<datetime>*, *<num_seconds>*) **新增秒**(*<timeOfDay>*, *<num_seconds>*)	將秒數加到 ***date*** 時間中。

文字函數(English)

語法	描述
<text1> **&** *<text2>*	將 ***text1*** 與 ***text2*** 等各個值合併，以組成單一的 ***text*** 值。 備註：您可以使用任何類型的變數。使用在規則階段作業中安裝的格式化程式，可進行值的格式化。
	將 ***text1*** 與 ***text2*** 等各個值合併，以組成單一的 ***text*** 值。 備註：您可以使用任何類型的變數。使用在規則階段作業中安裝的格式化程式，可進行值的格式化。
包含(*<text>*, *<substring>*)	傳回表示指定 ***text*** 值是否包含指定 ***text*** 子字串的布林值。***text*** 比較區分大小寫。
結尾為(*<text>*, *<substring>*)	傳回表示指定 ***text*** 值是否以指定 ***text*** 子字串結束的布林值。***text*** 比較區分大小寫。
是數字(*<text>*)	傳回表示指定 ***text*** 值是否代表有效數字的布林值。
長度(*<text>*)	傳回指定 ***text*** 值的字元長度。
開頭為(*<text>*, *<substring>*)	傳回表示指定 ***text*** 值是否以指定 ***text*** 子字串開始的布林值。***text*** 比較區分大小寫。
子字串(*<text>*, *<offset>*, *<length>*)	傳回以指定位移(其為以字元數計的指定長度)開始的 ***text*** 子字串。如果達到字串的結尾，便會傳回較少的字元數。
文字(*<number>*) **文字**(*<date>*) **文字**(*<datetime>*) **文字**(*<timeOfDay>*)	將指定的數字或 ***date attribute*** 轉換成 ***text*** 值。

實體與關係函數(English)

語法	描述
針對(*<relationship>*, *<Exp>*)	用來從一個 ***entity*** 參考至「一對一」、「多對一」或「多對多」***relationship*** 中(其中

語法	描述
	只有一個條件)的另一個 ***entity***。
針對範圍(*<relationship>*, *<alias>*) **針對範圍**(*<relationship>*)	用來從一個 ***entity*** 參考至「一對一」、「多對一」或「多對多」***relationship*** 中(其中有一或多個條件)的另一個 ***entity***。
針對所有(*<relationship>*, *<Exp>*)	用來從一個 ***entity*** 參考至「一對多」或「多對多」***relationship*** 中的另一個 ***entity*** (當您需要決定目標 ***entity*** 群組的所有成員是否需要滿足規則時)。 此形式是在規則中只有一個條件時使用。
針對全部範圍(*<relationship>*) **針對全部範圍**(*<relationship>*, *<alias>*)	用來從一個 ***entity*** 參考至「一對多」或「多對多」***relationship*** 中的另一個 ***entity*** (當您需要決定目標 ***entity*** 群組的所有成員是否需要滿足規則時)。 此形式是在規則中有一或多個條件時使用。
存在(*<relationship>*, *<Exp>*)	用來從一個 ***entity*** 參考至「一對多」或「多對多」***relationship*** 中的另一個 ***entity*** (當您需要決定目標 ***entity*** 群組的成員是否需要滿足規則時)。 此形式是在規則中只有一個條件時使用。
存在範圍(*<relationship>*) **存在範圍**(*<relationship>*, *<alias>*)	用來從一個 ***entity*** 參考至「一對多」或「多對多」***relationship*** 中的另一個 ***entity*** (當您需要決定目標 ***entity*** 群組的成員是否需要滿足規則時)。 此形式是在規則中有一或多個條件時使用。
是以下單位的成員(*<target>*, *<relationship>*) **是以下單位的成員**(*<target>*, *<alias>*, *<relationship>*)	用作結論以推論 ***entity*** 實例為 ***relationship*** 的成員。用作條件以測試 ***entity*** 實例是否為 ***relationship*** 的目標，而第二個 ***entity*** 實例為其來源。
不是以下單位的成員(*<target>*, *<relationship>*)	用作條件以測試 ***entity*** 實例是否不是 ***relationship*** 的目標，而第二個 ***entity*** 實例為其來源。
實例計次(*<relationship>*)	計算 ***entity*** 已存在的實例數。
實例計次如果(*<relationship>*, *<Exp>*)	計算 ***entity*** 已有的實例數，其特定 ***entity-level attribute*** 已有特定值。
實例最大(*<relationship>*, *<number-attr>*) **實例最大**(*<relationship>*, *<date-attr>*) **實例最大**(*<relationship>*, *<datetime-attr>*) **實例最大**(*<relationship>*, *<time-attr>*)	針對 ***entity*** 的所有實例，取得 ***entity-level*** 變數的最高/最近值。
實例最大如果(*<relationship>*, *<number-attr>*, *<condition>*) **實例最大如果**(*<relationship>*, *<date-attr>*, *<condition>*)	針對 ***entity*** 的所有實例(其特定 ***entity-level attribute*** 具有特定值)，取得 ***entity-level*** 變數的最高/最近值。

語法	描述
實例最大如果(*<relationship>*, *<datetime-attr>*, *<condition>*) **實例最大如果**(*<relationship>*, *<time-attr>*, *<condition>*)	
實例最小(*<relationship>*, *<number-attr>*) **實例最小**(*<relationship>*, *<date-attr>*) **實例最小**(*<relationship>*, *<datetime-attr>*) **實例最小**(*<relationship>*, *<time-attr>*)	針對 ***entity*** 的所有實例，取得 ***entity-level*** 變數的最低/最早值。
實例最小如果(*<relationship>*, *<number-attr>*, *<condition>*) **實例最小如果**(*<relationship>*, *<date-attr>*, *<condition>*) **實例最小如果**(*<relationship>*, *<datetime-attr>*, *<condition>*) **實例最小如果**(*<relationship>*, *<time-attr>*, *<condition>*)	針對 ***entity*** 的所有實例(其特定 ***entity-level attribute*** 具有特定值)，取得 ***entity-level*** 變數的最低/最早值。
實例總和(*<relationship>*, *<number-attr>*)	取得 ***entity-level*** 變數的所有實例總和。
實例總和如果(*<relationship>*, *<number-attr>*, *<condition>*)	取得 ***entity-level*** 變數的所有實例總和(其中 ***entity*** 所述，特定 ***entity-level*** 布林 ***attribute*** 為 true)。
實例值如果(*<relationship>*, *<number-attr>*, *<condition>*) **實例值如果**(*<relationship>*, *<text-attr>*, *<condition>*) **實例值如果**(*<relationship>*, *<date-attr>*, *<condition>*) **實例值如果**(*<relationship>*, *<datetime-attr>*, *<condition>*) **實例值如果**(*<relationship>*, *<time-attr>*, *<condition>*)	從唯一 ***entity*** 實例取得數值，該實例已透過條件從 ***relationship*** 的目標 ***entity*** 實例加以識別。 • 如果條件會識別單一目標 ***entity*** 實例，則數值為依據該 ***entity*** 實例計算的數值。 • 如果超過一個目標實例符合條件，則會傳回 ***uncertain***。 • 如果沒有目標實例符合條件，且 ***relationship*** 已知，則數值為 ***uncertain***。
實例等於(*<instance1>*, *<instance2>*)	決定 ***entity*** 的兩個實例是否為相同實例。
實例不等於(*<instance1>*, *<instance2>*)	決定 ***entity*** 的兩個實例是否不為相同實例。
推論實例(*<relationship>*, *<iden-*	用作結論以推論 ***entity*** 實例存在且為 ***relationship*** 的成員。

語法	描述
tity>) <*rel*>**(**<*identity*>**)**存在	

暫時推斷函數(English)

語法	描述
區間計次差別(<*start-date*>, <*end-date*>, <*variable*>) **區間計次差別**(<*start-date*>, <*end-date*>, <*condition*>)	計算變數於開始 ***date***(含)到結束 ***date***(不含)的間隔期間，其已知特定值的數目。
區間計次差別如果(<*start-date*>, <*end-date*>, <*variable*>, <*condition*>)	計算變數於開始 ***date***(含)到結束 ***date***(不含)的間隔期間，僅包含布林篩選為 true 時，其已知特定值的數目。
區間日總和(<*start-date*>, <*end-date*>, <*number-attr*>)	計算幣別或數字變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的總和。***attribute*** 會假定為每日數量。
區間日總和如果(<*start-date*>, <*end-date*>, <*number-attr*>, <*condition*>)	計算幣別或數字變數於開始 ***date***(含)到結束 ***date***(不含)的間隔期間，僅包含條件為 true 時，所有每日數值的總和。
區間最大(<*start-date*>, <*end-date*>, <*number-attr*>) **區間最大**(<*start-date*>, <*end-date*>, <*date-attr*>) **區間最大**(<*start-date*>, <*end-date*>, <*datetime-attr*>) **區間最大**(<*start-date*>, <*end-date*>, <*time-attr*>)	選取變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的最大值。
區間最大如果(<*start-date*>, <*end-date*>, <*number-attr*>, <*condition*>)	選取變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的最大值(僅包含條件為 true 時)。

語法	描述
區間最大如果(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **區間最大如果**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **區間最大如果**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	
區間最小(*<start-date>*, *<end-date>*, *<number-attr>*) **區間最小**(*<start-date>*, *<end-date>*, *<date-attr>*) **區間最小**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **區間最小**(*<start-date>*, *<end-date>*, *<time-attr>*)	選取變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的最小值。
區間最小如果(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **區間最小如果**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **區間最小如果**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **區間最小如果**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	選取變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的最小值(僅包含條件為 true 時)。
區間加權平均(*<start-date>*, *<end-date>*,	計算幣別或數字變數於開始 ***date***(含)到結束 ***date***(不含)之間隔期間的平均值(依每個值套用的時間範圍加權)。

語法	描述
<number-attribute>)	
區間加權平均如果(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	計算幣別或數字變數於開始***date***(含)到結束***date***(不含)的間隔期間、僅包含布林條件為true時的平均值(依每個值套用的時間範圍加權且在篩選為true時)。
區間內一律(*<start-date>*, *<end-date>*, *<condition>*)	傳回true，如果且只有在布林條件於開始***date***(含)到結束***date***(不含)的間隔期間皆為true時。
區間內至少日數日(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	傳回true，如果且只有在布林條件於開始***date***(含)到結束***date***(不含)的間隔期間，至少於指定的天數(未必為連續)為true時。
區間內連續數日(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	傳回true，如果且只有在布林條件於開始***date***(含)到結束***date***(不含)的間隔期間，至少於指定的連續天數為true時。
區間內有時候(*<start-date>*, *<end-date>*, *<condition>*)	如果且只有在布林條件於開始***date***(含)到結束***date***(不含)之間隔期間為true時，傳回true。
價值於(*<date>*, *<value>*)	於指定的***date***傳回指定***attribute***的值。
當上一個(*<date>*, *<condition>*)	傳回布林條件上次為true的***date***，自指定的***date***起(含)往後推。
當下一個(*<date>*, *<condition>*)	傳回布林條件下次為true的***date***，自指定的***date***起(含)往前推。
最晚()	將傳回與最晚之可能***date***相等的***date***值，亦即***date***保證比***date attribute***採用的或運算式可能評估的任何***date***晚。
最早()	傳回與最早之可能***date***相等的***date***值，亦即***date***保證比***date attribute***採用的或運算式可能評估的任何***date***早。
暫存日數自(*<date>*, *<end-date>*)	傳回每天變化且自***date***起完整天數的數字變數。
暫存週數自(*<date>*, *<end-date>*)	傳回每週變化的數字變數，且其為自***date***起的完整週數。
暫存月數自(*<date>*, *<end-date>*)	傳回每個月變化的數字變數，且其為自***date***起的完整月數。備註：如果提供的***date***在該月的第28日之後，且後續月份的天數較提供的月份少，則週年月份的變更點將會建立在該月的最後一天。例如，如果提供的***date***為2007年1月28、29、30或31日，則第一個變

語法	描述
	更點將會是2007年2月28日。
暫存年數自(*<date>*, *<end-date>*)	傳回每年變化的數字變數，且其為自 ***date*** 起的完整年份數。
暫存一律數日(*<days>*, *<condition>*)	傳回在一段期間中變化的布林 ***attribute***，如果且只有在布林條件針對所有指定前置天數(不含當日)為true時，其為true。
暫存連續數日(*<minDays>*, *<days>*, *<condition>*)	傳回在一段期間中變化的布林 ***attribute***，如果且只有在布林條件至少針對前置設定天數中任何時間的最短連續天數(不含當日)為true時，其為true。
暫存有時候數日(*<days>*, *<condition>*)	傳回在一段期間中變化的布林 ***attribute***，其只有在布林條件於指定的前置天數內(不包含當日)才為true。
暫存之後(*<date>*)	傳回在一段期間中變化的布林 ***attribute***，其在 ***date*** 之後為true，且在當時或之前為false。
暫存當日前(*<date>*)	傳回在一段期間中變化的布林 ***attribute***，其在 ***date*** 之前為true，且在當時或之後為false。
暫存當日(*<date>*)	傳回在一段期間中變化的布林 ***attribute***，其在 ***date*** 時為true，且在之前或之後為false。
暫存當日或當日後(*<date>*)	傳回在一段期間中變化的布林 ***attribute***，其在 ***date*** 當時或之後為true，且在之前為false。
暫存當日或當日前(*<date>*)	傳回在一段期間中變化的布林 ***attribute***，其在 ***date*** 當時和之前為true，且在之後為false。
暫存自開始日期起(*<relationship>*, *<date>*, *<value>*)	以自開始 ***date attribute*** 起有效的值，來從 ***relationship*** 傳回單一暫存 ***attribute***(位於來源 ***entity*** 層次)與實體上的值 ***attribute***。
暫存自結束日期起(*<relationship>*, *<date>*, *<value>*)	以直到結束 ***date attribute*** 前皆有效的值，來從 ***relationship*** 傳回單一暫存 ***attribute***(位於來源 ***entity*** 層次)與實體上的值 ***attribute***。
暫存自範圍起(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	以自開始 ***date attribute***(含)直到結束 ***date attribute***(不含)止有效的值，來從 ***relationship*** 傳回單一暫存 ***attribute***(位於來源 ***entity*** 層次)與實體上的值 ***attribute***。如果值在下一個開始 ***date*** 到期，則值為 ***uncertain***。
暫存為平日(*<startdate>*, *<end-date>*)	如果日期為平日，則傳回true，如果日期為自指定開始 ***date***(含)至結束 ***date***(不含)的週末，則傳回false。如果不在 ***date*** 範圍內，則傳回 ***uncertain***。
暫存每月一次(*<startdate>*, *<end-date>*, *<dayofmonth>*)	如果日期等於該月某日的參數，則傳回true，如果落在從指定開始 ***date***(含)至結束 ***date***(不含)的該月所有其他日，則傳回false。如果不在 ***date*** 範圍內，則傳回 ***uncertain***。當該月某日超過目前月份中的天數時，值會在該月的最後一天為true，因此函數會傳回確實在每月一天為true的值。

驗證事件函數(English)

語法	描述
錯誤(*<text>*)	錯誤事件是用來傳遞訊息給使用者，且讓其等到觸發該錯誤的條件不再作用後，再繼續進行調查。
警告(*<text>*)	警告事件是用來傳遞訊息給使用者，但讓其在觸發該錯誤的條件作用時仍可繼續作業。

已過時函數(English)

語法	描述
呼叫自訂函式(*<A>*, *<B>*)	傳回對代碼程式庫的外部呼叫結果。必須對決定引擎提供代碼程式庫，自訂函數呼叫才能順利執行。

Logické spojky(English)

Syntaxe	Popis
pokud	Volitelný termín, který se může vyskytovat na konci řádku závěru, který obsahuje následující důkaz.
a	Logická konjunkce mezi dvěma atributy ***attributes***
nebo	Logická disjunkce mezi dvěma atributy ***attributes***
buď **jeden z** **libovolný** **nejméně jedna z následujících podmínek je pravda** **je splněna kterákoli z následujících podmínek**	Prvek seskupení použitý při disjunkcích, kde je nutné seskupit dva nebo více atributů ***attributes***.
obojí **vše** **všechny následující podmínky jsou pravda** **jsou splněny všechny následující podmínky**	Prvek seskupení použitý při konjunkcích, kde je nutné seskupit dva nebo více atributů ***attributes***.
jinak	Termín, který se vyskytuje na konci pravidla tabulky a označuje klauzuli „jinak“.
je	Termín používaný v zadání legendy mezi stručnou frází a celým atributem ***attribute*** (***text***)

Logické funkce(English)

Syntaxe	Popis
není pravda, že *<attr>*	Použitý operátor vrací hodnotu pravda, pokud má atribut ***attribute*** hodnotu nepravda.
<var> **je jisté** **je jisté, zda** *<attr>*	Použitý operátor vrací hodnotu pravda, pokud má prvek ***attribute*** hodnotu, která není ***uncertain***.
<var> **není jisté** **není jisté, že** *<attr>* **není jisté, zda** *<attr>* **není jisté, že** *<attr>* **není jisté**	Použitý operátor vrací hodnotu pravda, pokud má atribut ***attribute*** hodnotu ***uncertain***.
<var> **je známo** **je známo, zda** *<attr>*	Použitý operátor vrací hodnotu pravda, pokud atribut ***attribute*** má nějakou hodnotu.
<var> **není známo** **není známo, zda** *<attr>* **neznámý**	Použitý operátor vrací hodnotu pravda, pokud atribut ***attribute*** nemá žádnou hodnotu.

Logické konstanty(English)

Syntaxe	Popis
pravda	Konstanta s hodnotou pravda používaná pro pravidla tabulky.
nepravda	Konstanta s hodnotou nepravda používaná pro pravidla tabulky.
není jisté	Konstanta ***uncertain*** používaná pro pravidla tabulky.

Porovnávací operátory(English)

Syntaxe	Popis
*<x>***<***<y>*	Menší než Poznámka: Neexistuje přirozená jazyková forma, pokud je tento operátor použit spolu s číselnými hodnotami a hodnotami měn.
<x> **>** *<y>*	Větší než Poznámka: Neexistuje přirozená jazyková forma, pokud je tento operátor použit spolu s číselnými hodnotami a hodnotami měn.
<x> **<=***<y>*	Menší nebo rovno
<x> **>=** *<y>*	Větší nebo rovno
*<x>***=***<y>*	Rovná se

Syntaxe	Popis
<x> **<>** *<y>*	Nerovná se

Číselné funkce(English)

Syntaxe	Popis
Číslo(*<numText>*)	Převede zadaný řetězec na číselnou hodnotu.
<x> **+** *<y>*	Matematické sčítání
<x> **-** *<y>*	Matematické odčítání
<x> ***** *<y>*	Matematické násobení
<x> **/** *<y>*	Matematické dělení
<x> **** *<y>*	Celočíselné dělení
<x> **modulo** *<y>*	Zbytek po celočíselném dělení
Maximální(*<x>*, *<y>*) **Maximální**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Vrátí větší ze dvou hodnot.
Minimální(*<x>*, *<y>*) **Minimální**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Vrátí menší ze dvou hodnot.
Xy(*<x>*, *<y>*)	y-tá mocnina x
Ex(*<x>*)	Obsahuje e na x
Abs(*<x>*) **|***<val>***|**	Absolutní hodnota x
Ln(*<x>*)	Přirozený logaritmus x
Log(*<x>*)	Logaritmus x (o základu 10)
DruháOdmocnina(*<x>*)	Druhá odmocnina x
Zaokrouhlit(*<x>*, *<n>*)	Zaokrouhlí x na n desetinných míst.
CeločíselnáHodnota(*<x>*, *<n>*)	hodnota x oříznutá na n desetinných míst
Sin(*<x>*)	Sinus x
Cos(*<x>*)	Kosinus x
Tg(*<x>*)	Tangens x
Arcsin(*<x>*)	Arkussinus x

Syntaxe	Popis
Arccos(*<x>*)	Arkussinus x
Arctg(*<x>*)	Arkustangens x

Funkce data(English)

Syntaxe	Popis
AktuálníDatum()	Vrátí aktuální ***date*** na začátku relace.
Datum(*<text>*)	Převede zadaný řetězec na hodnotu ***date***.
DatumProvedení(*<year>*, *<month>*, *<day>*)	Vrátí hodnotu ***date*** vytvořenou ze zadaného roku, měsíce a dne.
VybratDen(*<date/datetime>*)	Vrátí komponentu dnů z hodnoty ***date/datetime attribute***.
VybratMěsíc(*<date/datetime>*)	Vrátí komponentu měsíců z hodnoty ***date/datetime attribute***.
VybratRok(*<date/datetime>*)	Vrátí komponentu roků z hodnoty ***date/datetime attribute***.
DalšíDenVTýdnu(*<date/datetime>*, *<day>*)	Vrátí hodnotu ***date*** dalšího všedního dne k datu ***date***, před tímto datem nebo po něm (v závislosti na použité syntaxi).
DalšíDen(*<date>*, *<day>*, *<month>*)	Vrátí další instanci daného dne a měsíce po datu ***date***.
PřidatDny (*<date/datetime>*, *<num_days>*)	Přičte nebo odečte počet dnů k ***date***. Při použití stručné syntaktické formy tento počet musí být kladné celé číslo, aby se počet dnů přičetl ke vstupu ***date***, nebo záporné číslo, aby se počet dnů odečetl od vstupu ***date***.
PřidatTýdny(*<date/datetime>*, *<num_weeks>*)	Přičte počet týdnů k datu ***date***. Při použití stručné syntaktické formy tento počet musí být kladné celé číslo, aby se počet týdnů přičetl ke vstupu ***date***.
PřidatMěsíce(*<date/datetime>*, *<num_months>*)	Přičte počet měsíců k ***date***. Při použití stručné syntaktické formy tento počet musí být kladné celé číslo, aby se počet měsíců přičetl ke vstupu ***date***.
PřidatRoky(*<date/datetime>*, *<num_years>*)	Přičte počet roků k datu ***date***. Při použití stručné syntaktické formy tento počet musí být kladné celé číslo, aby se počet roků přičetl ke vstupu ***date***.
PočetDníVTýdnu(*<date1>*, *<date2>*)	Spočítá počet všedních dní mezi hodnotami data a času ***date***1 a ***date***2, to znamená počet dní spadajících mezi pondělí a pátek. Poznámka: První datum ***date*** je zahrnuto, druhé datum ***date*** zahrnuto není.
ZačátekRoku(*<date/datetime>*)	Vrátí první komponentu ***date*** v roce, do kterého hodnota ***date*** spadá.

Syntaxe	Popis
KonecRoku(*<date/datetime>*)	Vrátí poslední komponentu ***date*** v roce, do kterého hodnota ***date*** spadá.
DenníRozdíl(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet dní mezi hodnotami ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
VčetněDenníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet dní (včetně hraničních dat) mezi hodnotami ***date/datetime1*** a ***date/datetime2***. Tento výpočet zahrnuje obě hraniční data. Pokud jsou data stejná, je výsledek 1. Pořadí obou dat nemá na výsledek vliv.
MimoDenníRozdíl(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet dní (bez hraničních dat) mezi hodnotami ***date/datetime1*** a ***date/datetime2***. Tento výpočet nezahrnuje obě hraniční data. Pokud jsou data stejná, je výsledek 0. Pořadí obou dat nemá na výsledek vliv.
TýdenníRozdíl(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet uplynulých týdnů mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
VčetněTýdenníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet uplynulých týdnů (včetně krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
BezTýdenníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet uplynulých týdnů (bez krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
MěsíčníRozdíl(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet celých uplynulých měsíců mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
VčetněMěsíčníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet celých uplynulých měsíců (včetně krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
BezMěsíčníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet celých uplynulých měsíců (bez krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
RočníRozdíl(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet let mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
VčetněRočníhoRozdílu(*<date/datetime1>*, *<date/datetime2>*)	Vrátí počet let (včetně krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.
BezRočníhoRozdílu(*<date/datetime1>*, *<date/d-*	Vrátí počet let (bez krajních hodnot) mezi hodnotami data a času ***date/datetime1*** a ***date/datetime2***. Pořadí obou dat nemá na výsledek vliv.

Syntaxe	Popis
atetime2>)	

Funkce času(English)

Syntaxe	Popis
Čas(*<text>*)	Převede zadaný řetězec na čas.
VybratSekundu(*<time/datetime>*)	Vrátí komponentu sekund z hodnoty ***timeofday/datetime attribute***.
VybratMinutu(*<time/datetime>*)	Vrátí komponentu minut z hodnoty ***timeofday/datetime attribute***.
VybratHodinu(*<time/datetime>*)	Vrátí komponentu hodin z hodnoty ***timeofday/datetime attribute***.

Funkce data a času(English)

Syntaxe	Popis
AktuálníDatumAČas()	Vrátí aktuální ***date*** a čas na začátku relace.
DatumAČas(*<text>*)	Převede zadaný řetězec na hodnotu ***datetime***.
ZřetězenýDenAČas(*<date>*, *<time>*)	Nastaví čas ***date*** spojením hodnoty ***date*** a denní doby dohromady.
SekundovýRozdíl(*<datetime1>*, *<datetime2>*) **SekundovýRozdíl** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet sekund mezi hodnotami data a času ***datetime1*** a ***datetime2***.
VčetněSekundovéhoRozdílu (*<datetime1>*, *<datetime2>*) **VčetněSekundovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet sekund (včetně krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
BezSekundovéhoRozdílu (*<datetime1>*, *<datetime2>*) **BezSekundovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet sekund (bez krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
MinutovýRozdíl (*<datetime1>*, *<datetime2>*) **MinutovýRozdíl** (*<timeOfDay1>*,	Vrátí počet minut mezi hodnotami data a času ***datetime1*** a ***datetime2***.

Syntaxe	Popis
<timeOfDay2>)	
VčetněMinutovéhoRozdílu (*<datetime1>*, *<datetime2>*) **VčetněMinutovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet minut (včetně krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
BezMinutovéhoRozdílu(*<datetime1>*, *<datetime2>*) **BezMinutovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet minut (bez krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
HodinovýRozdíl (*<datetime1>*, *<datetime2>*) **HodinovýRozdíl** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet hodin mezi hodnotami data a času ***datetime1*** a ***datetime2***.
VčetněHodinovéhoRozdílu (*<datetime1>*, *<datetime2>*) **VčetněHodinovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet hodin (včetně krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
BezHodinovéhoRozdílu(*<datetime1>*, *<datetime2>*) **BezHodinovéhoRozdílu** (*<timeOfDay1>*, *<timeOfDay2>*)	Vrátí počet hodin (bez krajních hodnot) mezi hodnotami data a času ***datetime1*** a ***datetime2***.
VybratDatum(*<datetime>*)	Odečte ***date*** od ***datetime attribute***.
VybratČas(*<datetime>*)	Vybere čas a den z hodnoty data a času ***datetime attribute***. Lze použít k nastavení hodnoty ***timeofday attribute*** na čas provedení pravidla výběrem času z aktuální hodnoty ***date*** a čas.
PřidatHodiny(*<datetime>*, *<num_hours>*) **PřidatHodiny**(*<timeOfDay>*, *<num_hours>*)	Přičte počet hodin k hodnotě času ***date***.
PřidatMinuty(*<datetime>*, *<num_minutes>*) **PřidatMinuty**(*<timeOfDay>*, *<num_minutes>*)	Přičte počet minut k hodnotě času ***date***.

Syntaxe	Popis
PřidatSekundy(<*datetime*>, <*num_seconds*>) **PřidatSekundy** (<*timeOfDay*>, <*num_ seconds*>)	Přičte počet sekund k hodnotě času ***date***.

Textové funkce(English)

Syntaxe	Popis
<*text1*> **&** <*text2*>	Zkombinuje ***text1*** s ***text2*** atd. a vytvoří jedinou hodnotu ***text***. Poznámka: můžete použít proměnnou libovolného typu. Hodnoty jsou formátovány pomocí nástroje pro formátování, který je instalován v relaci pravidla.
	Zkombinuje ***text1*** s ***text2*** atd. a vytvoří jedinou hodnotu ***text***. Poznámka: můžete použít proměnnou libovolného typu. Hodnoty jsou formátovány pomocí nástroje pro formátování, který je instalován v relaci pravidla.
Obsahuje(<*text*>, <*substring*>)	Vrátí booleovskou hodnotu, která udává, zda daná hodnota ***text*** obsahuje daný dílčí řetězec ***text***. Porovnání ***text*** nerozlišuje malá a velká písmena.
KončíČím(<*text*>, <*substring*>)	Vrátí booleovskou hodnotu, která udává, zda daná hodnota ***text*** končí daným dílčím řetězcem ***text***. Porovnání ***text*** nerozlišuje malá a velká písmena.
JeČíslo(<*text*>)	Vrátí booleovskou hodnotu, která udává, zda daná hodnota ***text*** představuje platné číslo.
Délka(<*text*>)	Vrátí délku zadané hodnoty ***text*** ve znacích.
ZačínáČím(<*text*>, <*substring*>)	Vrátí booleovskou hodnotu, která udává, zda daná hodnota ***text*** začíná daným dílčím řetězcem ***text***. Porovnání ***text*** nerozlišuje malá a velká písmena.
Podřetězec(<*text*>, <*offset*>, <*length*>)	Vrátí dílčí řetězec z textu ***text*** začínající na daném odsazení, který má zadaný počet znaků. Pokud funkce dosáhne konec řetězce, vrátí méně znaků.
Text(<*number*>) **Text**(<*date*>) **Text**(<*datetime*>) **Text**(<*timeOfDay*>)	Převede zadané číslo nebo datum ***date attribute*** na hodnotu ***text***.

Funkce entity a vztahu(English)

Syntaxe	Popis
Pro(<*relationship*>, <*Exp*>)	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:1, N:1 nebo N:N ***relationship***, existuje-li pouze jedna podmínka.
ProRozsah(<*relationship*>,	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:1, N:1 nebo

Syntaxe	Popis
<alias>) **ProRozsah**(*<relationship>*)	N:N ***relationship***, existuje-li jedna nebo více podmínek.
ProVšechny(*<relationship>*, *<Exp>*)	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:N nebo N:N ***relationship***, když potřebujete určit, zda všechny členy cílové skupiny entit ***entity*** musí vyhovovat pravidlu. Tento tvar se používá, pokud pravidlo obsahuje jen jednu podmínku.
ProCelýRozsah(*<relationship>*) **ProCelýRozsah**(*<relationship>*, *<alias>*)	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:N nebo N:N ***relationship***, když potřebujete určit, zda všechny členy cílové skupiny entit ***entity*** musí vyhovovat pravidlu. Tento tvar se používá, pokud pravidlo obsahuje jednu nebo více podmínek.
Existuje(*<relationship>*, *<Exp>*)	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:N nebo N:N ***relationship***, když potřebujete určit, zda některé členy cílové skupiny entit ***entity*** musí vyhovovat pravidlu. Tento tvar se používá, pokud pravidlo obsahuje jen jednu podmínku.
ExistujícíRozsah(*<relationship>*) **ExistujícíRozsah**(*<relationship>*, *<alias>*)	Slouží k odkazování z jedné entity ***entity*** do jiné entity ***entity*** ve vztahu 1:N nebo N:N ***relationship***, když potřebujete určit, zda některé členy cílové skupiny entit ***entity*** musí vyhovovat pravidlu. Tento tvar se používá, pokud pravidlo obsahuje jednu nebo více podmínek.
JeČlenem(*<target>*, *<relationship>*) **JeČlenem**(*<target>*, *<alias>*, *<relationship>*)	Slouží jako závěr k odvození, že instance ***entity*** je členem vztahu ***relationship***. Slouží jako podmínka k otestování, že instance ***entity*** je cílem vztahu ***relationship***, jehož zdrojem je druhá instance ***entity***.
NeníČlenem(*<target>*, *<relationship>*)	Slouží jako podmínka k testování, zda instance entity ***entity*** není cílem vztahu ***relationship***, pro který je druhá instance entity ***entity*** zdrojem.
PočetInstancí(*<relationship>*)	Spočítá počet instancí existujících pro entitu ***entity***.
PočetInstancíPokud(*<relationship>*, *<Exp>*)	Spočítá počet instancí entity ***entity***, pro které má určitý atribut ***entity-level attribute*** konkrétní hodnotu.
MaximálníPočetInstancí(*<relationship>*, *<number-attr>*) **MaximálníPočetInstancí**(*<relationship>*, *<date-attr>*) **MaximálníPočetInstancí**(*<relationship>*, *<datetime-attr>*) **MaximálníPočetInstancí**(*<relationship>*, *<time-attr>*)	Získá nejvyšší/nejaktuálnější hodnotu proměnné ***entity-level*** pro všechny instance entity ***entity***.
MaximálníPočetInstancíPokud(*<relationship>*, *<number-attr>*,	Získá nejvyšší/nejaktuálnější hodnotu proměnné ***entity-level*** pro všechny instance entity ***entity***, u které má určitý atribut ***entity-level attribute*** určitou hodnotu.

Syntaxe	Popis
<condition>) **MaximálníPočetInstancíPokud** (*<relationship>*, *<date-attr>*, *<condition>*) **MaximálníPočetInstancíPokud** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MaximálníPočetInstancíPokud** (*<relationship>*, *<time-attr>*, *<condition>*)	
MinimálníPočetInstancí(*<relationship>*, *<number-attr>*) **MinimálníPočetInstancí**(*<relationship>*, *<date-attr>*) **MinimálníPočetInstancí**(*<relationship>*, *<datetime-attr>*) **MinimálníPočetInstancí**(*<relationship>*, *<time-attr>*)	Získá nejnižší/nejméně aktuální hodnotu proměnné ***entity-level*** pro všechny instance entity ***entity***.
MinimálníPočetInstancíPokud (*<relationship>*, *<number-attr>*, *<condition>*) **MinimálníPočetInstancíPokud** (*<relationship>*, *<date-attr>*, *<condition>*) **MinimálníPočetInstancíPokud** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MinimálníPočetInstancíPokud** (*<relationship>*, *<time-attr>*, *<condition>*)	Získá nejnižší/nejméně aktuální hodnotu proměnné ***entity-level*** pro všechny instance entity ***entity***, u které má určitý atribut ***entity-level attribute*** určitou hodnotu.
SoučetInstancí(*<relationship>*, *<number-attr>*)	Získá součet všech instancí proměnné ***entity-level***.
SoučetInstancíPokud(*<relationship>*, *<number-attr>*, *<condition>*)	Získá součet všech instancí proměnné ***entity-level***, u které pro entitu ***entity*** platí, že specifická booleovská entita ***entity-level*** má hodnotu ***attribute*** pravda.
HodnotaInstancePokud(*<relationship>*, *<number-attr>*, *<condition>*) **HodnotaInstancePokud**(*<relationship>*, *<text-attr>*,	Získá hodnotu z jedinečné instance ***entity***, určené z ***entity*** cílových instancí ***relationship*** na základě podmínky. • Pokud podmínka označí jedinou cílovou instanci ***entity***, pak jde o hodnotu vypočtenou oproti instanci ***entity***. • Pokud podmínku splňuje více než jedna cílová instance, pak je vráceno ***uncer-***

Syntaxe	Popis
<condition>) **HodnotaInstancePokud**(*<relationship>*, *<date-attr>*, *<condition>*) **HodnotaInstancePokud**(*<relationship>*, *<datetime-attr>*, *<condition>*) **HodnotaInstancePokud**(*<relationship>*, *<time-attr>*, *<condition>*)	***tain***. • Pokud podmínku nesplňuje žádná cílová instance a je známo ***relationship***, je hodnota ***uncertain***.
InstanceRovna(*<instance1>*, *<instance2>*)	Určí, zda jsou dvě instance entity ***entity*** totožné.
InstanceNeníRovna (*<instance1>*, *<instance2>*)	Určí, zda dvě instance entity ***entity*** nejsou totožné.
InferInstance(*<relationship>*, *<identity>*)	Slouží jako závěr k odvození, že instance ***entity*** existuje a je členem vztahu ***relationship***.

Funkce časové logiky(English)

Syntaxe	Popis
PřesnýPočetIntervalů(*<start-date>*, *<end-date>*, *<variable>*) **PřesnýPočetIntervalů**(*<start-date>*, *<end-date>*, *<condition>*)	Spočítá počet známých odlišných hodnot proměnné v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto).
PřesnýPočetIntervalůPokud (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Spočítá počet známých odlišných hodnot proměnné v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto). Zahrne pouze případy, kdy má booleovský filtr hodnotu pravda.
DenníSoučetIntervalů(*<start-date>*, *<end-date>*, *<number-attr>*)	Spočítá součet číselné nebo měnové proměnné v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto). Předpokládá se, že atribut ***attribute*** je denní množství.
DenníSoučetIntervalůPokud (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Vypočítá součet všech denních hodnot měny nebo číselné proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto). Zahrne pouze případy, kdy má podmínka hodnotu pravda.
MaximálníInterval(*<start-date>*, *<end-date>*, *<number-attr>*) **MaximálníInterval**(*<start-date>*, *<end-date>*, *<date-attr>*) **MaximálníInterval**(*<start-date>*,	Vybere maximální hodnotu proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto).

Syntaxe	Popis
<end-date>, *<datetime-attr>*) **MaximálníInterval**(*<start-date>*, *<end-date>*, *<time-attr>*)	
MaximálníIntervalPokud(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MaximálníIntervalPokud**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MaximálníIntervalPokud**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MaximálníIntervalPokud**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Vybere maximální hodnotu proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto). Zahrne pouze případy, kdy má podmínka hodnotu pravda.
MinimálníInterval(*<start-date>*, *<end-date>*, *<number-attr>*) **MinimálníInterval**(*<start-date>*, *<end-date>*, *<date-attr>*) **MinimálníInterval**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MinimálníInterval**(*<start-date>*, *<end-date>*, *<time-attr>*)	Vybere minimální hodnotu proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto).
MinimálníIntervalPokud(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MinimálníIntervalPokud**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MinimálníIntervalPokud**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MinimálníIntervalPokud**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Vybere minimální hodnotu proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto). Zahrne pouze případy, kdy má podmínka hodnotu pravda.
VáženýPrůměrIntervalu(*<start-date>*, *<end-date>*, *<number-attribute>*)	Vypočte průměrnou hodnotu měnové nebo číselné proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto) váženou podle časového rozmezí, kterého se jednotlivé hodnoty týkají.
VáženýPrůměrIntervaluPokud (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Vypočte průměrnou hodnotu měnové nebo číselné proměnné v intervalu od počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto), přičemž zahrne pouze případy, kdy má booleovská podmínka hodnotu pravda (váženo podle

Syntaxe	Popis
	časového rozmezí, kterého se jednotlivé hodnoty týkají a filtr má hodnotu pravda).
IntervalVždy(*<start-date>*, *<end-date>*, *<condition>*)	Vrátí hodnotu pravda, pokud a pouze pokud má booleovská podmínka hodnotu pravda vždy v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto).
IntervalNejmenšíhoPočtuDnů (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Vrátí hodnotu pravda, pokud a pouze pokud má booleovská podmínka hodnotu pravda alespoň pro zadaný počet dní (ne nezbytně jdoucích po sobě) v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto).
IntervalPoSoběJdoucíchDnů (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Vrátí hodnotu pravda, pokud a pouze pokud má booleovská podmínka hodnotu pravda alespoň pro zadaný počet po sobě jdoucích dní v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto).
IntervalNěkdy(*<start-date>*, *<end-date>*, *<condition>*)	Vrátí hodnotu pravda tehdy a pouze tehdy, pokud má booleovská podmínka někdy hodnotu pravda v intervalu od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto).
PřiHodnotě(*<date>*, *<value>*)	Vrací hodnotu daného atributu ***attribute*** k zadanému datu ***date***.
PřiPosledním(*<date>*, *<condition>*)	Vrátí datum ***date***, u kterého má booleovská podmínka hodnotu pravda, bráno zpětně (a včetně) od zadaného data ***date***.
PřiNásledujícím(*<date>*, *<condition>*)	Vrátí datum ***date***, u kterého bude mít booleovská podmínka příště hodnotu pravda, bráno dopředu (a včetně) od zadaného data ***date***.
Nejpozději()	Vrátí hodnotu ***date*** odpovídající nejpozdější možné hodnotě ***date*** - jmenovitě hodnotu ***date***, která určitě následuje po kterékoli jiné hodnotě ***date***, kterou může přebírat atribut ***date attribute*** nebo kterou může vyhodnotit výraz.
Nejdříve()	Vrátí hodnotu ***date*** odpovídající nejnižší možné hodnotě ***date*** - jmenovitě hodnotu ***date***, která určitě předchází kterékoli jiné hodnotě ***date***, kterou může přebírat atribut ***date attribute*** nebo kterou může vyhodnotit výraz.
DočasněOdeDne(*<date>*, *<end-date>*)	Vrátí číselnou proměnnou, která se mění každý den a představuje počet celých dní od data ***date***.
DočasněOdTýdne(*<date>*, *<end-date>*)	Vrátí číselnou proměnnou, která se mění každý týden a představuje počet celých týdnů od data ***date***.
DočasněOdMěsíce(*<date>*, *<end-date>*)	Vrátí číselnou proměnnou, která se mění každý měsíc, a představuje počet celých měsíců od data ***date***. Poznámka: Pokud zadané datum ***date*** následuje po 28. dni měsíce a následující měsíc má méně dní než zadaný měsíc, bude bod změny pro výroční měsíc vytvořen k poslednímu dni daného měsíce. Příklad: Pokud zadáte hodnotu ***date*** jako 28., 29., 30. nebo 31. ledna 2007, bude prvním bodem změny 28. únor 2007.
DočasněOdRoku(*<date>*, *<end-date>*)	Vrátí číselnou proměnnou, která se mění každý rok a představuje počet celých let od data ***date***.

Syntaxe	Popis
DočasněVždyDny(*<days>*, *<condition>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda tehdy a pouze tehdy, pokud má booleovská podmínka hodnotu pravda u všech zadaných předchozích dní, mimo aktuálního dne.
DočasněPoSoběJdoucíDny (*<minDays>*, *<days>*, *<condition>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda tehdy a pouze tehdy, pokud má booleovská podmínka hodnotu pravda u minimálního počtu po sobě jdoucích dní kdykoli v zadaném intervalu předchozích dní, mimo aktuálního dne.
DočasněNěkdyDny(*<days>*, *<condition>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda tehdy a pouze tehdy, pokud má booleovská podmínka někdy hodnotu pravda u zadaného rozsahu předchozích dní, mimo aktuálního dne.
DočasnéPoDatu(*<date>*)	Vrátí booleovskou hodnou ***attribute***, která se s časem mění a má hodnotu pravda po datu ***date*** a hodnotu nepravda k danému datu a před ním.
DočasnéDoData(*<date>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda před datem ***date*** a hodnotu nepravda k danému datu a po něm.
DočasnéKDatu(*<date>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda k datu ***date*** a hodnotu nepravda před a po něm.
DočasnéKDatuNeboPoTomtoDatu (*<date>*)	Vrátí booleovskou hodnotu ***attribute***, která se s časem mění a má hodnotu pravda k datu a po datu ***date*** a hodnotu nepravda před ním.
DočasnéDoDataNeboKTomutoDatu (*<date>*)	Vrátí booleovskou hodnou ***attribute***, která se s časem mění a má hodnotu pravda k datu a před datem ***date*** a hodnotu nepravda po něm.
DočasněOdPočátečníhoData(*<relationship>*, *<date>*, *<value>*)	Vrátí jediný časový atribut ***attribute*** (na úrovni zdrojové entity ***entity***) ze vztahu ***relationship*** a hodnoty ***attribute*** u entit, s hodnotami, které jsou platné od počátečního atributu ***date attribute***.
DočasněDoData(*<relationship>*, *<date>*, *<value>*)	Vrátí jediný časový atribut ***attribute*** (na úrovni zdrojové entity ***entity***) ze vztahu ***relationship*** a hodnoty ***attribute*** u entit, s hodnotami, které jsou platné do koncového atributu ***date attribute***.
DočasněVRozsahuOd (*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Vrátí jediný časový atribut ***attribute*** (na úrovni zdrojové entity ***entity***) ze vztahu ***relationship*** a hodnoty ***attribute*** u entit, s hodnotami, které jsou platné od počátečního atributu ***date attribute*** (včetně) do koncového atributu ***date attribute***. Hodnota je ***uncertain***, pokud vyprší před příštím počátečním atributem ***date***.
DočasněVšedníDen(*<startdate>*, *<enddate>*)	Vrátí hodnotu pravda u dat, která představují všední dny, a hodnotu nepravda u dat, která představují víkendy, od počátečního data ***date*** (včetně) do koncového data ***date*** (není zahrnuto). Vrátí hodnotu ***uncertain*** mimo rozsah hodnot ***date***.
DočasnéJednouMěšíčně (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Vrátí hodnotu pravda, pokud se den rovná parametru dne měsíce, a hodnotu nepravda u všech ostatních dní v měsíci od zadaného počátečního data ***date*** (včetně) po koncové datum ***date*** (není zahrnuto). Vrátí hodnotu ***uncertain*** u hodnot mimo rozsah hodnot ***date***. Když parametr dne v měsíci překročí počet dní v

Syntaxe	Popis
	aktuálním měsíci, bude vrácena hodnota pravda u posledního dne v měsíci, aby funkce vrátila hodnotu, která má hodnotu pravda přesně jeden den v měsíci.

Funkce události ověření(English)

Syntaxe	Popis
Chyba (*<text>*)	Chybová událost slouží k předání zprávy uživateli a zabránění v dalším šetření, dokud přestane platit podmínka, která vyvolala chybu.
Varování (*<text>*)	Událost varování slouží k předání zprávy uživateli, ale umožňuje jim pokračovat i přes podmínku, která dané varování vyvolala.

Odmítnuté funkce(English)

Syntaxe	Popis
VolatVlastníFunkci (*<A>*, *<B>*)	Vrátí výsledek externího volání knihovny kódů. Knihovna kódů musí být uvedena, aby bylo volání uživatelské funkce z modulu Determinations Engine úspěšné.

Logiske connectors(English)

Syntaks	Beskrivelse
hvis	Valgfri term, der kan forekomme i slutningen af en konklusionslinje, der har et efterfølgende bevis
og	Logisk konjunktion mellem to ***attributes***
eller	Logisk disjunktion mellem to ***attributes***
enten **en af** **nogen** **mindst en af de følgende er sand** **nogen af de følgende er opfyldt**	Grupperingselement, der bruges med disjunktioner, hvor to eller flere ***attributes*** skal grupperes
begge **alle** **alle af de følgende er rigtige** **alle af de følgende er opfyldt**	Grupperingselement, der bruges med konjunktioner, hvor to eller flere ***attributes*** skal grupperes
ellers	Term, der forekommer i slutningen af en tabelregel for at angive ellers-klausulen

Syntaks	Beskrivelse
er	Term, der bruges i en forklaringsangivelse mellem den forkortede sætning og den fulde ***attribute text***

Logiske funktioner(English)

Syntaks	Beskrivelse
det er ikke sandt at *<expr>*	Operator, der bruges til at returnere sand, hvis ***attribute*** har en værdi, der er falsk
<var> **er sikkert** **det er usikkert at** *<expr>*	Operator, der bruges til at returnere sand, hvis ***attribute*** har en værdi, der ikke er ***uncertain***
<var> **er usikkert** *<var>* **er ikke sikkert** **det er usikkert om** [**eller ikke**]*<expr>* **det er ikke sikkert at** *<expr>*	Operator, der bruges til at returnere sand, hvis ***attribute***-værdi er ***uncertain***
<var> **er kendt** **det er kendt at** [**eller ikke**]*<expr>*	Operator, der bruges til at returnere sand, hvis ***attribute*** har en værdi
<var> **er ikke kendt** **det er ikke kendt at** [**eller ikke**]*<expr>*	Operator, der bruges til at returnere sand, hvis ***attribute*** ikke har nogen værdi

Logiske konstanter(English)

Syntaks	Beskrivelse
sand	Konstant sand værdi, der bruges til tabelregler.
falsk	Konstant falsk værdi, der bruges til tabelregler.
ved ikke **usikker** **usikkert**	Konstant ***uncertain***-værdi, der bruges til tabelregler.

Sammenligningsoperatorer(English)

Syntaks	Beskrivelse
*<lhs>***<***<rhs>* *<lhs>* **er mindre end** *<rhs>* *<lhs>* **er før end** *<rhs>*	Mindre end Bemærk: Der er intet naturligt sprog, når denne operator bruges med numeriske værdier og valutaværdier.

Syntaks	Beskrivelse
<lhs> **>** *<rhs>* *<lhs>* **er større end** *<rhs>* *<lhs>* **er senere end** *<rhs>*	Større end Bemærk: Der er inget naturligt sprog, når denne operator bruges med numeriske værdier og valutaværdier.
*<lhs>***<=***<rhs>*	Mindre end eller lig med
<lhs> **>=** *<rhs>*	Større end eller lig med
*<lhs>***=***<rhs>*	Lig med
<lhs> **<>** *<rhs>*	Ikke lig med

Numeriske funktioner(English)

Syntaks	Beskrivelse
Tal(*<numText>*)	Konverterer den angivne streng til en talværdi
<x> **+** *<y>*	Matematisk addition
<x> **-** *<y>*	Matematisk subtraktion
<lhs> * *<rhs>*	Matematisk multiplikation
<lhs> **/** *<rhs>*	Matematisk division
<lhs> **** *<rhs>*	Heltalsdivision
<lhs> **modulo** *<rhs>*	Rest efter heltalsdivision
Maksimum(*<x>*, *<y>*) **Maksimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **den større af** *<val1>* **og** *<val2>* **den seneste af** *<val1>* **og** *<val2>*	Returnerer den største af to værdier
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **den mindre af** *<val1>* **og** *<val2>* **den tidligste af** *<val1>* **og** *<val2>*	Returnerer den mindste af to værdier
Xy(*<x>*, *<y>*) *<val>* **hævet til potensen af** *<power>*	x opløftet til potensen af y
Ex(*<x>*) **e hævet til potensen af** *<log-val>*	Konstant e opløftet til potensen af x
Abs(*<x>*) **den absolutte værdi af** *<val>*	Absolut værdi af x

Syntaks	Beskrivelse
| *<val>* **|**	
Ln(*<x>*) **den naturlige logaritme af** *<log-val>*	Naturlig logaritme til x
Log(*<x>*) **logaritme grundtal 10 af** *<log-val>*	Titalslogaritmen til x
Kvadr(*<x>*) **Kvadratroden af** *<val>*	Kvadratroden af x
Afrundet(*<x>*, *<n>*) *<val>* **afrundet til** *<num_places>* **decimale pladser**	Afrunder x til n decimaler
Afkortet(*<x>*, *<n>*) *<val>* **truncated til** *<num_places>* **decimale pladser**	x afkortet til n decimaler
Sin(*<x>*)	x's sinusværdi
Cos(*<x>*)	x's cosinus
Tan(*<x>*)	x's tangens
Asin(*<x>*)	x's arcus sinus
Acos(*<x>*)	Arcus cosinus for x
Atan(*<x>*)	x's arcus tangens

Datofunktioner(English)

Syntaks	Beskrivelse
DagsDato() **den aktuelle dato**	Returnerer aktuelle ***date*** i starten af sessionen.
Dato(*<text>*)	Konverterer den angivne streng til en ***date***-værdi
LavDato(*<year>*, *<month>*, *<day>*)	Returnerer en ***date*** dannet fra angivet år, måned og dag.
UdtrækDag(*<date/datetime>*)	Returnerer dagskomponenten for ***date/datetime attribute***.
UdtrækMåned(*<date/datetime>*)	Returnerer månedskomponenten for ***date/datetime attribute***.
UdtrækÅr(*<date/datetime>*)	Returnerer årskomponenten for ***date/datetime attribute***.

Syntaks	Beskrivelse
NæsteUgedag(*<date/datetime>*, *<day>*) **den næste mandag på eller efter** *<from-date>* **NæsteUgeDag(**_<from-date>_**, "Mandag")** **mandagen på eller før** *<from-date>* **NæsteUgeDag(SumDage(**_<from-date>_**, -6), "Mandag")** **den næste tirsdag på eller efter** *<from-date>* **NæsteUgeDag(**_<from-date>_**, "Tirsdag")** **tirsdagen på eller før** *<from-date>* **NæsteUgeDag(SumDage(**_<from-date>_**, -6), "Tirsdag")** **NæsteUgeDag(**_<from-date>_**, "Onsdag")** **den næste onsdag på eller efter** *<from-date>* **onsdagen på eller før** *<from-date>* **NæsteUgeDag(SumDage(**_<from-date>_**, -6), "Onsdag")** **den næste torsdag på eller efter** *<from-date>* **NæsteUgeDag(**_<from-date>_**, "Torsdag")** **torsdagen på eller før** *<from-date>* **NæsteUgeDag(SumDage(**_<from-date>_**, -6), "Torsdag")** **den næste fredag på eller efter** *<from-date>* **NæsteUgeDag(**_<from-date>_**, "Fredag")** **fredagen på eller før**	Returnerer ***date*** for den næste ugedag, der ligger samtidig med eller før/efter en ***date*** (afhængig af den anvendte synktaks).

Syntaks	Beskrivelse
<from-date> **NæsteUgeDag(SumDage (***<from-date>***, -6), "Fredag")** **den næste lørdag på eller efter** *<from-date>* **NæsteUgeDag(***<from-date>***, "Lørdag")** **NæsteUgeDag(SumDage (***<from-date>***, -6), "Lørdag")** **lørdagen på eller før** *<from-date>* **den næste søndag på eller efter** *<from-date>* **NæsteUgeDag(***<from-date>***, "Søndag")** **NæsteUgeDag(SumDage (***<from-date>***, -6), "Søndag")** **søndagen på eller før** *<from-date>*	
NæsteDato(*<date>*, *<day>*, *<month>*) **den forrige danske skatteårs start dato på eller før** *<from-date>* **den næste danske skatteårs slut dato på eller efter** *<from-date>*	Returnerer den næste instans af den givne dag og måned efter en ***date***.
SumDage (*<date/datetime>*, *<num_days>*) **datoen** *<num_days>* **dage efter** *<date>* **datoen** *<num_days>* **dage før** *<date>*	Lægger/trækker et antal dage til/fra en ***date***. Når Terse syntaktisk form bruges, skal tallet være et positivt heltal, for at dage kan lægges til input-***date*** eller et negativt tal, for at dage kan trækkes fra input-***date***.
SumUger(*<date/datetime>*, *<num_weeks>*) **datoen** *<num_weeks>* **uger efter** *<date>* **datoen** *<num_weeks>*	Lægger et antal uger til en ***date***. Når Terse syntaktisk form bruges, skal tallet være et positivt heltal, for at uger kan lægges til input-***date***.

Syntaks	Beskrivelse
uger før *<date>*	
SumMåneder(*<date/datetime>*, *<num_months>*) **datoen** *<num_months>* **måneder efter** *<date>* **datoen** *<num_months>* **måneder før** *<date>*	Lægger et antal måneder til en ***date***. Når Terse syntaktisk form bruges, skal tallet være et positivt heltal, for at måneder kan lægges til input-***date***.
SumÅr(*<date/datetime>*, *<num_years>*) **datoen** *<num_years>* **år efter** *<date>* **datoen** *<num_years>* **år før** *<date>*	Lægger et antal år til en ***date***. Når Terse syntaktisk form bruges, skal tallet være et positivt heltal, for at år kan lægges til input-***date***.
UgedagOptælling (*<date1>*, *<date2>*) **antallet af ugedage (inklusiv) mellem** *<date1>* **og** *<date2>*	Tæller antallet af ugedage mellem ***date***1 og ***date***2. Dvs. antallet af dage, der ligger mellem mandag og fredag. Bemærk: Den tidlige ***date*** er inklusiv og den senere ***date*** er eksklusiv.
ÅrStart(*<date/datetime>*) **den første dag i året hvor** *<from-date>* **falder**	Returnerer første ***date*** i det år, hvor en ***date*** ligger.
ÅrSlut(*<date/datetime>*) **den sidste dag i året hvor** *<from-date>* **falder**	Returnerer sidste ***date*** i det år, hvor en ***date*** ligger.
DagForskel(*<date/datetime1>*, *<date/datetime2>*) **antallet af dage fra** *<date1>* **til** *<date2>* **antallet af dage (inklusiv) fra** *<date1>* **til** *<date2>* **antallet af dage (eksklusiv) fra** *<date1>* **til** *<date2>*	Returnerer antallet af hele dage mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
DagForskelInklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer antallet af hele dage (inklusive) mellem ***date/datetime1*** og ***date/datetime2***. Denne beregning omfatter begge slutpunkter. Resultatet er 1, når datoerne er ens. Rækkefølgen af de to datoer påvirker ikke resultatet.
DagForskelEksklusiv	Returnerer antallet af hele dage (eksklusive) mellem ***date/datetime1*** og ***date/datetime2***.

Syntaks	Beskrivelse
(*<date/datetime1>*, *<date/datetime2>*)	Denne beregning udelader begge slutpunkter. Resultatet er 0, når datoerne er ens. Rækkefølgen af de to datoer påvirker ikke resultatet.
UgeForskel(*<date/datetime1>*, *<date/datetime2>*) **antallet af uge fra** *<date1>* **til** *<date2>*	Returnerer antallet af hele forløbne uger mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
UgeForskelInklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer det inklusive antal af hele forløbne uger mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
UgeForskelEksklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer det eksklusive antal af hele forløbne uger mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
MånedForskel(*<date/datetime1>*, *<date/datetime2>*) **antallet af måneder fra** *<date1>* **til** *<date2>*	Returnerer antallet af hele forløbne måneder mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
MånedForskelInklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer antallet af hele inklusive forløbne måneder mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
MånedForskelEksklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer antallet af hele eksklusive forløbne måneder mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
ÅrForskel(*<date/datetime1>*, *<date/datetime2>*) **antallet af år (inklusiv) mellem** *<date1>* **og** *<date2>* **antallet af hele år hvilke** *<date2>* **er efter** *<date1>*	Returnerer antallet af år mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
ÅrForskelInklusiv(*<date/datetime1>*, *<date/datetime2>*)	Returnerer det inklusive antal år mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.
ÅrForskelEksklusiv (*<date/datetime1>*, *<date/datetime2>*)	Returnerer det eksklusive antal år mellem ***date/datetime1*** og ***date/datetime2***. Rækkefølgen af de to datoer påvirker ikke resultatet.

Klokkeslætsfunktioner(English)

Syntaks	Beskrivelse
Klokkeslæt(*<text>*)	Konverterer den givne streng til et tidspunkt på dagen
UdtrækSekund(*<time/datetime>*)	Returnerer sekundkomponenten for ***timeofday/datetime attribute***.
UdtrækMinut(*<time/datetime>*)	Returnerer minutkomponenten for ***timeofday/datetime attribute***.
UdtrækTime(*<time/datetime>*)	Returnerer timekomponenten for ***timeofday/datetime attribute***.

Dato- og klokkeslætsfunktioner(English)

Syntaks	Beskrivelse
AktuelDatoKlokkeslæt()	Returnerer aktuelle ***date*** og klokkeslæt i starten af sessionen.
DatoKlokkeslæt(*<text>*)	Konverterer den angivne streng til en ***datetime***-værdi
SammenkædDatoKlokkeslæt (*<date>*, *<time>*)	Sætter ***date***/klokkeslæt ved at sammenføje ***date*** og tidspunkt på dagen.
SekundForskel(*<datetime1>*, *<datetime2>*) **SekundForskel** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer antallet af sekunder mellem ***datetime1*** og ***datetime2***.
SekundForskelInklusiv(*<datetime1>*, *<datetime2>*) **SekundForskelInklusiv** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det inklusive antal sekunder mellem ***datetime1*** og ***datetime2***.
SekundForskelEksklusiv (*<datetime1>*, *<datetime2>*) **SekundForskelEksklusiv** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det eksklusive antal sekunder mellem ***datetime1*** og ***datetime2***.
MinutForskel(*<datetime1>*, *<datetime2>*) **MinutForskel**(*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer antallet af minutter mellem ***datetime1*** og ***datetime2***.
MinutForskelInklusiv(*<datetime1>*, *<datetime2>*) **MinutForskelInklusiv** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det inklusive antal minutter mellem ***datetime1*** og ***datetime2***.

Syntaks	Beskrivelse
MinutForskelEksklusiv(*<datetime1>*, *<datetime2>*) **MinutForskelEksklusiv**(*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det eksklusive antal minutter mellem ***datetime1*** og ***datetime2***.
TimeForskel(*<datetime1>*, *<datetime2>*) **TimeForskel**(*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer antallet af timer mellem ***datetime1*** og ***datetime2***.
TimeForskelInklusiv(*<datetime1>*, *<datetime2>*) **TimeForskelInklusiv**(*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det inklusive antal timer mellem ***datetime1*** og ***datetime2***.
TimeForskelEksklusiv(*<datetime1>*, *<datetime2>*) **TimeForskelEksklusiv**(*<timeOfDay1>*, *<timeOfDay2>*)	Returnerer det eksklusive antal timer mellem ***datetime1*** og ***datetime2***.
UdtrækDato(*<datetime>*)	Udtrækker ***date*** fra ***datetime attribute***.
UdtrækKlokkeslæt(*<datetime>*)	Udtrækker tidspunktet på dagen fra en ***datetime attribute***. Kan bruges til at sætte værdien af ***timeofday attribute*** til det klokkeslæt, hvor reglen udføres, ved at udtrække klokkeslættet fra aktuel ***date*** og aktuelt klokkeslæt.
SumTimer(*<datetime>*, *<num_hours>*) **SumTimer**(*<timeOfDay>*, *<num_hours>*)	Lægger et antal timer til ***date***/klokkeslæt.
SumMinutter(*<datetime>*, *<num_minutes>*) **SumMinutter**(*<timeOfDay>*, *<num_minutes>*)	Lægger et antal minutter til ***date***/klokkeslæt.
SumSekunder(*<datetime>*, *<num_seconds>*) **SumSekunder**(*<timeOfDay>*, *<num_seconds>*)	Lægger et antal sekunder til ***date***/klokkeslæt.

Tekstfunktioner(English)

Syntaks	Beskrivelse
<text1> **&** *<text2>*	Kombinerer ***text1*** med ***text2*** osv. for at danne en enkelt ***text***-værdi. Bemærk: Variabler af en hvilken som helst type kan anvendes. Værdier formateres vha. det formateringsprogram, der er installeret i regelsessionen.
sammenkædningen af *<text1>* **&** *<text2>*	Kombinerer ***text1*** med ***text2*** osv. for at danne en enkelt ***text***-værdi. Bemærk: Variabler af en hvilken som helst type kan anvendes. Værdier formateres vha. det formateringsprogram, der er installeret i regelsessionen.
Indeholder(*<text>*, *<substring>*) *<text>* **indeholder** *<substring>*	Returnerer en boolesk værdi, der angiver, om den givne ***text***-værdi indeholder den givne ***text***-understreng. Der skelnes ikke mellem store og små bogstaver i ***text***-sammenligningen.
SlutterMed(*<text>*, *<substring>*) *<text>* **slutter med** *<substring>*	Returnerer en boolesk værdi, der angiver, om den givne ***text***-værdi slutter med den givne ***text***-understreng. Der skelnes ikke mellem store og små bogstaver i ***text***-sammenligningen.
ErTal(*<text>*) *<text>* **er et tal**	Returnerer en boolesk værdi, der angiver, om den givne ***text***-værdi repræsenterer et gyldigt tal.
Længde(*<text>*)	Returnerer tegnlængden for den givne ***text***-værdi.
StarterMed(*<text>*, *<substring>*) *<text>* **begynder med** *<substring>*	Returnerer en boolesk værdi, der angiver, om den givne ***text***-værdi starter med den givne ***text***-understreng. Der skelnes ikke mellem store og små bogstaver i ***text***-sammenligningen.
Understreng(*<text>*, *<offset>*, *<length>*)	Returnerer understrengen af ***text***, der starter ved den givne forskydning, der er den angivne længde i tegn. Færre tegn returneres, hvis slutninggen af strengen er nået.
Tekst(*<number>*) **Tekst**(*<date>*) **Tekst**(*<datetime>*) **Tekst**(*<timeOfDay>*)	Konverterer angivet tal eller ***date attribute*** til en ***text***-værdi.

Entitets- og relationsfunktioner(English)

Syntaks	Beskrivelse
Til(*<relationship>*, *<Exp>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til en", "Mange til en" eller "Mange til mange" ***relationship***, hvor der kun er én betingelse.
TilOmfang(*<relationship>*, *<alias>*) **TilOmfang**(*<relationship>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til en", "Mange til en" eller "Mange til mange" ***relationship***, hvor der er en eller flere betingelser.
TilAlle(*<relationship>*, *<Exp>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til mange" eller "Mange til mange" ***relationship***, når du har brug for at fastlægge, om alle medlemmer af mål-

Syntaks	Beskrivelse
	entity-gruppen skal opfylde reglen. Denne form bruges, når der kun er én betingelse i reglen.
TilAltOmfang(*<relationship>*) **TilAltOmfang**(*<relationship>*, *<alias>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til mange" eller "Mange til mange" ***relationship***, når du har brug for at fastlægge, om alle medlemmer af mål-***entity***-gruppen skal opfylde reglen. Denne form bruges, når der er en eller flere betingelser i reglen.
Findes(*<relationship>*, *<Exp>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til mange" eller "Mange til mange" ***relationship***, når du har brug for at fastlægge, om nogen medlemmer af mål-***entity***-gruppen skal opfylde reglen. Denne form bruges, når der kun er én betingelse i reglen.
FindesOmfang(*<relationship>*) **FindesOmfang**(*<relationship>*, *<alias>*)	Bruges til at referere fra en ***entity*** til en anden ***entity*** i en "En til mange" eller "Mange til mange" ***relationship***, når du har brug for at fastlægge, om nogen medlemmer af mål-***entity***-gruppen skal opfylde reglen. Denne form bruges, når der er en eller flere betingelser i reglen.
ErMedlemAf(*<target>*, *<relationship>*) **ErMedlemAf**(*<target>*, *<alias>*, *<relationship>*)	Bruges som en konklusion til at udlede, at en ***entity***-instans er medlem af en ***relationship***. Bruges som en betingelse til at teste, at en ***entity***-instans er et mål for en ***relationship***, hvortil en anden ***entity***-instans er kilden.
ErIkkeMedlemAf(*<target>*, *<relationship>*)	Bruges som en betingelse til at teste, at en ***entity***-instans ikke er et mål for en ***relationship***, hvortil en anden ***entity***-instans er kilden.
InstansOptælling(*<relationship>*) **antallet af** *<ent>*	Tæller antallet af instanser, der findes til en ***entity***.
InstansOptællingHvis(*<relationship>*, *<Exp>*) **antallet af** *<ent>* **for hvilke det betyder at** *<condition>*	Tæller antallet af instanser, der findes af en ***entity***, hvortil en bestemt ***entity-level attribute*** har en bestemt værdi.
InstansMaksimum (*<relationship>*, *<number-attr>*) **InstansMaksimum** (*<relationship>*, *<date-attr>*) **InstansMaksimum** (*<relationship>*, *<datetime-attr>*) **InstansMaksimum** (*<relationship>*, *<time-attr>*) *<date-attr>* **hvilke er den seneste for alle** [**af**]*<ent>* *<max-attr>* **hvilke er den største for alle** [**af**]*<ent>* **den seneste af alle** *<ent-attr>*	Henter højeste/nyeste værdi af en ***entity-level***-variabel til alle instanser af ***entity***.

Syntaks	Beskrivelse
den største af *<ent-attr>* **den største af alle** *<ent-attr>* **den største af** *<attr>* **for alle** [**af**]*<ent>* **den største af alle** *<attr>* **for** [**alle**]*<ent>* **den seneste af alle** *<attr>* **for** *<ent>*	
InstansMaksimumHvis(*<relationship>*, *<number-attr>*, *<condition>*) **InstansMaksimumHvis**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansMaksimumHvis**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstansMaksimumHvis**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **hvilke er den seneste for alle** [**af**]*<ent>* **for hvilke det betyder at** *<ent-test>* *<max-attr>* **hvilke er den største for alle** [**af**]*<ent>* **for hvilke det betyder at** *<ent-test>* **den seneste af alle** *<ent-attr>* **for hvilke det betyder at** *<ent-test>* **den største af alle** *<ent-attr>* **for hvilke det betyder at** *<ent-test>* **den største af** *<attr>* **for alle** [**af**]*<ent>* **for hvilke det betyder at** *<ent-test>*	Henter højeste/nyeste værdi af en ***entity-level***-variabel til alle instanser af ***entity***, hvortil en bestemt ***entity-level attribute*** har en bestemt værdi.
InstansMinimum(*<relationship>*, *<number-attr>*) **InstansMinimum**(*<relationship>*, *<date-attr>*) **InstansMinimum**(*<relationship>*, *<datetime-attr>*) **InstansMinimum**(*<relationship>*, *<time-attr>*) *<date-attr>* **hvilke er den tidligste for alle** [**af**]*<ent>* *<attr>* **hvilke er den mindste for alle** [**af**]*<ent>*	Henter laveste/mindste værdi af en ***entity-level***-variabel til alle instanser af ***entity***.

Syntaks	**Beskrivelse**
den mindste af alle *<ent-attr>* **den mindste af alle** *<attr>* **for** [**alle**]*<ent>* **den mindste af alle** *<attr>* **for** *<ent>*	
InstansMinimumHvis(*<relationship>*, *<number-attr>*, *<condition>*) **InstansMinimumHvis**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansMinimumHvis**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstansMinimumHvis**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **hvilke er den tidligste for alle** [**af**]*<ent>* **for hvilke det betyder at** *<ent-test>* **den mindste af alle** *<ent-attr>* **for hvilke det betyder at** *<ent-test>* **den mindste af alle** *<attr>* **for** *<ent>* **for hvilke det betyder at** *<ent-test>* **den tidligste af alle** *<attr>* **for** *<ent>* **for hvilke det betyder at** *<ent-test>*	Henter laveste/mindste værdi af en ***entity-level***-variabel til alle instanser af ***entity***, hvortil en bestemt ***entity-level attribute*** har en bestemt værdi.
InstansSum(*<relationship>*, *<number-attr>*) *<num-attr>* **totalt for alle** [**af**]*<ent>* **for hvilke det betyder at** *<ent-test>* *<num-attr>* **totalt for alle** [**af**]*<ent>* **den totale sum af** [**alle**]*<ent-attr>* **total sum for alle***<ent-attr>* **total for alle** *<ent>***,** *<attr>*	Henter summen af alle instanser af en ***entity-level***-variabel.
InstansSumHvis(*<relationship>*, *<number-attr>*, *<condition>*) **den totale sum af alle***<ent-attr>* **kun hvor** *<condition>* **den totale sum af** [**alle**]*<ent-*	Henter summen af alle instanser af en ***entity-level***-variabel, hvor det er sandt for ***entity***, at en bestemt boolesk ***attribute*** på ***entity-level*** er sand.

Syntaks	Beskrivelse
attr> **for hvilke det betyder at** *<condition>* **total for alle** *<ent>*, *<attr>* **kun hvor** *<condition>*	
InstansVærdiHvis(*<relationship>*, *<number-attr>*, *<condition>*) **InstansVærdiHvis**(*<relationship>*, *<text-attr>*, *<condition>*) **InstansVærdiHvis**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansVærdiHvis**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstansVærdiHvis**(*<relationship>*, *<time-attr>*, *<condition>*)	Henter en værdi fra en entydig ***entity***-instans, der identificeres fra ***entity***-målinstanserne for ***relationship*** af en betingelse. • Hvis betingelsen identificerer en enkelt ***entity***-målinstans, er værdien den værdi, der beregnes mod den ***entity***-instans. • Hvis mere end én målinstans opfylder betingelsen, returneres ***uncertain***. • Hvis ingen målinstanser opfylder betingelsen og ***relationship*** kendes, er værdien ***uncertain***.
InstansLig(*<instance1>*, *<instance2>*)	Fastlægger, om to instanser af en ***entity*** er den samme instans.
InstansIkkeLig(*<instance1>*, *<instance2>*)	Fastlægger, om to instanser af en ***entity*** ikke er den samme instans.
UdledInstans(*<relationship>*, *<identity>*) *<rel>***(***<identity>***) eksisterer**	Bruges som en konklusion til at udlede, at en ***entity***-instans findes og er medlem af en ***relationship***.

Tidsmæssige ræsonneringsfunktioner(English)

Syntaks	Beskrivelse
IntervalOptællingEnsartede (*<start-date>*, *<end-date>*, *<variable>*) **IntervalOptællingEnsartede** (*<start-date>*, *<end-date>*, *<condition>*)	Tæller antallet af kendte distinkte værdier til en variabel i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv).
IntervalOptællingEnsartedeHvis (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Tæller antallet af kendte distinkte værdier til variablen i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv), og inkluderer kun tidspunkter, hvor et boolesk filter er sandt.
IntervalDagligSum(*<start-date>*, *<end-date>*, *<number-attr>*)	Beregner summen af en valuta- eller talvariabel i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv). ***attribute*** antages at være en daglig mængde.
IntervalDagligSumHvis(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Beregner summen af alle daglige værdier til en valuta- eller talvariabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv) og inkluderer kun tidspunkter, hvor en betingelse er sand.

Syntaks	Beskrivelse
IntervalMaksimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMaksimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMaksimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMaksimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Vælger maks. værdien af en variabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv).
IntervalMaksimumHvis(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMaksimumHvis**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMaksimumHvis**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMaksimumHvis**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Vælger maks. værdien af en variabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv) og inkluderer kun tidspunkter, hvor en betingelse er sand.
IntervalMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Vælger min. værdien af en variabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv).
IntervalMinimumHvis(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMinimumHvis**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMinimumHvis**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMinimumHvis**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Vælger min. værdien af en variabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv) og inkluderer kun tidspunkter, hvor en betingelse er sand.
IntervalVægtetGennemsnit (*<start-date>*, *<end-date>*, *<num-*	Beregner den gennemsnitlige værdi af en valuta- eller talvariabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv) vægtet efter tidsrum, som hver værdi

Syntaks	Beskrivelse
ber-attribute>)	gælder for.
IntervalVægtetGennemsnitHvis(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Beregner den gennemsnitlige værdi af en valuta- eller talvariabel i intervallet fra en start***date*** (inklusiv) til en slut***date*** (eksklusiv) og inkluderer kun tidspunkter, hvor en boolesk betingelse er sand (vægtet efter det tidsrum, som hver værdi gælder for, og hvor filteret er sandt).
IntervalAltid(*<start-date>*, *<end-date>*, *<condition>*)	Returnerer sand, hvis og kun hvis en boolesk betingelse er sand på alle tidspunkter i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv).
IntervalMindstDage(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerer sand, hvis og kun hvis en boolesk betingelse er sand for mindst det angivne antal dage (ikke nødvendigvis fortløbende) i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv).
IntervalEfterfølgendeDage(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerer sand, hvis og kun hvis en boolesk betingelse er sand for mindst det givne antal fortløbende dage i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv).
IntervalSommetider(*<start-date>*, *<end-date>*, *<condition>*)	Returnerer sand, hvis og kun hvis en boolesk betingelse nogensinde er sand i intervallet fra start***date*** (inklusiv) til slut***date*** (eksklusiv).
VærdiVed(*<date>*, *<value>*)	Returnerer værdien af given ***attribute*** på den angivne ***date***.
NårSidste(*<date>*, *<condition>*)	Returnerer ***date***, hvor en boolesk betingelse sidst var sand, idet der ses bagud fra (og inklusiv) en angivet ***date***.
NårNæste(*<date>*, *<condition>*)	Returnerer ***date***, hvor en boolesk betingelse næste gang vil være sand, idet der ses fremad fra (og inklusiv) en angivet ***date***.
Senest()	Returnerer en ***date***-værdi svarende til den senest mulige ***date*** - nemlig en ***date***, der garanteres at ligge efter nogen anden ***date***, som en ***date attribute*** kan have, eller som et udtryk kan evalueres til.
Tidligst()	Returnerer en ***date***-værdi svarende til den tidligst mulige ***date*** - nemlig en ***date***, der garanteres at ligge før nogen anden ***date***, som en ***date attribute*** kan have, eller som et udtryk kan evalueres til.
TidsmæssigDageSiden(*<date>*, *<end-date>*)	Returnerer en talvariabel, der varierer hver dag og er antallet af hele dage siden ***date***.
TidsmæssigUgerSiden(*<date>*, *<end-date>*)	Returnerer en talvariabel, der varierer hver uge og er antallet af fulde uger siden ***date***.
TidsmæssigMånederSiden(*<date>*, *<end-date>*)	Returnerer en talværdi, der varierer hver måned og er antallet af fulde måneder siden ***date***. Bemærk: Hvor den angivne ***date*** ligger efter den 28. dag i måneden og en efterfølgende måned har færre dage end den angivne måned, oprettes ændringspunktet for jubilæumsmåneden på den sidste dag i den måned. Hvis f.eks. den angivne ***date*** er 28., 29., 30. or 31. januar 2007, vil det første ændringspunkt være den 28. februar 2007.

Syntaks	Beskrivelse
TidsmæssigÅrSiden(*<date>*, *<end-date>*)	Returnerer en talvariabel, der varierer hvert år og er antallet af fulde år siden ***date***.
TidsmæssigAltidDage(*<days>*, *<condition>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand, hvis og kun hvis en boolesk betingelse er sand til alle af et givet antal foregående dage, der ikke inkluderer den aktuelle dag.
TidsmæssigEfterfølgendeDage (*<minDays>*, *<days>*, *<condition>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand, hvis og kun hvis en boolesk betingelse er sand til mindst et minimum antal fortløbende dage på et hvilket som helst tidspunkt inden for det angivne antal foregående dage, der ikke inkluderer den aktuelle dag.
TidsmæssigSommetiderDage (*<days>*, *<condition>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand, hvis og kun hvis en boolesk betingelse nogensinde er sand inden for et angivet antal foregående dage, der ikke inkluderer den aktuelle dag.
TidsmæssigEfter(*<date>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand efter en ***date*** og falsk på og før.
TidsmæssigFør(*<date>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand før en ***date*** og falsk på og efter.
TidsmæssigDen(*<date>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand på en ***date*** og falsk før og efter.
TidsmæssigDenEllerEfter (*<date>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand på eller efter en ***date*** og falsk før.
TidsmæssigDenEllerFør(*<date>*)	Returnerer en boolesk ***attribute***, der varierer over tiden og er sand på og før en ***date*** og falsk efter.
TidsmæssigFraStartDato(*<relationship>*, *<date>*, *<value>*)	Returnerer en enkelt tidsmæssig ***attribute*** (på ***entity***-kildeniveau) fra en ***relationship*** og en værdi-***attribute*** på entiteterne med værdier, der får virkning fra en start***date attribute***.
TidsmæssigFraSlutDato(*<relationship>*, *<date>*, *<value>*)	Returnerer en enkelt tidsmæssig ***attribute*** (på ***entity***-kildeniveau) fra en ***relationship*** og en værdi-***attribute*** på entiteterne med værdier, der får virkning indtil en slut***date attribute***.
TidsmæssigFraInterval(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Returnerer en enkelt tidsmæssig ***attribute*** (på ***entity***-kildeniveau) fra en ***relationship*** og en værdi-***attribute*** på entiteterne med værdier, der får virkning fra en start***date attribute*** (inklusiv) indtil en slut***date attribute*** (eksklusiv). Værdien er ***uncertain***, hvis den udløber før den næste start***date***.
TidsmæssigErUgedag (*<startdate>*, *<enddate>*)	Returnerer sand på datoer, der er ugedage, og falsk på datoer, der er weekender, fra den angivne start***date*** (inklusiv) til slut***date*** (eksklusiv). Returnerer ***uncertain*** uden for ***date***-intervallet.
TidsmæssigEnGangPrMåned	Returnerer sand, hvis dagen er lig med dag-i-måned-parameteren og falsk på alle andre

Syntaks	Beskrivelse
(*<startdate>*, *<enddate>*, *<day-ofmonth>*)	dage i måneden fra den angivne start***date*** (inklusiv) til slut***date*** (eksklusiv). Returnerer ***uncertain*** uden for ***date***-intervallet. Når dag-i-måned overskrider antallet af dage i den aktuelle måned, er værdien sand på den sidste dag i den måned, så funktionen returnerer en værdi, der er sand nøjagtig én dag pr. måned.

Funktioner til valideringsbegivenhed(English)

Syntaks	Beskrivelse
Fejl (*<text>*)	En fejlhændelse bruges til at overføre en meddelelse til brugeren og forhindrer brugeren i at fortsætte med en undersøgelse, indtil den betingelse, der udløste fejlen, ikke længere gælder.
Advarsel (*<text>*)	En advarselshændelse bruges til at overføre en meddelelse til brugeren, men tillader, at brugeren fortsætter på trods af den betingelse, der udløste advarslen.

Forældede funktioner(English)

Syntaks	Beskrivelse
KaldTilpassetFunktion (*<A>*, *<B>*)	Returnerer resultatet af et eksternt kald til et kodebibliotek. Kodebiblioteket skal angives til Determinations-programmet, for at det tilpassede funktionskald kan gennemføres.

Logische connectors(English)

Syntaxis	Omschrijving
als	Optionele term die aan het einde van een conclusieregel kan staan waarop een bewijs volgt
en	Logische conjunctie tussen twee ***attributes***
of	Logische disjunctie tussen twee ***attributes***
ofwel **een van** **welke ook** **ten minste een van het volgende is waar** **aan elk van de volgende is voldaan** **minstens een van volgende is waar** **aan minstens een van volgende is voldaan** **een van volgende**	Groepeerelement dat wordt gebruikt met disjuncties waarbij twee of meer ***attributes*** moeten worden gegroepeerd

Syntaxis	Omschrijving
beide **alle** **alle van de volgende zijn waar** **aan alle van de volgende is voldaan** **alle volgenden zijn waar** **aan alle volgenden is voldaan** **alle volgenden** **allen** **beiden**	Groepeerelement dat wordt gebruikt met conjuncties waarbij twee of meer ***attributes*** moeten worden gegroepeerd
anders	Term die aan het einde van een tabel staat om de Anders-clausule aan te geven
is	Term die wordt gebruikt in een legenda-ingang tussen de afgekorte woordgroep en de volledige ***attribute text***

Logische functies(English)

Syntaxis	Omschrijving
het is niet waar dat *<expr>*	Operator waarmee 'waar' wordt geretourneerd als ***attribute*** een waarde heeft die onwaar is
<var> **is zeker** **zeker** *<var>* **het is zeker** [**of niet**]*<expr>*	Operator waarmee 'waar' wordt geretourneerd als ***attribute*** een waarde anders dan ***uncertain*** heeft
<var> **is onzeker** *<var>* **is niet zeker** **onzeker** *<var>* **het is onzeker dat** *<expr>* **het is onzeker** [**of niet**]*<expr>* **het is niet zeker dat** *<expr>* **onzeker**	Operator waarmee 'waar' wordt geretourneerd als de waarde van ***attribute uncertain*** is
<var> **is bekend** *<var>* **is nu bekend** **bekend** *<var>* **het is bekend** [**of niet**]*<expr>* **het is nu bekend** [**of niet**]*<expr>*	Operator waarmee 'waar' wordt geretourneerd als ***attribute*** een waarde heeft
<var> **is** [**nu**] **onbekend** **onbekend** *<var>* **het is** [**nu**] **onbekend** [**of niet**]*<expr>*	Operator waarmee 'waar' wordt geretourneerd als ***attribute*** geen waarde heeft

Syntaxis	Omschrijving
onbekend	

Logische constanten(English)

Syntaxis	Omschrijving
waar	Constante waar-waarde die wordt gebruikt voor tabelregels
onwaar	Constante onwaar-waarde die wordt gebruikt voor tabelregels
onzeker	Constante ***uncertain***-waarde die wordt gebruikt voor tabelregels

Vergelijkingsoperatoren(English)

Syntaxis	Omschrijving
*<lhs>***<***<rhs>* *<lhs>* **is kleiner dan** *<rhs>* *<lhs>* **is kleiner of gelijk aan** *<rhs>* *<lhs>* **is eerder dan** *<rhs>*	Kleiner dan Opmerking: er is geen natuurlijke taalvorm wanneer deze operator wordt gebruikt met numerieke waarden en valutawaarden.
<lhs> **>** *<rhs>* *<lhs>* **is groter dan** *<rhs>* *<lhs>* **is groter dan of gelijk aan** *<rhs>* *<lhs>* **is later dan** *<rhs>*	Groter dan Opmerking: er is geen natuurlijke taalvorm wanneer deze operator wordt gebruikt met numerieke waarden en valutawaarden.
*<lhs>***<=***<rhs>*	Kleiner dan of gelijk aan
<lhs> **>=** *<rhs>*	Groter dan of gelijk aan
*<lhs>***=***<rhs>* *<lhs>* **is gelijk aan** *<rhs>* *<lhs>* **is gelijk aan** *<rhs>*	Is gelijk aan
<lhs> **is niet gelijk aan** *<rhs>* *<lhs>* **<>** *<rhs>*	Niet gelijk aan

Numerieke functies(English)

Syntaxis	Omschrijving
Nummer(*<numText>*)	Zet de opgegeven string om in een getalwaarde.
<x> **+** *<y>*	Wiskundige optelling

Syntaxis	Omschrijving
<x> **-** *<y>*	Wiskundige aftrekking
<lhs> * *<rhs>*	Wiskundige vermenigvuldiging
<lhs> **/** *<rhs>*	Wiskundige deling
<lhs> **** *<rhs>*	Deling gehele getallen
<lhs> **modulo** *<rhs>*	Rest na deling met gehele getallen
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **het grootste van** *<val1>* **en** *<val2>* **de laatste van** *<val1>* **en** *<val2>*	Retourneert de grootste van twee waarden.
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **de kleinste van** *<val1>* **en** *<val2>* **de vroegste van** *<val1>* **en** *<val2>*	Retourneert de kleinste van twee waarden.
Xy(*<x>*, *<y>*) *<val>* **vermeerderd tot de macht van** *<power>*	x tot de macht y
Ex(*<x>*) **e tot de macht van** *<log-val>*	Constante e tot de macht x
Abs(*<x>*) **de absolute waarde van** *<val>* **|** *<val>* **|**	Absolute waarde van x
Ln(*<x>*) **de natuurlijke logaritme van** *<log-val>*	Natuurlijke logaritme van x
Log(*<x>*) **de logaritme met grondtal 10** *<log-val>*	Logaritme met grondtal 10 van x
VierkantsWortel(*<x>*) **de vierkante wortel van** *<val>*	Vierkantswortel van x
Afronden(*<x>*, *<n>*) *<val>* **afgerond op** *<num_places>* **decimalen na de komma**	Rondt x af op n decimalen.
Afkappen(*<x>*, *<n>*) *<val>* **getrunceerd op** *<num_places>* **decimalen na de komma**	x afgekapt af op n decimalen
Sin(*<x>*)	Sinus van x
Cos(*<x>*)	Cosinus van x
Tan(*<x>*)	Tangens van x

Syntaxis	Omschrijving
Asin(*<x>*)	Boogsinus van x
Acos(*<x>*)	Boogcosinus van x
Atan(*<x>*)	Boogtangens van x

Datumfuncties(English)

Syntaxis	Omschrijving
HuidigeDatum() **de huidige datum**	Retourneert de huidige ***date*** aan het begin van de sessie.
Datum(*<text>*)	Zet de opgegeven string om in een ***date***-waarde.
MakenDatum(*<year>*, *<month>*, *<day>*)	Retourneert een ***date*** die bestaat uit het jaar, de maand en de dag die zijn opgegeven.
ExtraherenDag(*<date/datetime>*)	Retourneert de dagcomponent van een ***date/datetime attribute***.
ExtraherenMaand(*<date/datetime>*)	Retourneert de maandcomponent van een ***date/datetime attribute***.
ExtraherenJaar(*<date/datetime>*)	Retourneert de jaarcomponent van een ***date/datetime attribute***.
VolgendeDagVanWeek(*<date/datetime>*, *<day>*) **de volgende maandag op of na** *<from-date>* **de maandag op of voor** *<from-date>* **de volgende dinsdag op of na** *<from-date>* **de dinsdag op of voor** *<from-date>* **de volgende woensdag op of na** *<from-date>* **de woensdag op of voor** *<from-date>* **de volgende donderdag op of na** *<from-date>* **de donderdag op of voor** *<from-date>* **de volgende vrijdag op of**	Retourneert de ***date*** van de volgende dag van de week op of na een ***date*** (afhankelijk van de gebruikte syntaxis).

Syntaxis	Omschrijving
na *<from-date>* **de vrijdag op of voor** *<from-date>* **de volgende zaterdag op of na** *<from-date>* **de zaterdag op of voor** *<from-date>* **de volgende zondag op of na** *<from-date>* **de zondag op of voor** *<from-date>*	
VolgendeDatum(*<date>*, *<day>*, *<month>*) **de startdatum van het voorgaande belastingjaar van de UK op of voor** *<from-date>* **de einddatum van het volgende belastingjaar van de UK op of na** *<from-date>*	Retourneert de volgende instantie van de gegeven dag en maand na ***date***.
ToevoegenDagen(*<date/datetime>*, *<num_days>*) **de datum** *<num_days>* **dagen na** *<date>* **de datum** *<num_days>* **dagen voor** *<date>*	Telt een aantal dagen op bij een ***date*** of trekt het ervan af. Bij gebruik van de beknopte syntactische vorm moet het getal een positief geheel getal zijn om dagen bij de ingevoerde ***date*** op te tellen, of een negatief getal om dagen van de ingevoerde ***date*** af te trekken.
ToevoegenWeken(*<date/datetime>*, *<num_weeks>*) **de datum** *<num_weeks>* **weken na** *<date>* **de datum** *<num_weeks>* **weken voor** *<date>*	Telt een aantal weken op bij een ***date***. Bij gebruik van de beknopte syntactische vorm moet het getal een positief geheel getal zijn om weken bij de ingevoerde ***date*** op te tellen.
ToevoegenMaanden(*<date/datetime>*, *<num_months>*) **de datum** *<num_months>* **maanden na** *<date>* **de datum** *<num_months>* **maanden voor** *<date>*	Telt een aantal maanden op bij een ***date***. Bij gebruik van de beknopte syntactische vorm moet het getal een positief geheel getal zijn om maanden bij de ingevoerde ***date*** op te tellen.
ToevoegenJaren(*<date/datetime>*, *<num_years>*) **de datum** *<num_years>*	Telt een aantal jaren op bij een ***date***. Bij gebruik van de beknopte syntactische vorm moet het getal een positief geheel getal zijn om jaren bij de ingevoerde ***date*** op te tellen.

Syntaxis	Omschrijving
jaren na *<date>* **de datum***<num_years>* **jaren voor** *<date>*	
WeekdagTelling(*<date1>*, *<date2>*) **het aantal weekdagen (inclusief) tussen** *<date1>* **en** *<date2>*	Telt het aantal weekdagen tussen ***date***1 en ***date***2, met andere woorden, het aantal dagen tussen maandag en vrijdag. Opmerking: de eerste ***date*** is inclusief, de laatste ***date*** exclusief.
JaarBegin(*<date/datetime>*) **de eerste dag van het jaar waarin** *<from-date>* **valt**	Retourneert de eerste ***date*** in het jaar waarin een ***date*** valt.
JaarEinde(*<date/datetime>*) **de laatste dag van het jaar waarin** *<from-date>* **valt**	Retourneert de laatste ***date*** in het jaar waarin een ***date*** valt.
DagVerschil(*<date/datetime1>*, *<date/datetime2>*) **het aantal dagen vanaf** *<date1>* **tot** *<date2>*	Retourneert het aantal hele dagen tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
DagVerschilInclusief (*<date/datetime1>*, *<date/datetime2>*) **het aantal dagen (inclusief) vanaf** *<date1>* **tot** *<date2>*	Retourneert het aantal hele dagen (inclusief) tussen ***date/datetime1*** en ***date/datetime2***. Bij deze berekening worden beide eindpunten opgenomen. Als de datums gelijk zijn, is het resultaat 1. De volgorde van de twee datums heeft geen invloed op het resultaat.
DagVerschilExclusief (*<date/datetime1>*, *<date/datetime2>*) **het aantal dagen (exclusief) vanaf***<date1>* **tot** *<date2>*	Retourneert het aantal hele dagen (exclusief) tussen ***date/datetime1*** en ***date/datetime2***. Bij deze berekening worden beide eindpunten uitgesloten. Als de datums gelijk zijn, is het resultaat 0. De volgorde van de twee datums heeft geen invloed op het resultaat.
WeekVerschil(*<date/datetime1>*, *<date/datetime2>*) **het aantal weken vanaf** *<date1>* **tot** *<date2>*	Retourneert het aantal hele verstreken weken tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
WeekVerschilInclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het inclusieve aantal hele verstreken weken tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
WeekVerschilExclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het exclusieve aantal hele verstreken weken tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.

Syntaxis	Omschrijving
MaandVerschil(*<date/datetime1>*, *<date/datetime2>*) **het aantal maanden vanaf** *<date1>* **tot** *<date2>*	Retourneert het aantal hele verstreken maanden tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
MaandVerschilInclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het aantal hele inclusieve verstreken maanden tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
MaandVerschilExclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het aantal hele exclusieve verstreken maanden tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
JaarVerschil(*<date/datetime1>*, *<date/datetime2>*) **het aantal jaren (inclusief) tussen** *<date1>* **en** *<date2>* **het aantal hele jaren dat** *<date2>* **is na** *<date1>*	Retourneert het aantal jaren tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
JaarVerschilInclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het inclusieve aantal jaren tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.
JaarVerschilExclusief (*<date/datetime1>*, *<date/datetime2>*)	Retourneert het exclusieve aantal jaren tussen ***date/datetime1*** en ***date/datetime2***. De volgorde van de twee datums heeft geen invloed op het resultaat.

Tijd-van-de-dagfuncties(English)

Syntaxis	Omschrijving
TijdVanDag(*<text>*)	Zet de gegeven string om in een tijdstip.
ExtraherenSeconde(*<time/datetime>*)	Retourneert de secondecomponent van een ***timeofday/datetime attribute***.
ExtraherenMinuut(*<time/datetime>*)	Retourneert de minuutcomponent van een ***timeofday/datetime attribute***.
ExtraherenUur(*<time/datetime>*)	Retourneert de uurcomponent van een ***timeofday/datetime attribute***.

Datum- en tijdfuncties(English)

Syntaxis	Omschrijving
HuidigeDatumTijd()	Retourneert de huidige ***date*** en tijd aan het begin van de sessie.
DatumTijd(*<text>*)	Zet de opgegeven string om in een ***datetime***-waarde.

Syntaxis	Omschrijving
SamenvoegenDatumTijd (*<date>*, *<time>*)	Stelt de tijd van ***date*** door de ***date*** en het tijdstip samen te voegen.
SecondeVerschil(*<datetime1>*, *<datetime2>*) **SecondeVerschil** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het aantal seconden tussen ***datetime1*** en ***datetime2***.
SecondeVerschilInclusief (*<datetime1>*, *<datetime2>*) **SecondeVerschilInclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het inclusieve aantal seconden tussen ***datetime1*** en ***datetime2***.
SecondeVerschilExclusief (*<datetime1>*, *<datetime2>*) **SecondeVerschilExclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het exclusieve aantal seconden tussen ***datetime1*** en ***datetime2***.
MinuutVerschil(*<datetime1>*, *<datetime2>*) **MinuutVerschil** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het aantal minuten tussen ***datetime1*** en ***datetime2***.
MinuutVerschilInclusief (*<datetime1>*, *<datetime2>*) **MinuutVerschilInclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het inclusieve aantal minuten tussen ***datetime1*** en ***datetime2***.
MinuutVerschilExclusief (*<datetime1>*, *<datetime2>*) **MinuutVerschilExclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het exclusieve aantal minuten tussen ***datetime1*** en ***datetime2***.
UurVerschil(*<datetime1>*, *<datetime2>*) **UurVerschil** (*<timeOfDay1>*,	Retourneert het aantal uren tussen ***datetime1*** en ***datetime2***.

Syntaxis	Omschrijving
<timeOfDay2>)	
UurVerschilInclusief (*<datetime1>*, *<datetime2>*) **UurVerschilInclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het inclusieve aantal uren tussen ***datetime1*** en ***datetime2***.
UurVerschilExclusief (*<datetime1>*, *<datetime2>*) **UurVerschilExclusief** (*<timeOfDay1>*, *<timeOfDay2>*)	Retourneert het exclusieve aantal uren tussen ***datetime1*** en ***datetime2***.
ExtraherenDatum(*<datetime>*)	Extraheert de ***date*** uit een ***datetime attribute***.
ExtraherenTijdstipDag (*<datetime>*)	Extraheert het tijdstip uit een ***datetime attribute***. Hiermee kan de waarde van een ***timeofday attribute*** worden ingesteld op de tijd waarop de regel wordt uitgevoerd door de tijd te extraheren uit de huidige ***date*** en tijd.
ToevoegenUren(*<datetime>*, *<num_hours>*) **ToevoegenUren** (*<timeOfDay>*, *<num_hours>*)	Telt een aantal uren op bij een ***date***-tijd.
ToevoegenMinuten(*<datetime>*, *<num_minutes>*) **ToevoegenMinuten** (*<timeOfDay>*, *<num_minutes>*)	Telt een aantal minuten op bij een ***date***-tijd.
ToevoegenSeconden (*<datetime>*, *<num_seconds>*) **ToevoegenSeconden** (*<timeOfDay>*, *<num_seconds>*)	Telt een aantal seconden op bij een ***date***-tijd.

Tekstfuncties(English)

Syntaxis	Omschrijving
<text1> **&** *<text2>*	Combineert ***text1*** met ***text2*** enzovoort en vormt hiermee een enkele ***text***-waarde. Opmerking: u kunt variabelen van elk willekeurig type gebruiken. Waarden worden ingedeeld met het indelingsprogramma dat is geïnstalleerd in de regelsessie.
de reeks van *<text1>* **&** *<text2>*	Combineert ***text1*** met ***text2*** enzovoort en vormt hiermee een enkele ***text***-waarde. Opmerking: u kunt variabelen van elk willekeurig type gebruiken. Waarden worden ingedeeld met het indelingsprogramma dat is geïnstalleerd in de regelsessie.
Bevat(*<text>*, *<substring>*) *<text>* **bevat** *<substring>*	Retourneert een booleaanse waarde die aangeeft of de gegeven ***text***-waarde de gegeven ***text***-substring bevat. De ***text***-vergelijking is niet hoofdlettergevoelig.
EindigtMet(*<text>*, *<substring>*) *<text>* **eindigt met** *<substring>*	Retourneert een booleaanse waarde die aangeeft of de gegeven ***text***-waarde eindigt met de gegeven ***text***-substring. De ***text***-vergelijking is niet hoofdlettergevoelig.
IsGetal(*<text>*) *<text>* **is een getal**	Retourneert een booleaanse waarde die aangeeft of de gegeven ***text***-waarde een geldig getal vertegenwoordigt.
Lengte(*<text>*)	Retourneert de tekenlengte van de gegeven ***text***-waarde.
BegintMet(*<text>*, *<substring>*) *<text>* **begint met** *<substring>*	Retourneert een booleaanse waarde die aangeeft of de gegeven ***text***-waarde begint met de gegeven ***text***-substring. De ***text***-vergelijking is niet hoofdlettergevoelig.
Subtekenreeks (*<text>*, *<offset>*, *<length>*)	Retourneert de substring van ***text*** die begint bij de gegeven verschuiving en de de opgegeven lengte in tekens heeft. Als het einde van de string wordt bereikt, worden minder tekens geretourneerd.
Tekst(*<number>*) **Tekst**(*<date>*) **Tekst**(*<datetime>*) **Tekst**(*<timeOfDay>*)	Zet het opgegeven getal of ***date attribute*** om in een ***text***-waarde.

Entiteits- en relatiefuncties(English)

Syntaxis	Omschrijving
Voor(*<relationship>*, *<Exp>*) **in het geval van** *<ent>***,** *<attr>* *<val>***, als geldt dat** *<ent>*	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-een'-, 'veel-op-een'- of 'veel-op-veel'-***relationship*** waarbij slechts één conditie is.
VoorBereik(*<relationship>*, *<alias>*)	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-een'-, 'veel-op-een'- of 'veel-op-veel'-***relationship*** waarbij er een of meer condities zijn.

Syntaxis	Omschrijving
VoorBereik(*<relationship>*)	
VoorAlle(*<relationship>*, *<Exp>*) **elk van** *<ent-attr>* **voor elk van** *<ent>***,** *<attr>* **voor alle van** *<ent>***,** *<attr>*	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-veel'- of 'veel-op-veel'-***relationship*** wanneer u moet vaststellen of alle leden van de doel-***entity***-groep aan de regel moeten voldoen. Deze vorm wordt gebruikt wanneer de regel slechts één voorwaarde bevat.
VoorAlleBereiken(*<relationship>*) **VoorAlleBereiken**(*<relationship>*, *<alias>*)	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-veel'- of 'veel-op-veel'-***relationship*** wanneer u moet vaststellen of alle leden van de doel-***entity***-groep aan de regel moeten voldoen. Deze vorm wordt gebruikt wanneer de regel een of meer voorwaarden bevat.
Bestaat(*<relationship>*, *<Exp>*) **ten minste één van** *<ent-attr>* **voor ten minste één van** *<ent>***,** *<attr>*	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-veel'- of 'veel-op-veel'-***relationship*** wanneer u moet vaststellen of leden van de doel-***entity***-groep aan de regel moeten voldoen. Deze vorm wordt gebruikt wanneer de regel slechts één voorwaarde bevat.
BestaatBereik(*<relationship>*) **BestaatBereik**(*<relationship>*, *<alias>*)	Hiermee wordt van de ene ***entity*** naar de andere ***entity*** verwezen in een 'een-op-veel'- of 'veel-op-veel'-***relationship*** wanneer u moet vaststellen of leden van de doel-***entity***-groep aan de regel moeten voldoen. Deze vorm wordt gebruikt wanneer de regel een of meer voorwaarden bevat.
IsLidVan(*<target>*, *<relationship>*) **IsLidVan**(*<target>*, *<alias>*, *<relationship>*)	Hiermee kan worden vastgesteld of een ***entity***-instantie lid is van een ***relationship***. Wordt gebruikt als een conditie om te testen of een ***entity***-instantie een doel is van een ***relationship*** waarvoor een tweede ***entity***-instantie de bron is.
IsGeenLidVan(*<target>*, *<relationship>*)	Wordt gebruikt als een conditie om te testen of een ***entity***-instantie geen doel is van een ***relationship*** waarvoor een tweede ***entity***-instantie de bron is.
InstantieAantal (*<relationship>*) **het aantal van** *<ent>*	Telt het aantal instanties dat bestaat voor een ***entity***.
InstantieAantalAls(*<relationship>*, *<Exp>*) **het aantal van** *<ent>* **waarvoor geldt dat** *<condition>*	Telt het aantal instanties van een ***entity*** waarvoor een bepaalde ***entity-level attribute*** een bepaalde waarde heeft.
InstantieMaximum(*<relationship>*, *<number-attr>*) **InstantieMaximum**(*<relationship>*, *<date-attr>*) **InstantieMaximum**(*<relationship>*, *<datetime-attr>*) **InstantieMaximum**(*<rela-*	Haalt de hoogste/meest recente waarde op van een ***entity-level***-variabele voor alle instanties van de ***entity***.

Syntaxis	Omschrijving
tionship>, *<time-attr>*) *<date-attr>* **die de laatste is voor alle** [**van**]*<ent>* *<max-attr>* **die de grootste is voor alle** [**van**]*<ent>* **de laatste van alle** *<ent-attr>* **de grootste van alle** *<ent-attr>* **de grootste van** *<attr>* **voor alle** [**of**]*<ent>* **de grootste van alle** *<attr>* **voor** [**alle**]*<ent>* **de laatste van alle** *<attr>* **voor** *<ent>*	
InstantieMaximumAls(*<relationship>*, *<number-attr>*, *<condition>*) **InstantieMaximumAls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstantieMaximumAls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstantieMaximumAls**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **die de laatste is voor alle** [**van**]*<ent>* **waarvoor geldt dat** *<ent-test>* *<max-attr>* **die de grootste is voor alle** [**van**]*<ent>* **waarvoor geldt dat** *<ent-test>* **de laatste van alle** *<ent-attr>* **waarvoor geldt dat** *<ent-test>* **de grootste van alle** *<ent-attr>* **waarvoor geldt dat** *<ent-test>* **de grootste van** *<attr>* **voor alle** [**of**]*<ent>* **waarvoor geldt dat** *<ent-test>*	Haalt de hoogste/meest recente waarde op van een ***entity-level***-variabele voor alle instanties van de ***entity*** waarvoor een bepaalde ***entity-level attribute*** een bepaalde waarde heeft.
InstantieMinimum(*<relationship>*, *<number-attr>*)	Haalt de laagste/minst recente waarde op van een ***entity-level***-variabele voor alle instanties van de ***entity***.

Syntaxis	Omschrijving
InstantieMinimum(*<relationship>*, *<date-attr>*) **InstantieMinimum**(*<relationship>*, *<datetime-attr>*) **InstantieMinimum**(*<relationship>*, *<time-attr>*) *<date-attr>* **die de vroegste is voor alle** [**van**]*<ent>* *<attr>* **die het minste is voor alle** [**van**]*<ent>* **de minste van alle** *<ent-attr>* **de minste van alle** *<attr>* **voor** [**alle**]*<ent>* **de vroegste van alle** *<attr>* **voor** *<ent>*	
InstantieMinimumAls(*<relationship>*, *<number-attr>*, *<condition>*) **InstantieMinimumAls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstantieMinimumAls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstantieMinimumAls**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **die de vroegste is voor alle** [**van**]*<ent>* **waarvoor geldt dat** *<ent-test>* *<num-attr>* **die het minste is voor alle** [**van**]*<ent>* **waarvoor geldt dat** *<ent-test>* **de minste van alle** *<ent-attr>* **waarvoor geldt dat** *<ent-test>* **de minste van alle** *<attr>* **voor** *<ent>* **waarvoor geldt dat** *<ent-test>* **de vroegste van alle** *<attr>* **voor** *<ent>* **waarvoor geldt dat** *<ent-test>*	Haalt de laagste/minst recente waarde op van een ***entity-level***-variabele voor alle instanties van de ***entity*** waarvoor een bepaalde ***entity-level attribute*** een bepaalde waarde heeft.
InstantieTotaal	Haalt de som op van alle instanties van een ***entity-level***-variabele.

Syntaxis	Omschrijving
(*<relationship>*, *<number-attr>*) *<num-attr>* **bij elkaar opgeteld voor alle** [**van**]*<ent>* **waarvoor geldt dat** *<ent-test>* *<num-attr>* **is in totaal voor alle**[**van**]*<ent>* **het totale aantal van** [**alle**]*<ent-attr>* **het totaal voor alle***<ent-attr>* **totaal voor alle** *<ent>*, *<attr>*	
InstantieTotaalAls(*<relationship>*, *<number-attr>*, *<condition>*) **het totale aantal van alle***<ent-attr>* **alleen waar** *<condition>* **het totale aantal van** [**alle**]*<ent-attr>* **waarvoor geldt dat** *<condition>* **totaal voor alle***<ent>*, *<attr>* **alleen waar** *<condition>*	Haalt de som op van alle instanties van een ***entity-level***-variabele waarbij voor de ***entity*** geldt dat een bepaalde booleaanse waarde ***attribute*** van de ***entity-level*** waar is.
InstantieWaardeAls(*<relationship>*, *<number-attr>*, *<condition>*) **InstantieWaardeAls**(*<relationship>*, *<text-attr>*, *<condition>*) **InstantieWaardeAls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstantieWaardeAls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstantieWaardeAls**(*<relationship>*, *<time-attr>*, *<condition>*)	Haalt een waarde op van een unieke ***entity***-instantie, met een conditie geïdentificeerd vanuit de doel-***entity***-instanties van een ***relationship***. • Als de conditie een enkele doel-***entity***-instantie identificeert, wordt de waarde berekend tegen deze ***entity***-instantie. • Als meer dan één doelinstantie aan de conditie voldoet, wordt ***uncertain*** geretourneerd. • Als er geen doelinstantie aan de conditie voldoet en de ***relationship*** is bekend, is de waarde ***uncertain***.
InstantieGelijkaan (*<instance1>*, *<instance2>*)	Stelt vast of twee instanties van een ***entity*** dezelfde instantie zijn.
InstantieNietGelijkaan (*<instance1>*, *<instance2>*)	Stelt vast of twee instanties van een ***entity*** niet dezelfde instantie zijn.
AfleidenInstantie(*<rela-*	Deze optie wordt als conclusie gebruikt om af te leiden dat er een ***entity***-instantie bestaat

Syntaxis	Omschrijving
tionship>, *<identity>*) *<rel>***(***<identity>***) bestaat**	en dat deze lid is van een ***relationship***.

Temporele-redeneringsfuncties(English)

Syntaxis	Omschrijving
IntervalIndividuTelling(*<start-date>*, *<end-date>*, *<variable>*) **IntervalIndividuTelling**(*<start-date>*, *<end-date>*, *<condition>*)	Telt het aantal bekende afzonderlijke waarden voor de variabele in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief).
IntervalIndividuTellingAls(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Telt het aantal bekende afzonderlijke waarden voor de variabele in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief), met alleen het aantal keren dat een booleaans filter waar is.
IntervalDagelijksTotaal(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalDagelijkseSom(***<start>***,** *<end>***,** *<var>***)**	Berekent de som van een valuta- of getalvariabele in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief). Het ***attribute*** is naar verwachting een dagelijkse hoeveelheid.
IntervalDagelijksTotaalAls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalDagelijkseSom(***<start>***,** *<end>***,** *<var>***,** *<condition>***)**	Berekent de som van alle dagelijkse waarden voor een valuta- of getalvariabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief), met alleen de keren dat een conditie waar is.
IntervalMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Selecteert de maximumwaarde van een variabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief).
IntervalMaximumAls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMaximumAls**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMaximumAls**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMaximumAls**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Selecteert de maximumwaarde van een variabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief), met alleen de keren dat een conditie waar is.

Syntaxis	**Omschrijving**
IntervalMaximum(*<start>***,** *<end>***,** *<var>***,** *<condition>***)**	
IntervalMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Selecteert de minimumwaarde van een variabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief).
IntervalMinimumAls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMinimumAls**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMinimumAls**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMinimumAls**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*) **IntervalMinimum(** *<start>***,** *<end>***,** *<var>***,** *<condition>***)**	Selecteert de minimumwaarde van een variabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief), met alleen de keren dat een conditie waar is.
IntervalGewogenGemiddelde (*<start-date>*, *<end-date>*, *<number-attribute>*) **IntervalGewogenGemiddelde** **(** *<start>***,** *<end>***,** *<var>***)**	Berekent de gemiddelde waarde van een valuta- of getalvariabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief) gewogen per periode waarop elke waarde van toepassing is.
IntervalGewogenGemiddeldeAls (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*) **IntervalGewogenGemiddelde** **(** *<start>***,** *<end>***,** *<var>***,** *<condition>***)**	Berekent de gemiddelde waarde van een valuta- of getalvariabele in het interval vanaf een begin-***date*** (inclusief) tot een eind-***date*** (exclusief) met alleen de keren dat een booleaanse conditie waar is (gewogen per periode waarop elke waarde van toepassing is en waarbij het filter waar is).
IntervalAltijd(*<start-date>*, *<end-date>*, *<condition>*)	Retourneert 'waar' uitsluitend als een booleaanse conditie altijd waar is in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief).
IntervalTenMinsteDagen(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Retourneert 'waar' uitsluitend als een booleaanse conditie waar voor ten minste het opgegeven aantal dagen (niet per se opeenvolgend) in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief).
IntervalOpeenvolgendeDagen (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Retourneert 'waar' uitsluitend als een booleaanse conditie waar voor ten minste een gegeven aantal opeenvolgende dagen in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief).

Syntaxis	Omschrijving
IntervalSoms(*<start-date>*, *<end-date>*, *<condition>*)	Retourneert 'waar' uitsluitend als een booleaanse conditie ooit waar is in het interval vanaf de begin-***date*** (inclusief) tot de eind-***date*** (exclusief).
WaardeOp(*<date>*, *<value>*)	Retourneert de waarde van het gegeven ***attribute*** bij de opgegeven ***date***.
WanneerLaatst(*<date>*, *<condition>*)	Retourneert de datum ***date*** waarop een booleaanse conditie voor het laatst waar was, waarbij wordt teruggekeken vanaf (inclusief) een opgegeven ***date***.
WanneerVolgende(*<date>*, *<condition>*)	Retourneert de datum ***date*** waarop een booleaanse conditie opnieuw waar is, waarbij wordt vooruitgekeken vanaf (inclusief) een opgegeven ***date***.
Laatst()	Retourneert een ***date***-waarde equivalent aan de laatste mogelijke ***date***, namelijk een ***date*** die gegarandeerd later is dan enige andere ***date*** die een ***date attribute*** kan aannemen of waarnaar een expressie kan worden geëvalueerd.
Vroegst()	Retourneert een ***date***-waarde equivalent aan de eerst mogelijke ***date***, namelijk een ***date*** die gegarandeerd eerder is dan enige andere ***date*** die een ***date attribute*** kan aannemen of waarnaar een expressie kan worden geëvalueerd.
TemporeelDagenSinds(*<date>*, *<end-date>*)	Retourneert een getalvariabele die elke dag varieert en en het aantal volledige dagen aangeeft sinds de ***date***.
TemporeelWekenSinds(*<date>*, *<end-date>*)	Retourneert een getalvariabele die elke week varieert en en het aantal volledige weken aangeeft sinds de ***date***.
TemporeelMaandenSinds(*<date>*, *<end-date>*)	Retourneert een getalvariabele die elke maand varieert en het aantal volledige maanden sinds de ***date*** aangeeft. Opmerking: in de gevallen waarin de opgegeven ***date*** later is dan de 28e dag van de maand en de volgende maand heeft minder dagen dan de opgegeven maand, wordt het wijzigingspunt voor de jubileummaand gemaakt op de laatste dag van deze maand. Als de opgegeven ***date*** bijvoorbeeld 28, 29, 30 of 31 januari 2007 is, is het eerste wijzigingspunt 28 februari 2007.
TemporeelJarenSinds(*<date>*, *<end-date>*)	Retourneert een getalvariabele die elk jaar varieert en en het aantal volledige jaren aangeeft sinds de ***date***.
TemporeelAltijdDagen(*<days>*, *<condition>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en dat uitsluitend waar is als een booleaanse conditie waar is voor alle waarden van een gegeven aantal voorafgaande dagen, de huidige dag niet inbegrepen.
TemporeelOpeenvolgendDagen(*<minDays>*, *<days>*, *<condition>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en dat uitsluitend waar is als een booleaanse conditie waar is gedurende ten minste een minimum aantal opeenvolgende dagen op elk willekeurig moment binnen het eerder ingestelde aantal dagen, de huidige dag niet inbegrepen.
TemporeelSomsDagen(*<days>*, *<condition>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en dat uitsluitend waar is als een booleaanse conditie ooit waar is binnen een opgegeven aantal voorafgaande dagen, de huidige dag niet inbegrepen.
TemporeelNa(*<date>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en waar is

Syntaxis	Omschrijving
	na een ***date*** en hierop en hiervóór onwaar is.
TemporeelVoor(*<date>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en waar is vóór een ***date*** en hierop en hierna onwaar is.
TemporeelOp(*<date>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en waar is op een ***date*** en hiervóór en hierna onwaar is.
TemporeelOpOfNa(*<date>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en waar is op of na een ***date*** en hiervóór onwaar is.
TemporeelOpOfVoor(*<date>*)	Retourneert een booleaans ***attribute*** dat in de loop van de tijd varieert en waar is op en vóór een ***date*** en hierna onwaar is.
TemporeelGebaseerdOpBegindatum (*<relationship>*, *<date>*, *<value>*)	Retourneert een enkel temporeel ***attribute*** (op het niveau van de bron-***entity***) op basis van een ***relationship*** en een waarde-***attribute*** voor de entiteiten met waarden die van kracht worden vanaf een begin-***date attribute***.
TemporeelGebaseerdOpEinddatum (*<relationship>*, *<date>*, *<value>*)	Retourneert een enkel temporeel ***attribute*** (op het niveau van de bron-***entity***) op basis van een ***relationship*** en een waarde-***attribute*** voor de entiteiten met waarden die van kracht worden tot een eind-***date attribute***.
TemporeelGebaseerdOpReeks(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Retourneert een enkel temporeel ***attribute*** (op het niveau van de bron-***entity***) op basis van een ***relationship*** en een waarde-***attribute*** voor de entiteiten met waarden die van kracht worden vanaf een begin-***date attribute*** tot een eind-***date attribute*** (exclusief). De waarde is ***uncertain*** als deze verloopt vóór de volgende begin-***date***.
TemporeelIsWeekdag(*<startdate>*, *<enddate>*)	Retourneert 'waar' voor werkdagen en 'onwaar' voor weekends vanaf de opgegeven begin-***date*** (inclusief) tot de eind-***date*** (exclusief). Retourneert ***uncertain*** buiten het ***date***-bereik.
TemporeelEensPerMaand (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Retourneert waar als de dag gelijk is aan de dag-van-maandparameter en onwaar op alle andere dagen van de maand vanaf de opgegeven begin-***date*** (inclusief) naar de eind-***date*** (exclusief). Retourneert ***uncertain*** buiten het ***date***-bereik. Als de dag-van-de-maandparameter het aantal dagen in de huidige maand overschrijdt, is de waarde waar op de laatste dag van de maand, zodat de functie een waarde retourneert die exact één dag per maand waar is.

Validatiegebeurtenisfuncties(English)

Syntaxis	Omschrijving
Fout(*<text>*)	Een foutgebeurtenis wordt gebruikt om een bericht aan de gebruiker door te geven, zodat de gebruiker pas kan doorgaan met een onderzoek totdat de conditie waardoor de fout is getriggerd, niet meer van toepassing is.
Waarschuwing	Een waarschuwingsgebeurtenis wordt gebruikt om een bericht aan de gebruiker door te geven, maar biedt

Syntaxis	Omschrijving
(*<text>*)	de gebruiker wel de gelegenheid door te gaan, ondanks de conditie waardoor de waarschuwing is getriggerd.

Verouderde functies(English)

Syntaxis	Omschrijving
AanroepAangepasteFunctie (*<A>*, *<B>*)	Retourneert het resultaat van een externe aanroep naar een codebibliotheek. De aangepaste functieaanroep slaagt alleen wanneer de codebibliotheek wordt opgegeven voor de vaststellings-engine.

Logical connectors

Syntax	Description
if	Optional term that can appear at the end of a conclusion line that has a following proof
and	Logical conjunction between two ***attributes***
or	Logical disjunction between two ***attributes***
either **one of** **any** **at least one of the following is true** **any of the following are satisfied**	Grouping element used with disjunctions where two or more ***attributes*** need to be grouped
both **all** **all of the following are true** **all of the following are satisfied**	Grouping element used with conjunctions where two or more ***attributes*** need to be grouped
otherwise	Term that appears at the end of a table rule to indicate the otherwise clause
is	Term that is used in a legend entry between the abbreviated phrase and the full ***attribute text***

Logical functions

Syntax	Description
it is not true that *<expr>*	Operator used to return true if ***attribute*** has a value which is false
<var> **is certain**	Operator used to return true if ***attribute*** has a value which is not

Syntax	Description
it is certain whether [**or not**]*<expr>*	***uncertain***
<var> **is uncertain** *<var>* **is not certain** **it is uncertain that** *<expr>* **it is uncertain whether** [**or not**]*<expr>* **it is not certain that** *<expr>*	Operator used to return true if ***attribute*** value is ***uncertain***
<var> **is known** *<var>* **is currently known** **it is known whether** [**or not**]*<expr>* **it is currently known whether** [**or not**]*<expr>*	Operator used to return true if ***attribute*** has any value
<var> **is** [**currently**] **unknown** **it is** [**currently**] **unknown whether** [**or not**]*<expr>*	Operator used to return true if ***attribute*** has no value

Logical constants

Syntax	Description
true	Constant true value used for table rules.
false	Constant false value used for table rules.
uncertain	Constant ***uncertain*** value used for table rules.

Comparison operators

Syntax	Description
*<x>***<***<y>* *<x>* **is earlier than** *<y>*	Less than Note: there is no natural language form when this operator is used with numerical and currency values.
<x> **>** *<y>* *<x>* **is later than** *<y>*	Greater than Note: there is no natural language form when this operator is used with numerical and currency values.
*<x>***<=***<y>* *<x>* **is less than or equal to** *<y>* *<x>* **is on or earlier than** *<y>* *<x>* **is at or earlier than** *<y>*	Less than or equal to
<x> **>=** *<y>* *<x>* **is greater than or equal**	Greater than or equal to

Syntax	Description
to *<y>* *<x>* **is on or later than** *<y>* *<x>* **is at or later than** *<y>*	
*<x>***=***<y>* *<x>* **is equal to** *<y>* *<x>* **equals** *<y>*	Equals
<x> **is not equal to** *<y>* *<x>* **<>** *<y>*	Not equal

Numerical functions

Syntax	Description
Number(*<numText>*)	Convert the specified string into a number value
<x> **+** *<y>*	Mathematical addition
<x> **-** *<y>*	Mathematical subtraction
<x> * *<y>*	Mathematical multiplication
<x> **/** *<y>*	Mathematical division
<x> **** *<y>*	Integer division
<x> **modulo** *<y>*	Remainder after integer division
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **the greater of** *<x>* **and** *<y>* **the latest of** *<x>* **and** *<y>*	Returns the greater of two values
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **the lesser of** *<x>* **and** *<y>* **the earliest of** *<x>* **and** *<y>*	Returns the lesser of two values
Xy(*<x>*, *<y>*) *<x>* **raised to the power of** *<y>*	x to the power of y
Ex(*<x>*) **e to the power of** *<x>*	Constant e to the power of x
Abs(*<x>*) **the absolute value of** *<x>* **|***<x>***|**	Absolute value of x

Syntax	Description
Ln(*<x>*) **the natural logarithm of** *<x>*	Natural logarithm of x
Log(*<x>*) **the logarithm base 10 of** *<x>*	Logarithm base 10 of x
Sqrt(*<x>*) **the square root of** *<x>*	Square root of x
Round(*<x>*, *<n>*) *<x>* **rounded to** *<n>* **decimal place** *<x>* **rounded to** *<n>* **decimal places**	Rounds x to n decimal places
Trunc(*<x>*, *<n>*) *<x>* **truncated to** *<n>* **decimal place** *<x>* **truncated to** *<n>* **decimal places**	x truncated to n decimal places
Sin(*<x>*)	Sine of x
Cos(*<x>*)	Cosine of x
Tan(*<x>*)	Tangent of x
Asin(*<x>*)	Arcsine of x
Acos(*<x>*)	Arccosine of x
Atan(*<x>*)	Arctangent of x

Date functions

Syntax	Description
CurrentDate() **the current date**	Returns today's ***date***.
Date(*<text>*)	Converts the specified string into a ***date*** value
MakeDate(*<year>*, *<month>*, *<day>*)	Returns a ***date*** formed from the specified year, month, and day.
ExtractDay(*<date/datetime>*)	Returns the day component of a ***date/datetime attribute***.
ExtractMonth(*<date/datetime>*)	Returns the month component of a ***date/datetime attribute***.
ExtractYear(*<date/datetime>*)	Returns the year component of a ***date/datetime attribute***.

Syntax	Description
NextDayOfTheWeek (*<date/datetime>*, *<day>*) **the next Monday on or after** *<from-date>* **the Monday on or before** *<from-date>* **the next Tuesday on or after** *<from-date>* **the Tuesday on or before** *<from-date>* **the next Wednesday on or after** *<from-date>* **the Wednesday on or before** *<from-date>* **the next Thursday on or after** *<from-date>* **the Thursday on or before** *<from-date>* **the next Friday on or after** *<from-date>* **the Friday on or before** *<from-date>* **the next Saturday on or after** *<from-date>* **the Saturday on or before** *<from-date>* **the next Sunday on or after** *<from-date>* **the Sunday on or before** *<from-date>*	Returns the ***date*** of the next weekday on or after a ***date***.
NextDate(*<date>*, *<day>*, *<month>*) **the previous UK tax year start date on or before** *<from-date>* **the next UK tax year end date on or after** *<from-date>*	NextDateReturns the next instance of the given day and month after a ***date***.
AddDays(*<date/datetime>*, *<num_days>*) **the date** *<num_days>* **days after** *<datetime>*	Adds 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***.

Syntax	Description
the date *<num_days>* **days before** *<datetime>* **the date** *<num_days>* **day after** *<datetime>* **the date** *<num_days>* **day before** *<datetime>* **the time** *<num_days>* **days after** *<datetime>* **the time** *<num_days>* **days before** *<datetime>* **the time** *<num_days>* **day after** *<datetime>* **the time** *<num_days>* **day before** *<datetime>*	
AddWeeks (*<date/datetime>*, *<num_weeks>*) **the date** *<num_weeks>* **weeks after** *<datetime>* **the date** *<num_weeks>* **weeks before** *<datetime>* **the date** *<num_weeks>* **week after** *<datetime>* **the date** *<num_weeks>* **week before** *<datetime>* **the time** *<num_weeks>* **weeks after** *<datetime>* **the time** *<num_weeks>* **weeks before** *<datetime>* **the time** *<num_weeks>* **week after** *<datetime>* **the time** *<num_weeks>* **week before** *<datetime>*	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***.
AddMonths(*<date/datetime>*, *<num_months>*) **the date** *<num_months>* **months after** *<datetime>* **the date** *<num_months>* **months before** *<datetime>* **the date** *<num_months>* **month after** *<datetime>* **the date** *<num_months>*	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***.

Syntax	Description
month before *<datetime>* **the time** *<num_months>* **months after** *<datetime>* **the time** *<num_months>* **months before** *<datetime>* **the time** *<num_months>* **month after** *<datetime>* **the time** *<num_months>* **month before** *<datetime>*	
AddYears(*<date/datetime>*, *<num_years>*) **the date** *<num_years>* **years after** *<datetime>* **the date** *<num_years>* **years before** *<datetime>* **the date** *<num_years>* **year after** *<datetime>* **the date** *<num_years>* **year before** *<datetime>* **the time** *<num_years>* **years after** *<datetime>* **the time** *<num_years>* **years before** *<datetime>* **the time** *<num_years>* **year after** *<datetime>* **the time** *<num_years>* **year before** *<datetime>*	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***.
WeekdayCount(*<date1>*, *<date2>*) **the number of weekdays (inclusive) between** *<date1>* **and** *<date2>*	Counts the number of weekdays between ***date***1 and ***date***2. That is, the number of days falling between Monday and Friday. Note: The earlier ***date*** is inclusive and the later ***date*** is exclusive.
YearStart(*<date/datetime>*) **the first day of the year in which** *<from-date>* **falls**	Returns the first ***date*** in the year in which a ***date*** falls.
YearEnd(*<date/datetime>*) **the last day of the year in which** *<from-date>* **falls**	Returns the last ***date*** in the year in which a ***date*** falls.
DayDifference(*<date/datetime1>*,	Returns the number of whole days between ***date/datetime1*** and ***date/datetime2***. Where the second DATE is earlier than the first DATE, the result is 0.

Syntax	Description
<date/datetime2>) **the number of days from** *<date/datetime1>* **to** *<date/datetime2>*	
DayDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of days (inclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole days (inclusive) between ***date/datetime1*** and ***date/datetime2***. This calculation includes both endpoints. Where the dates are the same, the result is 1. Where the second DATE is earlier than the first DATE, the result is 0.
DayDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of days (exclusive) from** *<date/datetime1>* **to** *<date/datetime2>*	Returns the number of whole days (exclusive) between ***date/datetime1*** and ***date/datetime2***. This calculation excludes both endpoints. Where the dates are the same, the result is 0. Where the second DATE is earlier than the first DATE, the result is also 0.
WeekDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of weeks from** *<date1>* **to** *<date2>*	Returns the number of whole elapsed weeks between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.
WeekDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of weeks (inclusive) from** *<date1>* **to** *<date2>*	Returns the inclusive number of whole elapsed weeks between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.
WeekDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of weeks (exclusive) from** *<date1>* **to** *<date2>*	Returns the exclusive number of whole elapsed weeks between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.
MonthDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of months from** *<date1>* **to** *<date2>*	Returns the number of whole elapsed months between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.

Syntax	Description
MonthDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of months (inclusive) from** *<date1>* **to** *<date2>*	Returns the number of whole inclusive elapsed months between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.
MonthDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of months (exclusive) from** *<date1>* **to** *<date2>*	Returns the number of whole exclusive elapsed months between ***date/datetime1*** and ***date/datetime2***. If the second DATE is earlier than the first DATE, then the result will be 0.
YearDifference(*<date/datetime1>*, *<date/datetime2>*) **the number of whole years which** *<date2>* **is after** *<date1>* **the number of years between** *<date1>* **and** *<date2>*	Returns the number of years between ***date/datetime1*** and ***date/datetime2***. NOTE: The earlier DATE is inclusive and the later DATE is exclusive.
YearDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of years (inclusive) between** *<date1>* **and** *<date2>*	Returns the inclusive number of years between ***date/datetime1*** and ***date/datetime2***.
YearDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **the number of years (exclusive) between** *<date1>* **and** *<date2>*	Returns the exclusive number of years between ***date/datetime1*** and ***date/datetime2***.

Time of day functions

Syntax	Description
TimeOfDay(*<text>*)	Converts the given string into a time of day
ExtractSecond(*<time/datetime>*)	Returns the second component of a ***timeofday/datetime attribute***.

Syntax	Description
ExtractMinute(*<time/datetime>*)	Returns the minute component of a ***timeofday/datetime attribute***.
ExtractHour(*<time/datetime>*)	Returns the hour component of a ***timeofday/datetime attribute***.

Date and time functions

Syntax	Description
CurrentDateTime() **the current date time**	Sets the ***date*** time to the current ***date*** and time.
DateTime(*<text>*)	Converts the specified string into a ***datetime*** value
ConcatenateDateTime (*<date>*, *<time>*) *<date>* **at** *<time-of-day>* *<time-of-day>* **on** *<date>*	Sets the ***date*** time by joining the ***date*** and time of day together.
SecondDifference(*<datetime1>*, *<datetime2>*) **SecondDifference** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds from** *<date1>* **to** *<date2>*	Returns the number of seconds between ***datetime1*** and ***datetime2***.
SecondDifferenceInclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds (inclusive) from** *<date1>* **to** *<date2>*	Returns the inclusive number of seconds between ***datetime1*** and ***datetime2***.
SecondDifferenceExclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of seconds (exclusive) from** *<date1>* **to** *<date2>*	Returns the exclusive number of seconds between ***datetime1*** and ***datetime2***.
MinuteDifference(*<datetime1>*, *<datetime2>*)	Returns the number of minutes between ***datetime1*** and ***datetime2***.

Syntax	Description
MinuteDifference (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes from** *<datetime1>* **to** *<datetime2>*	
MinuteDifferenceInclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes (inclusive) from** *<date1>* **to** *<date2>*	Returns the inclusive number of minutes between ***datetime1*** and ***datetime2***.
MinuteDifferenceExclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of minutes (exclusive) from** *<date1>* **to** *<date2>*	Returns the exclusive number of minutes between ***datetime1*** and ***datetime2***.
HourDifference (*<datetime1>*, *<datetime2>*) **HourDifference** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of hours from** *<date1>* **to** *<date2>*	Returns the number of hours between ***datetime1*** and ***datetime2***.
HourDifferenceInclusive (*<datetime1>*, *<datetime2>*) **HourDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **the number of hours (inclusive) from** *<date1>* **to** *<date2>*	Returns the inclusive number of hours between ***datetime1*** and ***datetime2***.
HourDifferenceExclusive (*<datetime1>*, *<datetime2>*) **HourDifferenceExclusive** (*<timeOfDay1>*,	Returns the exclusive number of hours between ***datetime1*** and ***datetime2***.

Syntax	**Description**
<timeOfDay2>) **the number of hours (exclusive) from** *<date1>* **to** *<date2>*	
ExtractDate(*<datetime>*)	Extracts the ***date*** from a ***datetime attribute***.
ExtractTimeOfDay(*<datetime>*)	Extracts the time of day from a ***datetime attribute***. Can be used to set the value of a ***timeofday attribute*** to the time the rule is executed by extracting the time from the current ***date*** and time.
AddHours(*<datetime>*, *<num_hours>*) **AddHours**(*<timeOfDay>*, *<num_hours>*) **the time** *<num_hours>* **hours after** *<datetime>* **the time** *<num_hours>* **hours before** *<datetime>* **the time** *<num_hours>* **hour after** *<datetime>* **the time** *<num_hours>* **hour before** *<datetime>*	Adds a number of hours to a ***date*** time.
AddMinutes(*<datetime>*, *<num_minutes>*) **AddMinutes**(*<timeOfDay>*, *<num_minutes>*) **the time** *<num_minutes>* **minutes after** *<datetime>* **the time** *<num_minutes>* **minutes before** *<datetime>* **the time** *<num_minutes>* **minute after** *<datetime>* **the time** *<num_minutes>* **minute before** *<datetime>*	Adds a number of minutes to a ***date*** time.
AddSeconds(*<datetime>*, *<num_seconds>*) **AddSeconds**(*<timeOfDay>*, *<num_seconds>*) **the time** *<num_seconds>* **seconds after** *<datetime>* **the time** *<num_seconds>* **seconds before** *<datetime>* **the time** *<num_seconds>*	Adds a number of seconds to a ***date*** time.

Syntax	Description
second after *<datetime>* **the time** *<num_seconds>* **second before** *<datetime>*	

Text functions

Syntax	Description
<text1> **&** *<text2>*	Combines ***text1*** with ***text2*** and so on to form a single ***text*** value
the concatenation of *<text1>* **&** *<text2>*	Combines ***text1*** with ***text2*** and so on to form a single ***text*** value. Note: that you can use variables of any type. Values are formatted using the formatter that is installed in the rule session.
Contains(*<text>*, *<sub-string>*) *<text>* **contains** *<sub-string>*	Returns a boolean value indicating whether the given ***text*** value contains the given ***text*** sub-string. The text comparison is case-insensitive.
EndsWith(*<text>*, *<sub-string>*) *<text>* **ends with** *<sub-string>*	Returns a boolean value indicating whether the given ***text*** value ends with the given ***text*** sub-string. The text comparison is case-insensitive.
IsNumber(*<text>*) *<text>* **is a number**	Returns a boolean value indicating whether the given ***text*** value represents a valid number.
Length(*<text>*) **the length of** *<text>*	Returns the character length of the given ***text*** value.
StartsWith(*<text>*, *<substring>*) *<text>* **starts with** *<sub-string>*	Returns a boolean value indicating whether the given ***text*** value starts with the given ***text*** sub-string. The text comparison is case-insensitive.
Substring(*<text>*, *<offset>*, *<length>*)	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.
Text(*<number>*) **Text**(*<date>*) **Text**(*<datetime>*) **Text**(*<timeOfDay>*)	Convert the specified number or ***date attribute*** into a ***text*** value.

Entity and relationship functions

Syntax	Description
For(*<relationship>*, *<Exp>*) **in the case of** *<relationship>***,** *<attr>* *<val>***, in the case of** *<relationship>*	Used to refer from one ***entity*** to another ***entity*** in a "to-one" ***relationship*** where there is only one condition.
ForScope(*<relationship>*, *<alias>*) **ForScope**(*<relationship>*) **in the case of** *<relationship>* **in the case of** *<relationship>* **(**<alias>**)**	Used to refer from one ***entity*** to another ***entity*** in a "to-one" ***relationship*** where there are one or more conditions.
ForAll(*<relationship>*, *<Exp>*) **each of** *<relationship-attr>* **for each of** *<relationship>***,** *<attr>* **for all of** *<relationship>***,** *<attr>*	Used to refer from one ***entity*** to another ***entity*** in a "to-many" ***relationship***, when you need to determine whether all members of the target ***entity*** group need to satisfy the rule.
ForAllScope(*<relationship>*) **ForAllScope**(*<relationship>*, *<alias>*) **for all of** *<relationship>* **each of** *<relationship>* **for each of** *<relationship>* **for all of** *<relationship>* **(**<alias>**)** **each of** *<relationship>* **(**<alias>**)** **for each of** *<relationship>* **(**<alias>**)**	Used to refer from one ***entity*** to another ***entity*** in a "to-many" ***relationship***, when you need to determine whether all members of the target ***entity*** group need to satisfy the rule. This form is used when there are one or more conditions in the rule.
Exists(*<relationship>*, *<Exp>*) **at least one of** *<relationship-attr>* **for at least one of** *<relationship>***,** *<attr>*	Used to refer from one ***entity*** to another ***entity*** in a "to-many" ***relationship***, when you need to determine whether any members of the target ***entity*** group need to satisfy the rule.
ExistsScope(*<relationship>*) **ExistsScope**(*<relationship>*, *<alias>*) **at least one of** *<relationship>* **for at least one of** *<relationship>* **at least one of** *<relationship>* **(**<alias>**)** **for at least one of** *<relationship>* **(**<alias>**)**	Used to refer from one ***entity*** to another ***entity*** in a "to-many" ***relationship***, when you need to determine whether any members of the target ***entity*** group need to satisfy the rule.
IsMemberOf(*<target>*, *<relationship>*) **IsMemberOf**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **is a member of** *<relationship>* <ent-target> **(**<alias>**) is a member of** *<relationship>*	Used to test that an ***entity*** instance is a target of a ***relationship*** for which a second ***entity*** instance is the source.
IsNotMemberOf(*<target>*, *<relationship>*) <ent-target> **is not a member of** *<relationship>*	Used to test that an ***entity*** instance is not a target of a ***relationship*** for which a second ***entity*** instance is the source.

Syntax	Description
InstanceCount(*<relationship>*) **the number of** *<relationship>*	Counts the number of instances that exist for an ***entity***.
InstanceCountIf(*<relationship>*, *<Exp>*) **the number of** *<relationship>* **for which it is the case that** *<condition>*	Counts the number of instances there are of an ***entity*** for which a particular ***entity-level attribute*** has a particular value.
InstanceMaximum(*<relationship>*, *<number-attr>*) **InstanceMaximum**(*<relationship>*, *<date-attr>*) **InstanceMaximum**(*<relationship>*, *<datetime-attr>*) **InstanceMaximum**(*<relationship>*, *<time-attr>*) *<date-attr>* **which is the latest for all** [**of**]*<relationship>* *<max-attr>* **which is the greatest for all** [**of**]*<relationship>* **the latest of all** *<relationship-attr>* **the latest of all** *<attr>* **for** *<relationship>* **the greatest of** [**all**]*<relationship-attr>* **the greatest of** [**all**]*<attr>* **for** [**all**] [**of**]*<relationship>*	Obtains the highest/most recent value of an ***entity-level*** variable for all instances of the ***entity***.
InstanceMaximumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **which is the latest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<max-attr>* **which is the greatest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the latest of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the greatest of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the greatest of** *<attr>* **for all** [**of**]*<relationship>* **for which it is the case that**	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.

Syntax	Description
<ent-test>	
InstanceMinimum(*<relationship>*, *<number-attr>*) **InstanceMinimum**(*<relationship>*, *<date-attr>*) **InstanceMinimum**(*<relationship>*, *<datetime-attr>*) **InstanceMinimum**(*<relationship>*, *<time-attr>*) *<date-attr>* **which is the earliest for all** [**of**]*<relationship>* *<attr>* **which is the least for all** [**of**]*<relationship>* **the earliest of all** *<relationship-attr>* **the earliest of all** *<attr>* **for** *<relationship>* **the least of** [**all**]*<relationship-attr>* **the least of** [**all**]*<attr>* **for** [**all**][**of**]*<relationship>*	Obtains the lowest/least recent value of an ***entity-level*** variable for all instances of the ***entity***.
InstanceMinimumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **which is the earliest for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<num-attr>* **which is the least for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the least of all** *<relationship-attr>* **for which it is the case that** *<ent-test>* **the least of all** *<attr>* **for** *<relationship>* **for which it is the case that** *<ent-test>* **the earliest of all** *<attr>* **for** *<relationship>* **for which it is the case that** *<ent-test>*	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.
InstanceSum(*<relationship>*, *<number-attr>*)	Obtains the sum of all instances of an ***entity-level*** variable.

Syntax	Description
<num-attr>(**totaled** | **totalled**) **for all** [**of**]*<relationship>* **the total amount of** [**all**]*<relationship-attr>* **the total for all** *<relationship-attr>* **total for all** *<relationship>***,** *<attr>*	
InstanceSumIf(*<relationship>*, *<number-attr>*, *<condition>*) *<num-attr>* **totalled for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* *<num-attr>* **totaled for all** [**of**]*<relationship>* **for which it is the case that** *<ent-test>* **the total amount of all** *<relationship-attr>* **only where** *<condition>* **the total amount of** [**all**]*<relationship-attr>* **for which it is the case that** *<condition>* **total for all** *<relationship>***,** *<attr>* **only where** *<condition>*	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.
InstanceValueIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<time-attr>*, *<condition>*)	Obtains 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 the value is ***uncertain***.
InstanceEquals(*<instance1>*, *<instance2>*) <ent-target> **is** <ent-target>	Determines if two instances of an ***entity*** are the same instance.
InstanceNotEquals(*<instance1>*, *<instance2>*) <ent-target> **is not** <ent-target>	Determines if two instances of an ***entity*** are not the same instance.
InferInstance(*<relationship>*, *<identity>*) *<rel>***(***<identity>***) exists**	Used as a conclusion to infer that an ***entity*** instance exists and is a member of a ***relationship***.

Temporal reasoning functions

Syntax	Description
IntervalCountDistinct (*<start-date>*, *<end-date>*, *<variable>*) **IntervalCountDistinct** (*<start-date>*, *<end-date>*, *<condition>*)	Counts the number of known distinct values for the variable, in the interval from the start ***date*** (inclusive) to the end ***date*** (exclusive).
IntervalCountDistinctIf (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	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.
IntervalDailySum(*<start-date>*, *<end-date>*, *<number-attr>*)	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.
IntervalDailySumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	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.
IntervalMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Selects the maximum value of a variable in the interval from a start ***date*** (inclusive) to an end ***date*** (exclusive).
IntervalMaximumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	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.
IntervalMinimum(*<start-*	Selects the minimum value of a variable in the interval from a start ***date*** (inclusive) to an end

Syntax	Description
date>, *<end-date>*, *<number-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	***date*** (exclusive).
IntervalMinimumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	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.
IntervalWeightedAverage (*<start-date>*, *<end-date>*, *<number-attribute>*)	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.
IntervalWeightedAverageIf (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	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).
IntervalAlways(*<start-date>*, *<end-date>*, *<condition>*)	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).
IntervalAtLeastDays(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	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).
IntervalConsecutiveDays (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	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).
IntervalSometimes(*<start-*	Returns true if and only if a boolean condition is ever true in the interval from the start ***date***

Syntax	Description
date>, *<end-date>*, *<condition>*)	(inclusive) to the end ***date*** (exclusive).
ValueAt(*<date>*, *<value>*)	Returns the value of the given ***attribute*** at the specified ***date***.
WhenLast(*<date>*, *<condition>*)	Returns the ***date*** on which a boolean condition was last true, looking backwards from (and including) a specified ***date***.
WhenNext(*<date>*, *<condition>*)	Returns the ***date*** on which a boolean condition will next be true, looking forwards from (and including) a specified ***date***.
Latest()	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.
Earliest()	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.
TemporalDaysSince(*<date>*, *<end-date>*)	Returns a number variable that varies every day and is the number of full days since the ***date***.
TemporalWeeksSince (*<date>*, *<end-date>*)	Returns a number variable that varies every week and is the number of full weeks since the ***date***.
TemporalMonthsSince (*<date>*, *<end-date>*)	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.
TemporalYearsSince (*<date>*, *<end-date>*)	Returns a number variable that varies every year and is the number of full years since the ***date***.
TemporalAlwaysDays (*<days>*, *<condition>*)	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.
TemporalConsecutiveDays (*<minDays>*, *<days>*, *<condition>*)	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.
TemporalSometimesDays (*<days>*, *<condition>*)	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.
TemporalAfter(*<date>*)	Returns a boolean ***attribute*** that varies over time and is true after a ***date*** and false on and before.
TemporalBefore(*<date>*)	Returns a boolean ***attribute*** that varies over time and is true before a ***date*** and false on and afterwards.

Syntax	Description
TemporalOn(*<date>*)	Returns a boolean ***attribute*** that varies over time and is true on a ***date*** and false before and afterwards.
TemporalOnOrAfter (*<date>*)	Returns a boolean ***attribute*** that varies over time and is true on or after a ***date*** and false before.
TemporalOnOrBefore (*<date>*)	Returns a boolean ***attribute*** that varies over time and is true on and before a ***date*** and false afterwards.
TemporalFromStartDate (*<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***.
TemporalFromEndDate (*<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***.
TemporalFromRange(*<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***.
TemporalIsWeekday (*<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.
TemporalOncePerMonth (*<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
Error (*<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.
Warning (*<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
CallCustomFunction(*<A>*, *<B>*)	CallCustomFunction

Conectores lógicos(English)

Sintaxe	**Descrição**
se	Termo opcional que pode ser apresentado no final de uma linha de conclusão com uma prova a acompanhar
e	Conjunção lógica entre dois atributos: ***attributes***
ou	Disjunção lógica entre dois atributos: ***attributes***
qualquer um **um de** **algum** **ao menos um destes é verdadeiro** **qualquer um destes está satisfeito**	Elemento de agrupamento utilizado com disjunções em que um ou mais atributos (***attributes***) necessitam de ser agrupados
ambos **tudo** **todos estes são verdadeiros** **todos estes estão satisfeitos**	Elemento de agrupamento utilizado com conjunções em que um ou mais atributos (***attributes***) necessitam de ser agrupados
caso contrário	Termo apresentado no final de uma regra de tabela para indicar a cláusula de caso contrário (otherwise)
é	Termo utilizado numa entrada de legenda entre a expressão abreviada e o valor de texto de atributo integral: ***attribute text***

Funções lógicas(English)

Sintaxe	**Descrição**
não é verdade que *<expr>*	Operador utilizado para devolver verdadeiro se o atributo (***attribute***) tiver um valor que seja falso
<var> **é certo** *<var>* **é certa** *<var>* **é certos** *<var>* **é certas** **é certo que** [**ou não**]*<expr>*	Operador utilizado para devolver verdadeiro se o atributo (***attribute***) tiver um valor que não seja ***uncertain***
<var> **é incerto** *<var>* **é incerta** *<var>* **é incertos**	Operador utilizado para devolver verdadeiro se o atributo (***attribute***) for ***uncertain***

Sintaxe	Descrição
<var> **é incertas** **é incerto se** [**ou não**]*<expr>* **é incerto que** [**ou não**]*<expr>* **não é certo que** *<expr>* **incerto**	
<var> **é conhecido** *<var>* **é conhecida** *<var>* **é conhecidos** *<var>* **é conhecidas** **sabe-se se** [**ou não**]*<expr>*	Operador utilizado para devolver verdadeiro se o atributo (***attribute***) tiver qualquer valor
<var> **é desconhecido** *<var>* **é desconhecida** *<var>* **é desconhecidos** *<var>* **é desconhecidas** **não se sabe se** [**ou não**]*<expr>* **desconhecido**	Operador utilizado para devolver verdadeiro se o atributo (***attribute***) não tiver um valor

Constantes lógicas(English)

Sintaxe	Descrição
verdadeiro	Valor de constante verdadeiro utilizado para as regras de tabela.
falso	Valor de constante falso utilizado para as regras de tabela.
incerto	Valor de constante ***uncertain*** utilizado para as regras de tabela.

Operadores de comparação(English)

Sintaxe	Descrição
*<x>***<***<y>*	Menor que Nota: não existe uma língua natural quando este operador é utilizado com valores numéricos e monetários.
<x> **>** *<y>*	Maior que Nota: não existe uma língua natural quando este operador é utilizado com valores numéricos e monetários.
*<x>***<=***<y>*	Menor que ou igual a
<x> **>=** *<y>*	Maior que ou igual a

Sintaxe	Descrição
<x>=*<y>*	Igual a
<x> **<>** *<y>*	Diferente de

Funções numéricas(English)

Sintaxe	Descrição
Número(*<numText>*)	Converte a cadeia de caracteres especificada num valor numérico
<x> **+** *<y>*	Adição
<x> **-** *<y>*	Subtração
<x> ***** *<y>*	Multiplicação
<x> **/** *<y>*	Divisão
<x> **** *<y>*	Divisão inteira
<x> **modulo** *<y>*	Resto da divisão inteira
Máximo(*<x>*, *<y>*) **Máximo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Devolve o maior de dois valores
Mínimo(*<x>*, *<y>*) **Mínimo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Devolve o menor de dois valores
Xy(*<x>*, *<y>*)	x à potência de y
Ex(*<x>*)	Constante e à potência de x
Abs(*<x>*) **|***<val>***|**	Valor absoluto de x
Ln(*<x>*)	Logaritmo natural de x
Log(*<x>*)	Logaritmo na base 10 de x
Raiz quadrada(*<x>*)	Raiz quadrada de x
Arredond(*<x>*, *<n>*)	Arredonda x para n casas decimais
Trunc(*<x>*, *<n>*)	x truncado para n casas decimais
Sen(*<x>*)	Seno de x
Cos(*<x>*)	Co-seno de x
Tan(*<x>*)	Tangente de x

Sintaxe	Descrição
Asen(<*x*>)	Arco seno de x
Acos(<*x*>)	Arco co-seno de x
Atan(<*x*>)	Arco tangente de x

Funções de data(English)

Sintaxe	Descrição
DataAtual()	Devolve a data atual (***date***) no início da sessão.
Data(<*text*>)	Converte a cadeia de caracteres especificada num valor de data: ***date***
DataDeCriação(<*year*>, <*month*>, <*day*>)	Devolve uma data (***date***) formada a partir do ano, mês e dia especificado.
ExtrairDia(<*date/datetime*>)	Devolve o componente de dia de uma data: ***date/datetime attribute***.
ExtrairMês(<*date/datetime*>)	Devolve o componente de mês de uma data: ***date/datetime attribute***.
ExtrairAno(<*date/datetime*>)	Devolve o componente de ano de uma data: ***date/datetime attribute***.
PróximoDiaDaSemana (<*date/datetime*>, <*day*>)	Devolve a data ***date*** do dia de semana seguinte que ocorre na ou antes/depois de uma data ***date*** (dependendo da sintaxe utilizada).
PróximaData(<*date*>, <*day*>, <*month*>)	Devolve a instância seguinte do dia e mês fornecidos após uma data: ***date***.
AdicionarDias (<*date/datetime*>, <*num_days*>)	Acrescenta/subtrai um número de dias a uma data ***date***. Quando utilizar a forma sintática concisa, o número deve ser um número inteiro positivo de forma a acrescentar dias à data da entrada de dados ***date*** ou um número negativo de forma a subtrair dias da data de entrada de dados ***date***.
AdicionarSemanas(<*date/datetime*>, <*num_weeks*>)	Acrescenta um número de semanas a uma data (***date***). Quando utilizar a forma sintática concisa, o número deve ser um número inteiro positivo de forma a acrescentar semanas à data da entrada de dados (***date***).
AdicionarMeses(<*date/datetime*>, <*num_months*>)	Acrescenta um número de meses a uma data (***date***). Quando utilizar a forma sintática concisa, o número deve ser um número inteiro positivo de forma a acrescentar meses à data da entrada de dados (***date***).
AdicionarAnos(<*date/datetime*>, <*num_years*>)	Acrescenta um número de anos a uma data (***date***). Quando utilizar a forma sintática concisa, o número deve ser um número inteiro positivo de forma a acrescentar anos à data da entrada de dados (***date***).
ContagemDosDiasDaSemana (<*date1*>, <*date2*>)	Conta o número de dias da semana entre a data (***date***)1 e a data (***date***)2. Ou seja, o número de dias entre segunda-feira e sexta-feira. Nota: A data (***date***) anterior é inclusiva e a data (***date***) posterior é exclusiva.

Sintaxe	Descrição
InícioDoAno(*<date/datetime>*)	Devolve a primeira data (***date***) do ano a que pertence a data (***date***).
FimDoAno(*<date/datetime>*)	Devolve a última data (***date***) do ano a que pertence a data (***date***).
DiferençaDeDia(*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de dias completos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeDiaInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de dias completos (inclusivamente) entre ***date/datetime1*** e ***date/datetime2***. Este cálculo exclui ambos os pontos terminais. Em que as datas sejam iguais, o resultado é 1. A ordem das duas datas não afeta o resultado.
DiferençaDeDiaExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de dias completos (exclusivamente) entre ***date/datetime1*** e ***date/datetime2***. Este cálculo exclui ambos os pontos terminais. Nos casos em que as datas sejam iguais, o resultado é 0. A ordem das duas datas não afeta o resultado.
DiferençaDeSemana(*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de semanas completas decorridas entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeSemanaInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de semanas completas decorridas inclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeSemanaExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de semanas completas decorridas exclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMês(*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de meses completos decorridos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMêsInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de meses completos decorridos inclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeMêsExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de meses completos decorridos exclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAno(*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de anos entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAnoInclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de anos inclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.
DiferençaDeAnoExclusiva (*<date/datetime1>*, *<date/datetime2>*)	Devolve o número de anos exclusivamente entre ***date/datetime1*** e ***date/datetime2***. A ordem das duas datas não afeta o resultado.

Funções de hora do dia(English)

Sintaxe	Descrição
HoraDoDia(*<text>*)	Converte a cadeia de caracteres fornecida numa hora do dia
ExtrairSegundo(*<time/datetime>*)	Devolve o componente de segundo de um horário: ***timeofday/datetime attribute***.
ExtrairMinuto(*<time/datetime>*)	Devolve o componente de minuto de um horário: ***timeofday/datetime attribute***.
ExtrairHora(*<time/datetime>*)	Devolve o componente de hora de um horário: ***timeofday/datetime attribute***.

Funções de data e hora(English)

Sintaxe	Descrição
DataHoraAtual()	Devolve a data (***date***) e hora atuais no início da sessão.
DataHora(*<text>*)	Converte a cadeia de caracteres especificada num valor de data: ***datetime***
ConcatenarDataHora(*<date>*, *<time>*)	Define a hora da data (***date***) através da junção da data (***date***) com a hora do dia.
DiferençaDeSegundo(*<datetime1>*, *<datetime2>*) **DiferençaDeSegundo**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número de segundos entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeSegundoInclusiva(*<datetime1>*, *<datetime2>*) **DiferençaDeSegundoInclusiva**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número inclusivamente de segundos entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeSegundoExclusiva(*<datetime1>*, *<datetime2>*) **DiferençaDeSegundoExclusiva**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número exclusivamente de segundos entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeMinuto(*<datetime1>*, *<datetime2>*) **DiferençaDeMinuto**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número de minutos entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeMinutoInclusiva(*<datetime1>*, *<datetime2>*) **DiferençaDeMinutoInclusiva**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número inclusivamente de minutos entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeMinutoExclusiva(*<datetime1>*, *<datetime2>*) **DiferençaDeMinutoExclusiva**(*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número exclusivamente de minutos entre os valores de data-hora: ***datetime1*** e ***datetime2***.

Sintaxe	Descrição
DiferençaDeHora(*<datetime1>*, *<datetime2>*) **DiferençaDeHora** (*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número de horas entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeHoraInclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeHoraInclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número inclusivamente de horas entre os valores de data-hora: ***datetime1*** e ***datetime2***.
DiferençaDeHoraExclusiva (*<datetime1>*, *<datetime2>*) **DiferençaDeHoraExclusiva** (*<timeOfDay1>*, *<timeOfDay2>*)	Devolve o número exclusivamente de horas entre os valores de data-hora: ***datetime1*** e ***datetime2***.
ExtrairData(*<datetime>*)	Extrai a data (***date***) de uma data-hora (***datetime attribute***).
ExtrairHoraDoDia(*<datetime>*)	Extrai a hora do dia a partir de uma data-hora (***datetime attribute***). Pode ser utilizado para definir o valor de uma hora do dia (***timeofday attribute***) com a hora em que a regra é executada através da extração da hora a partir da data (***date***) e hora.
AdicionarHoras(*<datetime>*, *<num_hours>*) **AdicionarHoras**(*<timeOfDay>*, *<num_hours>*)	Acrescenta um número de horas a uma hora: ***date***.
AdicionarMinutos(*<datetime>*, *<num_minutes>*) **AdicionarMinutos** (*<timeOfDay>*, *<num_minutes>*)	Acrescenta um número de minutos a uma hora: ***date***.
AdicionarSegundos (*<datetime>*, *<num_seconds>*) **AdicionarSegundos** (*<timeOfDay>*, *<num_seconds>*)	Acrescenta um número de segundos a uma hora: ***date***.

Funções de texto(English)

Sintaxe	Descrição
<text1> **&** *<text2>*	Combina o valor do texto ***text1*** com o valor do texto ***text2***, e assim sucessivamente, até formar um valor do texto ***text*** único. Nota: é possível utilizar variáveis de qualquer tipo. Os valores são formatados com o formatador instalado na sessão de execução de regras.
	Combina o valor do texto ***text1*** com o valor do texto ***text2***, e assim sucessivamente, até formar um valor do texto ***text*** único.

Sintaxe	Descrição
	Nota: é possível utilizar variáveis de qualquer tipo. Os valores são formatados com o formatador instalado na sessão de execução de regras.
Contém(*<text>*, *<substring>*)	Devolve um valor booleano a indicar se o valor ***text*** fornecido contém a subcadeia de caracteres ***text*** fornecida. A comparação ***text*** não é sensível a maiúsculas e minúsculas.
TerminaCom (*<text>*, *<substring>*)	Devolve um valor booleano a indicar se o valor ***text*** fornecido termina com a subcadeia de caracteres ***text*** fornecida. A comparação ***text*** não é sensível a maiúsculas e minúsculas.
ÉNúmero(*<text>*)	Devolve um valor booleano a indicar se o valor de texto (***text***) fornecido representa um número válido.
Comprimento (*<text>*)	Devolve o comprimento do carácter do valor de texto (***text***) fornecido.
ComeçaCom (*<text>*, *<substring>*)	Devolve um valor booleano a indicar se o valor ***text*** fornecido começa com a subcadeia de caracteres ***text*** fornecida. A comparação ***text*** não é sensível a maiúsculas e minúsculas.
Subsequência (*<text>*, *<offset>*, *<length>*)	Devolve a subcadeia de caracteres de texto (***text***) que inicia com o desvio fornecido, que corresponde ao comprimento especificado em caracteres. São devolvidos menos caracteres se o final da cadeia de caracteres for atingida.
Texto(*<number>*) **Texto**(*<date>*) **Texto**(*<datetime>*) **Texto** (*<timeOfDay>*)	Converte o número especificado ou atributo de data (***date attribute***) num valor de texto (***text***).

Funções de entidade e relação(English)

Sintaxe	Descrição
Para(*<relationship>*, *<Exp>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Uma", "Muitas para Uma" ou "Muitas para Muitas" ***relationship*** onde só existe uma condição.
ParaEscopo(*<relationship>*, *<alias>*) **ParaEscopo**(*<relationship>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Uma", "Muitas para Uma" ou "Muitas para Muitas" ***relationship*** onde existe uma ou mais condições.
ParaTudo(*<relationship>*, *<Exp>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Muitas" ou "Muitas para Muitas" ***relationship***, quando é necessário determinar se todos membros do grupo de entidades ***entity*** de destino têm de cumprir a regra. Esta forma é utilizada quando só existe uma condição na regra.

Sintaxe	Descrição
ParaTodosEscopos(*<relationship>*) **ParaTodosEscopos**(*<relationship>*, *<alias>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Muitas" ou "Muitas para Muitas" ***relationship***, quando é necessário determinar se todos os membros do grupo de entidades ***entity*** de destino têm de cumprir a regra. Esta forma é utilizada quando existe uma ou mais condições na regra.
Existe(*<relationship>*, *<Exp>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Muitas" ou "Muitas para Muitas" ***relationship***, quando é necessário determinar se quaisquer membros do grupo de entidades ***entity*** de destino têm de cumprir a regra. Esta forma é utilizada quando só existe uma condição na regra.
ExisteEscopo(*<relationship>*) **ExisteEscopo**(*<relationship>*, *<alias>*)	Utilizado para fazer referência de uma entidade ***entity*** para outra entidade ***entity*** numa relação "Uma para Muitas" ou "Muitas para Muitas" ***relationship***, quando é necessário determinar se quaisquer membros do grupo de entidades ***entity*** de destino têm de cumprir a regra. Esta forma é utilizada quando existe uma ou mais condições na regra.
ÉMembroDe(*<target>*, *<relationship>*) **ÉMembroDe**(*<target>*, *<alias>*, *<relationship>*)	Utilizado como conclusão para demonstrar que uma instância de entidade (***entity***) é membro de uma relação (***relationship***). Utilizado como uma condição para testar se uma instância de entidade (***entity***) é um destino de uma relação (***relationship***) para a qual uma segunda instância de entidade (***entity***) seja a origem.
NãoÉMembroDe(*<target>*, *<relationship>*)	Utilizado como condição para testar se uma instância de entidade (***entity***) não é um destino de uma relação (***relationship***) para a qual uma segunda instância de entidade (***entity***) seja a origem.
ContagemDeInstâncias (*<relationship>*)	Conta o número de instâncias que existem para uma entidade: ***entity***.
ContagemDeInstânciasSe (*<relationship>*, *<Exp>*)	Conta o número de instâncias existentes para uma entidade (***entity***) para a qual um atributo ao nível da entidade (***entity-level attribute***) em particular tem um valor específico.
MáximoDeInstâncias(*<relationship>*, *<number-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<date-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<datetime-attr>*) **MáximoDeInstâncias**(*<relationship>*, *<time-attr>*)	Obtém o valor mais elevado/mais recente de uma variável ao nível da entidade (***entity-level***) para todas as instâncias da entidade (***entity***).
MáximoDeInstânciasSe (*<relationship>*, *<number-attr>*, *<condition>*) **MáximoDeInstânciasSe** (*<relationship>*, *<date-attr>*,	Obtém o valor mais elevado/mais recente de uma variável ao nível da entidade (***entity-level***) para todas as instâncias da entidade (***entity***) para a qual um atributo ao nível da entidade (***entity-level attribute***) em particular tem um valor específico.

Sintaxe	Descrição
<condition>) **MáximoDeInstânciasSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MáximoDeInstânciasSe** (*<relationship>*, *<time-attr>*, *<condition>*)	
MínimoDeInstâncias(*<relationship>*, *<number-attr>*) **MínimoDeInstâncias**(*<relationship>*, *<date-attr>*) **MínimoDeInstâncias**(*<relationship>*, *<datetime-attr>*) **MínimoDeInstâncias**(*<relationship>*, *<time-attr>*)	Obtém o valor mais baixo/menos recente de uma variável ao nível da entidade (***entity-level***) para todas as instâncias da entidade (***entity***).
MínimoDeInstânciasSe (*<relationship>*, *<number-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<date-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MínimoDeInstânciasSe** (*<relationship>*, *<time-attr>*, *<condition>*)	Obtém o valor mais baixo/menos recente de uma variável ao nível da entidade (***entity-level***) para todas as instâncias da entidade (***entity***) para a qual um atributo ao nível da entidade (***entity-level attribute***) em particular tem um valor específico.
SomaDeInstâncias(*<relationship>*, *<number-attr>*)	Obtém a soma de todas as instâncias de uma variável ao nível da entidade: ***entity-level***.
SomaDeInstânciasSe(*<relationship>*, *<number-attr>*, *<condition>*)	Obtém a soma de todas as instâncias de uma variável ao nível da entidade (***entity-level***) para a qual se verifica ser verdadeiro uma entidade (***entity***) com atributo (***attribute***) Booleano específico ao nível da entidade (***entity-level***).
ValorIfInstância(*<relationship>*, *<number-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<text-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<date-attr>*, *<condition>*)	Obtém um valor de uma instância de entidade ***entity*** exclusiva, identificada a partir das instâncias de entidade ***entity*** de destino de uma relação ***relationship*** por uma condição. • Se a condição identificar uma única instância de entidade ***entity*** de destino, o valor corresponde ao valor calculado relativamente a essa instância de entidade ***entity***. • Se mais do que uma instância de destino cumprir a condição, é devolvido o estado ***uncertain***. • Se nenhuma instância de destino cumprir a condição e a relação ***relationship*** for conhecida, o valor será ***uncertain***.

Sintaxe	Descrição
ValorIfInstância(*<relationship>*, *<datetime-attr>*, *<condition>*) **ValorIfInstância**(*<relationship>*, *<time-attr>*, *<condition>*)	
IgualdadesDeInstância (*<instance1>*, *<instance2>*)	Determina se duas instâncias de uma entidade (***entity***) são a mesma instância.
SemIgualdadesDeInstância (*<instance1>*, *<instance2>*)	Determina se duas instâncias de uma entidade (***entity***) não são a mesma instância.
InstânciaDemonstrada (*<relationship>*, *<identity>*) *<rel>***(***<identity>***) existe**	Utilizado como uma conclusão para demonstrar que uma instância de entidade ***entity*** existe e é um membro de uma relação ***relationship***.

Funções de raciocínio temporal(English)

Sintaxe	Descrição
ContagemDeIntervalosDistintos (*<start-date>*, *<end-date>*, *<variable>*) **ContagemDeIntervalosDistintos** (*<start-date>*, *<end-date>*, *<condition>*)	Conta o número de valores distintos conhecidos para a variável, no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente).
ContagemDeIntervalosDistintosSe (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Conta o número de valores distintos conhecidos para a variável, no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente), incluindo apenas as ocorrências em que um filtro booleano é verdadeiro.
SomaDiáriaDeIntervalos(*<start-date>*, *<end-date>*, *<number-attr>*)	Calcula a soma de uma variável monetária ou numérica, no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente). O atributo (***attribute***) é assumido como uma quantidade diária.
SomaDiáriaDeIntervalosSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Calcula a soma de todos os valores diários para uma variável monetária ou numérica, no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente), incluindo apenas as ocorrências em que uma condição é verdadeira.
MáximoDeIntervalos(*<start-date>*, *<end-date>*, *<number-attr>*) **MáximoDeIntervalos**(*<start-date>*, *<end-date>*, *<date-attr>*) **MáximoDeIntervalos**(*<start-date>*,	Seleciona o valor máximo de uma variável no intervalo a partir uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente).

Sintaxe	Descrição
<end-date>, *<datetime-attr>*) **MáximoDeIntervalos**(*<start-date>*, *<end-date>*, *<time-attr>*)	
MáximoDeIntervalosSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MáximoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MáximoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MáximoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Seleciona o valor máximo de uma variável no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente), incluindo apenas as ocorrências em que uma condição é verdadeira.
MínimoDeIntervalos(*<start-date>*, *<end-date>*, *<number-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<date-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MínimoDeIntervalos**(*<start-date>*, *<end-date>*, *<time-attr>*)	Seleciona o valor mínimo de uma variável no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente).
MínimoDeIntervalosSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MínimoDeIntervalosSe**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Seleciona o valor mínimo de uma variável no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente), incluindo apenas as ocorrências em que uma condição é verdadeira.
MédiaPonderadaDeIntervalos (*<start-date>*, *<end-date>*, *<number-attribute>*)	Calcula o valor médio de uma variável monetária ou numérica no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente) ponderada pelo espaço de tempo a que se aplica cada valor.
MédiaPonderadaDeIntervalosSe (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Calcula o valor médio de uma variável monetária ou numérica no intervalo a partir de uma data (***date***) de início (inclusivamente) até uma data (***date***) de fim (exclusivamente), incluindo apenas as ocorrências em que uma condição booleana é ver-

Sintaxe	Descrição
	dadeira (ponderada pelo espaço de tempo a que se aplica cada valor e em que o filtro seja verdadeiro).
IntervaloSempre(*<start-date>*, *<end-date>*, *<condition>*)	Devolve verdadeiro se, e apenas se, uma condição booleana for sempre verdadeira no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente).
IntervalNoMínimo(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Devolve verdadeiro se, e apenas se, uma condição booleana for verdadeira pelo menos para o número de dias especificado (não necessariamente consecutivos) no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente).
IntervaloDiasConsecutivos(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Devolve verdadeiro se, e apenas se, uma condição booleana for verdadeira pelo menos para determinado número de dias consecutivos fornecidos no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente).
IntervaloAlgumasVezes(*<start-date>*, *<end-date>*, *<condition>*)	Devolve verdadeiro se, e apenas se, uma condição booleana for alguma vez verdadeira no intervalo a partir da data (***date***) de início (inclusivamente) até à data (***date***) de fim (exclusivamente).
ValorEm(*<date>*, *<value>*)	Devolve o valor do atributo (***attribute***) fornecido na data (***date***) especificada.
QuandoÚltimo(*<date>*, *<condition>*)	Devolve a data (***date***) na qual uma condição booleana foi verdadeira pela última vez, ao observar retroativamente a partir de (e incluindo) uma data (***date***) especificada.
QuandoPróximo(*<date>*, *<condition>*)	Devolve a data (***date***) na qual uma condição booleana será verdadeira de seguida, ao observar retroativamente a partir de (e incluindo) uma data (***date***) especificada.
Último()	Devolve um valor de data (***date***) equivalente ao valor de data (***date***) mais recente possível, nomeadamente uma data (***date***) garantidamente posterior a qualquer outra data (***date***) que um atributo de data (***date attribute***) possa obter ou que uma expressão possa equivaler.
Primeiro()	Devolve um valor de data (***date***) equivalente ao valor de data (***date***) mais antigo possível, nomeadamente uma data (***date***) garantidamente anterior a qualquer outra data (***date***) que um atributo de data (***date attribute***) possa obter ou que uma expressão possa equivaler.
TemporalDiasDesde(*<date>*, *<end-date>*)	Devolve uma variável numérica que varia todos os dias e corresponde ao número de dias completos a partir da data: ***date***.
TemporalSemanasDesde(*<date>*, *<end-date>*)	Devolve uma variável numérica que varia todas as semanas e corresponde ao número de semanas completas a partir da data: ***date***.
TemporalMesesDesde(*<date>*, *<end-date>*)	Devolve uma variável numérica que varia todos os meses e corresponde ao número de meses completos a partir da data: ***date***. Nota: Em que a data (***date***) fornecida seja posterior ao dia 28 do mês e um mês subsequente tenha menos dias do que o

Sintaxe	Descrição
	mês fornecido, o ponto de mudança para o mês de aniversário será criado no último dia desse mês. Por exemplo, se a data (***date***) fornecida for 28, 29, 30 ou 31 de Janeiro de 2007, o primeiro ponto de mudança será 28 de Fevereiro de 2007.
TemporalAnosDesde(*<date>*, *<end-date>*)	Devolve uma variável numérica que varia todos os anos e corresponde ao número de anos completos a partir da data: ***date***.
TemporalSempreDias(*<days>*, *<condition>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro se, e apenas se, uma condição booleana for verdadeira para todos dentro de determinado número de dias precedentes, não incluindo o dia atual.
TemporalDiasConsecutivos (*<minDays>*, *<days>*, *<condition>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro se, e apenas se, uma condição booleana for verdadeira para pelo menos um número mínimo de dias consecutivos em qualquer altura no âmbito do número definido de dias precedentes, não incluindo o dia atual.
TemporalAlgunsDias(*<days>*, *<condition>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro se, e apenas se, uma condição booleana for alguma vez verdadeira no âmbito de um número especificado de dias precedentes, não incluindo o dia atual.
TemporalApós(*<date>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro após uma data (***date***) e falso na data e antes da mesma.
TemporalAntes(*<date>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro antes de uma data (***date***) e falso na data ou após a mesma.
TemporalEm(*<date>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro numa data (***date***) e falso antes da data ou após a mesma.
TemporalEmOuApós(*<date>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro numa data (***date***) ou após a mesma e falso antes da data.
TemporalEmOuAntes(*<date>*)	Devolve um atributo (***attribute***) booleano que varia ao longo do tempo e que é verdadeiro numa data e antes de uma data (***date***) e falso após a mesma.
TemporalDaDataDeInício(*<relationship>*, *<date>*, *<value>*)	Devolve um único atributo (***attribute***) temporal (ao nível da entidade (***entity***) de origem) a partir de uma relação (***relationship***) e de um atributo (***attribute***) de valor nas entidades, com valores que entram em vigor a partir de um atributo de data (***date attribute***) de início.
TemporalDaDataDeTérmino(*<relationship>*, *<date>*, *<value>*)	Devolve um único atributo (***attribute***) temporal (ao nível da entidade (***entity***) de origem) a partir de uma relação (***relationship***) e de um atributo (***attribute***) de valor nas entidades, com valores que entram em vigor até um atributo de data (***date attribute***) de fim.
TemporalDoIntervalo(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Devolve um único atributo (***attribute***) temporal (ao nível da entidade (***entity***) de origem) a partir de uma relação (***relationship***) e de um atributo (***attribute***) de valor nas entidades, com valores que entram em vigor a partir de uma data (***date attribute***) de início (inclusivamente) até um atributo de data (***date attribute***) de

Sintaxe	Descrição
	fim (exclusivamente). O valor é ***uncertain*** se expirar antes da data (***date***) de início seguinte.
TemporalDiaDaSemana (<*startdate*>, <*enddate*>)	Devolve verdadeiro nas datas que são dias da semana e falso nas datas que são fins-de-semana a partir da data (***date***) de início (inclusivamente) especificada até à data (***date***) de fim (exclusivamente). Devolve ***uncertain*** fora do intervalo de data: ***date***.
TemporalUmaVezPorMês (<*startdate*>, <*enddate*>, <*day-ofmonth*>)	Devolve verdadeiro se o dia for igual ao parâmetro do dia do mês e falso em todos os outros dias do mês a partir da data (***date***) de início (inclusivamente) especificada até à data (***date***) de fim (exclusivamente). Devolve ***uncertain*** fora do intervalo de data (***date***). Quando o dia do mês excede o número de dias do mês atual, o valor é verdadeiro no último dia desse mês, de forma que a função devolve um valor que é verdadeiro exatamente um dia por mês.

Funções do evento de validação(English)

Sintaxe	Descrição
Erro (<*text*>)	Um evento de erro é utilizado para transmitir uma mensagem ao utilizador e evitar que continue uma investigação até a condição que desencadeou esse erro já não se aplicar.
Aviso (<*text*>)	Um evento de aviso é utilizado para transmitir uma mensagem ao utilizador, mas permite que continue apesar da condição que desencadeou esse aviso.

Funções recusadas(English)

Sintaxe	Descrição
ChamarFunçãoPersonalizada (<*A*>, <*B*>)	Devolve o resultado de uma chamada externa para uma biblioteca de códigos. A biblioteca de códigos deve ser fornecida para o Determinations Engine para que a chamada de função personalizada tenha êxito.

Loogiset operaattorit(English)

Syntaksi	Kuvaus
jos	Valinnainen termi, joka voi esiintyä sellaisen päätelmärivin lopussa, jota seuraa todennus
ja	Looginen yhteys kahden määritteen ***attributes*** välillä
tai	Looginen disjunktio kahden määritteen ***attributes*** välillä
yksi **jompikumpi**	Ryhmityselementit, joita käytetään disjunktioissa, joissa vähintään kaksi määritettä ***attributes*** täytyy ryhmitellä

Syntaksi	Kuvaus
jokin seuraavista **mikä tahansa** **ainakin yksi seuraavista on tosi** **mikä tahansa seuraavista on täytetty**	
molemmat **kaikki** **kaikki seuraavat ovat tosia** **kaikki seuraavat on täytetty**	Ryhmityselementit, joita käytetään yhteyksissä, joissa vähintään kaksi määritettä ***attributes*** täytyy ryhmitellä
muussa tapauksessa	Termi, joka esiintyy taulusäännön lopussa ja ilmaisee muussa tapauksessa -lausekkeen
on	Termi, jota käytetään selitesyötössä lyhennetyn fraasin ja täyden kohteen ***attribute text*** välillä

Loogiset funktiot(English)

Syntaksi	Kuvaus
ei ole tosi, että *<expr>*	Operaattori, jota käytetään palauttamaan arvo tosi, jos määritteen ***attribute*** arvo on epätosi
<var> **on varma** **on varmaa, onko** *<expr>* **on varma, että** *<expr>*	Operaattori, jota käytetään palauttamaan arvo tosi, jos määritteen ***attribute*** arvo on eri kuin ***uncertain***
<var> **on epävarma** *<var>* **ei ole varma** **on epävarmaa, että** *<expr>* **on epävarmaa, onko** *<expr>* **ei ole varmaa, että** *<expr>* **epävarma**	Operaattori, jota käytetään palauttamaan arvo tosi, jos määritteen ***attribute*** arvo on ***uncertain***
<var> **on tunnettu** **tiedetään, onko** *<expr>* **on tiedossa, että** *<expr>*	Operaattori, jota käytetään palauttamaan arvo tosi, jos määritteellä ***attribute*** on mitään arvoa
<var> **on tuntematon** **ei tiedetä, onko** *<expr>* **ei ole tiedossa, onko** *<expr>* **tuntematon**	Operaattori, jota käytetään palauttamaan arvo tosi, jos määritteellä ***attribute*** ei ole arvoa

Loogiset vakiot(English)

Syntaksi	Kuvaus
tosi	Tosi-vakioarvo, jota käytetään taulusäännöissä.
epätosi	Epätosi-vakioarvo, jota käytetään taulusäännöissä.
epävarma	Vakioarvo ***uncertain***, jota käytetään taulusäännöissä.

Vertailuoperaattorit(English)

Syntaksi	Kuvaus
*<x>***<***<y>* *<x>* **on vähemmän kuin** *<y>*	Pienempi kuin Huomaa, ettei luonnollista kielimuotoa ole, kun tätä operaattoria käytetään numeeristen arvojen ja valuutta-arvojen yhteydessä.
<x> **>** *<y>* *<x>* **on suurempi kuin** *<y>* *<x>* **on myöhäisempi kuin** *<y>*	Suurempi kuin Huomaa, ettei luonnollista kielimuotoa ole, kun tätä operaattoria käytetään numeeristen arvojen ja valuutta-arvojen yhteydessä.
*<x>***<=***<y>* *<x>* **on vähemmän tai yhtä suuri kuin** *<y>*	Pienempi tai yhtä suuri kuin
<x> **>=** *<y>* *<x>* **on suurempi tai yhtä suuri kuin** *<y>*	Suurempi tai yhtä suuri kuin
*<x>***=***<y>* *<x>* **on yhtäsuuri kuin** *<y>* *<x>* **on yhtä kuin** *<y>*	Yhtä suuri kuin
<x> **!=** *<y>* *<x>* **<>** *<y>* *<x>* **on erisuuri kuin** *<y>* *<x>* **ei ole yhtä kuin** *<y>* *<x>* **ei ole sama kuin** *<y>*	Eri suuri

Numeeriset funktiot(English)

Syntaksi	Kuvaus
Määrä(*<numText>*)	Muunna määritetty merkkijono numeroarvoksi
<x> **+** *<y>*	Yhteenlasku
<x> **-** *<y>*	Vähennyslasku

Syntaksi	Kuvaus
<x> * *<y>*	Kertolasku
<x> **/** *<y>*	Jakolasku
<x> **** *<y>*	Kokonaisluvun jakolasku
<x> **modulo** *<y>*	Kokonaisluvun jakojäännös
Enimmäisarvo(*<x>*, *<y>*) **Enimmäisarvo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Palauttaa kahdesta arvosta suuremman
Vähimmäisarvo(*<x>*, *<y>*) **Vähimmäisarvo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Palauttaa kahdesta arvosta pienemmän
Xy(*<x>*, *<y>*) *<x>* **korotettuna** *<y>* **. potenssiin** *<x>* **potenssiin** *<y>*	X potenssiin y
Eksponentti(*<x>*) **e potenssiin** *<x>*	Vakio e potenssiin x
Itseisarvo(*<x>*) **|***<val>***|** *<x>* **itseisarvo**	X:n absoluuttinen arvo
LuonnollinenLogaritmi(*<x>*) **luonnollinen logaritmi** *<x>***:sta** **luonnollinen logaritmi** *<x>***:stä**	X:n luonnollinen logaritmi
Logaritmi(*<x>*) **10-kanta logaritmi** *<x>***:sta** **10-kanta logaritmi** *<x>***:stä**	X:n 10-kantainen logaritmi
Neliöjuuri(*<x>*) **neliöjuuri** *<x>***:sta** **neliöjuuri** *<x>***:stä**	X:n neliöjuuri
Pyöristys(*<x>*, *<n>*) *<x>* **pyöristettynä** *<n>* **desimaaliin**	Pyöristää x:n n:n desimaalin tarkkuudelle
Katkaisu(*<x>*, *<n>*) *<x>* **katkaistuna** *<n>* **desimaaliin**	X pyöristettynä n:n desimaalin tarkkuudelle
Sini(*<x>*)	X:n sini
Kosini(*<x>*)	X:n kosini
Tangentti(*<x>*)	X:n tangentti
Arkussini(*<x>*)	X:n arkussini

Syntaksi	Kuvaus
Arkuskosini(*<x>*)	X:n arkussini
Arkustangentti(*<x>*)	X:n arkustangentti

Päivämäärän funktiot(English)

Syntaksi	Kuvaus
NykyinenPvm()	Palauttaa kuluvan pvm:n ***date*** istunnon alussa.
Pvm(*<text>*)	Muuntaa määritetyn merkkijonon arvoksi ***date***
TeonPvm(*<year>*, *<month>*, *<day>*)	Palauttaa päivämäärän ***date***, joka muodostetaan määritetystä vuodesta, kuukaudesta ja päivästä.
PoimiPäivä(*<date/datetime>*)	Palauttaa päivän arvosta ***date/datetime attribute***.
PoimiKuukausi(*<date/datetime>*)	Palauttaa kuukauden arvosta ***date/datetime attribute***.
PoimiVuosi(*<date/datetime>*)	Palauttaa vuoden arvosta ***date/datetime attribute***.
SeuraavaViikonpäivä(*<date/datetime>*, *<day>*)	Palauttaa seuraavan viikonpäivän päivämäärän ***date*** päivämääränä ***date***, sitä ennen tai sen jälkeen (käytetyn syntaksin mukaan).
SeuraavaPvm(*<date>*, *<day>*, *<month>*)	Palauttaa annetun päivän ja kuukauden seuraavan instanssin päivämäärän ***date*** jälkeen.
LisääPäivät(*<date/datetime>*, *<num_days>*)	Lisää päiviä päivämäärään ***date*** tai poistaa siitä päiviä. Käytettäessä suppeaa syntaktista muotoa numeron täytyy olla positiivinen kokonaisluku, jotta syötteeseen ***date*** voidaan lisätä päiviä, tai negatiivinen luku, jotta syötteestä ***date*** voidaan vähentää päiviä.
LisääViikot(*<date/datetime>*, *<num_weeks>*)	Lisää viikkoja päivämäärään ***date***. Käytettäessä suppeaa syntaktista muotoa numeron täytyy olla positiivinen kokonaisluku, jotta syötteeseen ***date*** voidaan lisätä viikkoja.
LisääKuukaudet(*<date/datetime>*, *<num_months>*)	Lisää kuukausia päivämäärään ***date***. Käytettäessä suppeaa syntaktista muotoa numeron täytyy olla positiivinen kokonaisluku, jotta syötteeseen ***date*** voidaan lisätä kuukausia.
LisääVuodet(*<date/datetime>*, *<num_years>*)	Lisää vuosia päivämäärään ***date***. Käytettäessä suppeaa syntaktista muotoa numeron täytyy olla positiivinen kokonaisluku, jotta syötteeseen ***date*** voidaan lisätä vuosia.
ViikonpäivienMäärä(*<date1>*, *<date2>*)	Laskee arkipäivien määrän päivien ***date***1 ja ***date***2 välillä. Mukaan lasketaan vain päivät maanantaista perjantaihin. Huom: aikaisempi ***date*** sisältyy laskelmaan, myöhäisempi ***date*** ei sisälly.
VuodenAlku(*<date/datetime>*)	Palauttaa ensimmäisen päivämäärän ***date*** vuotena, jolle ***date*** osuu.
VuodenLoppu(*<date/datetime>*)	Palauttaa viimeisen päivämäärän ***date*** vuotena, jolle ***date*** osuu.

Syntaksi	Kuvaus
PäivienErotus(*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten päivien lukumäärän arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta tulokseen.
PäivienErotusSisällyttävä (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten päivien lukumäärän (sisällyttävä) arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Laskennassa otetaan mukaan alku- ja loppupäivä. Jos päivämäärät ovat samat, tulos on 1. Päivämäärien järjestys ei vaikuta tulokseen.
PäivienErotusPoissulkeva (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten päivien lukumäärän (poissulkeva) arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Laskennassa jätetään pois alku- ja loppupäivä. Jos päivämäärät ovat samat, tulos on 0. Päivämäärien järjestys ei vaikuta tulokseen.
ViikkojenErotus(*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden viikkojen määrän päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.
ViikkojenErotusSisällyttävä (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden viikkojen (sisällyttävä) määrän päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.
ViikkojenErotusPoissulkeva (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden viikkojen (poissulkeva) määrän päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.
KuukausienErotus(*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden kuukausien määrän arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta tulokseen.
KuukausienErotusSisällyttävä (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden kuukausien määrän (sisällyttävä) arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta tulokseen.
KuukausienErotusPoissulkeva (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa kokonaisten kuluneiden kuukausien määrän (poissulkeva) arvojen ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta tulokseen.
VuosienErotus(*<date/datetime1>*, *<date/datetime2>*)	Palauttaa vuosien määrän päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.
VuosienErotusSisällyttävä (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa vuosien määrän (poissulkeva) päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.
VuosienErotusPoissulkeva (*<date/datetime1>*, *<date/datetime2>*)	Palauttaa vuosien määrän (poissulkeva) päivämäärien ***date/datetime1*** ja ***date/datetime2*** välillä. Päivämäärien järjestys ei vaikuta lopputulokseen.

Kellonajan funktiot(English)

Syntaksi	Kuvaus
Kellonaika(*<text>*)	Muuntaa annetun merkkijonon kellonajaksi
PoimiSekunti(*<time/datetime>*)	Palauttaa sekunnin arvosta ***timeofday/datetime attribute***.
PoimiMinuutti(*<time/datetime>*)	Palauttaa minuutin arvosta ***timeofday/datetime attribute***.
PoimiTunti(*<time/datetime>*)	Palauttaa tunnin arvosta ***timeofday/datetime attribute***.

Päivämäärän ja kellonajan funktiot(English)

Syntaksi	Kuvaus
NykyinenPvmAika()	Palauttaa kuluvan pvm:n ***date*** ja kellonajan istunnon alussa.
PvmAika(*<text>*)	Muuntaa määritetyn merkkijonon arvoksi ***datetime***
LyhennettyPvmAika(*<date>*, *<time>*)	Asettaa ajan ***date*** liittämällä ajan ***date*** ja kellonajan.
SekuntienErotus(*<datetime1>*, *<datetime2>*) **SekuntienErotus** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa sekuntien määrän arvojen ***datetime1*** ja ***datetime2*** välillä.
SekuntienErotusSisällyttävä (*<datetime1>*, *<datetime2>*) **SekuntienErotusSisällyttävä** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa sekuntien määrän (sisällyttävä) arvojen ***datetime1*** ja ***datetime2*** välillä.
SekuntienErotusPoissulkeva (*<datetime1>*, *<datetime2>*) **SekuntienErotusPoissulkeva** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa sekuntien määrän (poissulkeva) arvojen ***datetime1*** ja ***datetime2*** välillä.
MinuuttienErotus(*<datetime1>*, *<datetime2>*) **MinuuttienErotus** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa minuuttien määrän arvojen ***datetime1*** ja ***datetime2*** välillä.
MinuuttienErotusSisällyttävä (*<datetime1>*, *<datetime2>*) **MinuuttienErotusSisällyttävä** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa minuuttien määrän (sisällyttävä) arvojen ***datetime1*** ja ***datetime2*** välillä.
MinuuttienErotusPoissulkeva (*<datetime1>*, *<datetime2>*) **MinuuttienErotusPoissulkeva** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa minuuttien määrän (poissulkeva) arvojen ***datetime1*** ja ***datetime2*** välillä.

Syntaksi	Kuvaus
TuntienErotus(*<datetime1>*, *<datetime2>*) **TuntienErotus**(*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa tuntien määrän arvojen ***datetime1*** ja ***datetime2*** välillä.
TuntienErotusSisällyttävä (*<datetime1>*, *<datetime2>*) **TuntienErotusSisällyttävä** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa tuntien määrän (sisällyttävä) arvojen ***datetime1*** ja ***datetime2*** välillä.
TuntienErotusPoissulkeva (*<datetime1>*, *<datetime2>*) **TuntienErotusPoissulkeva** (*<timeOfDay1>*, *<timeOfDay2>*)	Palauttaa tuntien määrän (poissulkeva) arvojen ***datetime1*** ja ***datetime2*** välillä.
PoimiPvm(*<datetime>*)	Erottaa päivämäärän ***date*** arvosta ***datetime attribute***.
PoimiKellonaika(*<datetime>*)	Erottaa kellonajan arvosta ***datetime attribute***. Voidaan käyttää kellonajan ***timeofday attribute*** arvon asettamiseen ajaksi, jolloin sääntö on suoritettu erottamalla aika tämän hetkisestä päivämäärästä ***date*** ja ajasta.
LisääTunnit(*<datetime>*, *<num_hours>*) **LisääTunnit**(*<timeOfDay>*, *<num_hours>*)	Lisää tunteja kohteen ***date*** aikaan.
LisääMinuutit(*<datetime>*, *<num_minutes>*) **LisääMinuutit**(*<timeOfDay>*, *<num_minutes>*)	Lisää minuutteja kohteen ***date*** aikaan.
LisääSekunnit(*<datetime>*, *<num_seconds>*) **LisääSekunnit**(*<timeOfDay>*, *<num_seconds>*)	Lisää sekunteja kohteen ***date*** aikaan.

Tekstin funktiot(English)

Syntaksi	Kuvaus
<text1> **&** *<text2>*	Yhdistää arvot ***text1*** ja ***text2*** jne. yhdeksi arvoksi ***text***. Huomaa, että voit käyttää kaikentyyppisiä muuttujia. Arvot muotoillaan säännön istunnossa asennetussa muotoiluohjelmassa.
yhdistäminen *<text1>* **&** *<text2>*	Yhdistää arvot ***text1*** ja ***text2*** jne. yhdeksi arvoksi ***text***. Huomaa, että voit käyttää kaikentyyppisiä muuttujia. Arvot muotoillaan säännön istunnossa asennetussa muotoiluohjelmassa.

Syntaksi	Kuvaus
Sisältää(*<text>*, *<substring>*) *<text>* **sisältää** *<substring>*	Palauttaa totuusarvon, joka kertoo, sisältääkö annettu merkkijono ***text*** annetun alimerkkijonon ***text***. Vertailussa ***text*** merkkikoko ei ole merkitsevä.
LoppuuKohteeseen (*<text>*, *<substring>*) *<text>* **päättyy** *<substring>*	Palauttaa totuusarvon, joka kertoo, päättyykö annettu merkkijono ***text*** annettuun alimerkkijonoon ***text***. Vertailussa ***text*** merkkikoko ei ole merkitsevä.
OnNumero(*<text>*) *<text>* **on luku**	Palauttaa totuusarvon, joka kertoo, edustaako annettu syöte ***text*** sallittua numeroa.
Pituus(*<text>*)	Palauttaa annetun merkkijonon ***text*** pituuden merkkeinä.
AlkaaKohteella(*<text>*, *<substring>*) *<text>* **alkaa** *<substring>*	Palauttaa totuusarvon, joka kertoo, alkaako annettu merkkijono ***text*** annetulla alimerkkijonolla ***text***. Vertailussa ***text*** merkkikoko ei ole merkitsevä.
Alimerkkijono(*<text>*, *<offset>*, *<length>*)	Palauttaa merkkijonon ***text*** määrätyn pituisen alimerkkijonon, joka alkaa annetusta siirtymästä. Merkkejä palautetaan vähemmän, jos merkkijonon loppu saavutetaan.
Teksti(*<number>*) **Teksti**(*<date>*) **Teksti**(*<datetime>*) **Teksti**(*<timeOfDay>*)	Muunna määritetty numero tai ***date attribute text***-arvoksi.

Yksikön ja suhteen funktiot(English)

Syntaksi	Kuvaus
Kohteelle(*<relationship>*, *<Exp>*)	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi yhteen-, monta yhteen- tai monta moneen -suhteessa ***relationship***, kun ehtoja on vain yksi.
KohteelleLaajuus(*<relationship>*, *<alias>*) **KohteelleLaajuus**(*<relationship>*)	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi yhteen-, monta yhteen- tai monta moneen -suhteessa ***relationship***, kun ehtoja on vähintään yksi.
Kaikille(*<relationship>*, *<Exp>*) **jokainen** *<relationship-attr>*	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi moneen- tai monta moneen -suhteessa ***relationship***, kun täytyy määrittää, tarvitseeko kohdeyksikön ***entity*** ryhmän kaikkien jäsenten täyttää sääntöä. Tätä muotoa käytetään, kun säännössä on vain yksi ehto.
KaikilleLaajuus(*<relationship>*) **KaikilleLaajuus**(*<relationship>*, *<alias>*)	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi moneen- tai monta moneen -suhteessa ***relationship***, kun täytyy määrittää, tarvitseeko kohdeyksikön ***entity*** ryhmän kaikkien jäsenten täyttää sääntöä.

Syntaksi	**Kuvaus**
	Tätä muotoa käytetään, kun säännössä on vähintään yksi ehto.
On(*<relationship>*, *<Exp>*) **vähintään yksi** *<relationship-attr>*	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi moneen- tai monta moneen -suhteessa ***relationship***, kun täytyy määrittää, tarvitseeko kohdeyksikön ***entity*** ryhmän minkään jäsenten täyttää sääntöä. Tätä muotoa käytetään, kun säännössä on vain yksi ehto.
OnLaajuus(*<relationship>*) **OnLaajuus**(*<relationship>*, *<alias>*)	Käytetään viittaamaan yksiköstä ***entity*** yksikköön ***entity*** yksi moneen- tai monta moneen -suhteessa ***relationship***, kun täytyy määrittää, tarvitseeko kohdeyksikön ***entity*** ryhmän minkään jäsenten täyttää sääntöä. Tätä muotoa käytetään, kun säännössä on vähintään yksi ehto.
OnJäsen(*<target>*, *<relationship>*) **OnJäsen**(*<target>*, *<alias>*, *<relationship>*)	Käytetään päätelmänä sen johtamiseen, että yksikön ***entity*** instanssi on suhteen ***relationship*** jäsen. Käytetään ehtona testaamaan, että yksikön ***entity*** instanssi on sellaisen suhteen ***relationship*** kohde, jonka toinen yksikön ***entity*** instanssi on lähde.
EiOleJäsen(*<target>*, *<relationship>*)	Käytetään ehtona testaamaan, että yksikön ***entity*** instanssi ei ole sellaisen suhteen ***relationship*** kohde, jonka toinen yksikön ***entity*** instanssi on lähde.
InstanssienMäärä(*<relationship>*)	Laskee yksikölle ***entity*** olemassa olevien instanssien määrän.
InstanssienMääräJos(*<relationship>*, *<Exp>*)	Laskee olemassa olevien instanssien määrän yksikölle ***entity***, jossa tietyllä määritteellä ***entity-level attribute*** on tietty arvo.
InstanssienEnimmäismäärä (*<relationship>*, *<number-attr>*) **InstanssienEnimmäismäärä** (*<relationship>*, *<date-attr>*) **InstanssienEnimmäismäärä** (*<relationship>*, *<datetime-attr>*) **InstanssienEnimmäismäärä** (*<relationship>*, *<time-attr>*)	Hakee muuttujan ***entity-level*** korkeimman/viimeisimmän arvon kaikissa yksikön ***entity*** instansseissa.
InstanssienEnimmäismääräJos (*<relationship>*, *<number-attr>*, *<condition>*) **InstanssienEnimmäismääräJos** (*<relationship>*, *<date-attr>*, *<condition>*) **InstanssienEnimmäismääräJos** (*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanssienEnimmäismääräJos** (*<relationship>*, *<time-attr>*, *<condition>*)	Hakee muuttujan ***entity-level*** korkeimman/viimeisimmän arvon kaikissa yksikön ***entity*** instansseissa, joissa tietyllä määritteellä ***entity-level attribute*** on tietty arvo.

Syntaksi	Kuvaus
InstanssienVähimmäismäärä(*<relationship>*, *<number-attr>*) **InstanssienVähimmäismäärä**(*<relationship>*, *<date-attr>*) **InstanssienVähimmäismäärä**(*<relationship>*, *<datetime-attr>*) **InstanssienVähimmäismäärä**(*<relationship>*, *<time-attr>*)	Hakee muuttujan ***entity-level*** alhaisimman/vanhimman arvon kaikissa yksikön ***entity*** instansseissa.
InstanssienVähimmäismääräJos(*<relationship>*, *<number-attr>*, *<condition>*) **InstanssienVähimmäismääräJos**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanssienVähimmäismääräJos**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanssienVähimmäismääräJos**(*<relationship>*, *<time-attr>*, *<condition>*)	Hakee muuttujan ***entity-level*** alhaisimman/vanhimman arvon kaikissa yksikön ***entity*** instansseissa, joissa tietyllä määritteellä ***entity-level attribute*** on tietty arvo.
InstanssienSumma(*<relationship>*, *<number-attr>*)	Hakee kaikkien muuttujan ***entity-level*** instanssien summan.
InstanssienSummaJos(*<relationship>*, *<number-attr>*, *<condition>*)	Hakee kaikkien muuttujan ***entity-level*** sellaisten instanssien summan, joissa yksiköstä ***entity*** on tosi, että tietyn yksikkötason ***entity-level*** totuusarvo ***attribute*** on tosi.
InstanssinArvoJos(*<relationship>*, *<number-attr>*, *<condition>*) **InstanssinArvoJos**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanssinArvoJos**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanssinArvoJos**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanssinArvoJos**(*<relationship>*, *<time-attr>*, *<condition>*)	Hakee arvon yksilöivästä instanssista ***entity***, joka tunnistetaan ***relationship*** kohteen ***entity*** instansseista ehdolla. • Jos ehto määrittää yksittäisen kohteen ***entity*** instanssin, arvo on sitä ***entity*** instanssia vastaan laskettu arvo. • Jos useampi kuin yksi kohdeinstanssi täyttää ehdon, palautetaan ***uncertain***. • Jos yksikään kohdeinstanssi ei täytä ehtoa ja ***relationship*** tunnetaan, arvo on ***uncertain***.
InstanssiYhtäSuuriKuin(*<instance1>*, *<instance2>*)	Määrittää, ovatko yksikön ***entity*** kaksi instanssia samoja instansseja.
InstanssiErisuuriKuin(*<instance1>*, *<instance2>*)	Määrittää, ovatko yksikön ***entity*** kaksi instanssia eri instansseja.
JohdaInstanssi(*<relationship>*,	Käytetään päätelmänä sen johtamiseen, että yksikön ***entity*** instanssi on olemassa ja

Syntaksi	Kuvaus
<identity>) *<rel>***(***<identity>***) on olemassa**	on suhteen ***relationship*** jäsen.

Ajallisen perustelun funktiot(English)

Syntaksi	Kuvaus
VälienMääräEro(*<start-date>*, *<end-date>*, *<variable>*) **VälienMääräEro**(*<start-date>*, *<end-date>*, *<condition>*)	Laskee muuttujan tunnettujen erillisten arvojen määrän alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VälienMääräEroJos(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Laskee muuttujan tunnettujen erillisten arvojen määrän alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä vain aikoina, jolloin totuusarvosuodatin on tosi.
VäliPäivienSumma(*<start-date>*, *<end-date>*, *<number-attr>*)	Laskee valuutan tai numeromuuttujan summan alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä. Määritteen ***attribute*** oletetaan olevan päivittäinen määrä.
VäliPäivienSummaJos(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Laskee valuutan tai numeromuuttujan kaikkien päivittäisten arvojen summan alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä vain aikoina, jolloin ehto on tosi.
VäliEnimmäisarvo(*<start-date>*, *<end-date>*, *<number-attr>*) **VäliEnimmäisarvo**(*<start-date>*, *<end-date>*, *<date-attr>*) **VäliEnimmäisarvo**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **VäliEnimmäisarvo**(*<start-date>*, *<end-date>*, *<time-attr>*)	Valitsee muuttujan enimmäisarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VäliEnimmäisarvoJos(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **VäliEnimmäisarvoJos**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*)	Valitsee muuttujan enimmäisarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä vain aikoina, jolloin ehto on tosi.

Syntaksi	Kuvaus
VäliEnimmäisarvoJos (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **VäliEnimmäisarvoJos** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	
VäliVähimmäisarvo(*<start-date>*, *<end-date>*, *<number-attr>*) **VäliVähimmäisarvo**(*<start-date>*, *<end-date>*, *<date-attr>*) **VäliVähimmäisarvo**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **VäliVähimmäisarvo**(*<start-date>*, *<end-date>*, *<time-attr>*)	Valitsee muuttujan vähimmäisarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VäliVähimmäisarvoJos (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **VäliVähimmäisarvoJos** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **VäliVähimmäisarvoJos** (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **VäliVähimmäisarvoJos** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Valitsee muuttujan vähimmäisarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä vain aikoina, jolloin ehto on tosi.
VäliPainotettuKeskiarvo (*<start-date>*, *<end-date>*, *<number-attribute>*)	Laskee valuutan tai numeromuuttujan keskiarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä painotettuna aikavälillä, jota kukin arvo koskee.
VäliPainotettuKeskiarvoJos (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Laskee valuutan tai numeromuuttujan keskiarvon alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä vain aikoina, jolloin totuusarvoehto on tosi (painotettuna aikavälillä, jota kukin arvo koskee ja joissa suodatin on tosi).
VäliAina(*<start-date>*, *<end-date>*, *<condition>*)	Palauttaa arvon tosi vain jos totuusarvoehto on tosi kaikkina aikoina alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VäliPäiviäVähintään(*<start-*	Palauttaa arvon tosi vain jos totuusarvoehto on tosi vähintään määritettynä määränä päiviä

Syntaksi	Kuvaus
date>, *<end-date>*, *<NumDays>*, *<condition>*)	(ei välttämättä peräkkäisinä) alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VäliPeräkkäisiäPäiviä (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Palauttaa arvon tosi vain jos totuusarvoehto on tosi vähintään määritettynä määränä peräkkäisiä päiviä alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
VäliJoskus(*<start-date>*, *<end-date>*, *<condition>*)	Palauttaa arvon tosi vain jos totuusarvoehto on kertaakaan tosi alkupäivämäärän ***date*** (sisällyttävä) ja loppupäivämäärän ***date*** (poissulkeva) välillä.
Arvo(*<date>*, *<value>*)	Palauttaa annetun määritteen ***attribute*** arvon määritettynä päivämääränä ***date***.
MilloinViimeisin(*<date>*, *<condition>*)	Palauttaa päivämäärän ***date***, jona totuusarvoehto on viimeksi ollut tosi, määritetystä päivämäärästä (sisällyttävä) ***date*** taaksepäin.
MilloinSeuraava(*<date>*, *<condition>*)	Palauttaa päivämäärän ***date***, jona totuusarvoehto on seuraavan kerran tosi, määritetystä päivämäärästä (sisällyttävä) ***date*** eteenpäin.
Viimeisin()	Palauttaa arvon ***date***, joka vastaa myöhäisintä mahdollista arvoa ***date***. Arvon ***date*** taataan olevan myöhempi kuin mikään muu arvo ***date***, jonka ***date attribute*** voi ottaa tai lauseke voi arvioida.
Aikaisin()	Palauttaa arvon ***date***, joka vastaa aikaisinta mahdollista arvoa ***date***. Arvon ***date*** taataan olevan aikaisempi kuin mikään muu arvo ***date***, jonka ***date attribute*** voi ottaa tai lauseke voi arvioida.
AikaPäiviäHetkestä(*<date>*, *<end-date>*)	Palauttaa numeromuuttujan, joka muuttuu joka päivä ja on täysien päivien määrä ***date*** jälkeen.
AikaViikkojaHetkestä (*<date>*, *<end-date>*)	Palauttaa numeromuuttujan, joka muuttuu viikoittain ja on täysien viikkojen määrä ***date*** jälkeen.
AikaKuukausiaHetkestä (*<date>*, *<end-date>*)	Palauttaa numeroarvon, joka muuttuu kuukausittain ja on kokonaisten kuukausien määrä ***date*** alkaen. Huomautus: Jos annettu ***date*** on kuukauden 28. päivän jälkeen ja seuraavassa kuukaudessa on vähemmän päiviä kuin annetussa kuukaudessa, vuotuisen kuukauden vaihtopiste luodaan kyseisen kuukauden viimeiselle päivälle. Esimerkiksi jos annettu ***date*** on 28., 29., 30. tai 31. tammikuuta 2007, ensimmäinen vaihtopiste on 28. helmikuuta 2007.
AikaVuosiaHetkestä (*<date>*, *<end-date>*)	Palauttaa numeromuuttujan, joka muuttuu vuosittain ja on täysien vuosien määrä ***date*** jälkeen.
AikaAinaPäivät(*<days>*, *<condition>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja jonka arvo on tosi vain jos totuusarvoehto on tosi annettuna määränä edeltäviä päiviä tämänhetkinen päivä poissulkien.
AikaPeräkkäisiäPäiviä (*<minDays>*, *<days>*, *<condition>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi vain jos totuusarvoehto on tosi vähintään vähimmäismääränä peräkkäisiä päiviä minä tahansa aikana määrättynä määränä edeltäviä päiviä tämänhetkinen päivä poislukien.
AikaJoskusPäivät(*<days>*,	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi vain jos totuusarvoehto

Syntaksi	Kuvaus
<condition>)	on tosi jonakin päivänä annettuna määränä edeltäviä päiviä tämänhetkinen päivä poissulkien.
AikaJälkeen(*<date>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja jonka arvo on tosi päivämäärän ***date*** jälkeen ja epätosi kyseisenä päivänä ja aikaisemmin.
AikaEnnen(*<date>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi ennen päivämäärää ***date*** ja epätosi kyseisenä päivänä ja sitä myöhemmin.
Aikana(*<date>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi päivämääränä ***date*** ja epätosi sitä ennen ja sen jälkeen.
AikanaTaiJälkeen(*<date>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi päivämääränä ***date*** ja sen jälkeen ja epätosi sitä ennen.
AikanaTaiEnnen(*<date>*)	Palauttaa totuusarvon ***attribute***, joka vaihtelee eri aikoina ja on tosi päivämääränä ***date*** ja sitä ennen ja epätosi sitä myöhemmin.
AikaAloituspäivästä(*<relationship>*, *<date>*, *<value>*)	Palauttaa yksittäisen ajallisen määritteen ***attribute*** (lähteen ***entity*** tasolta) suhteesta ***relationship*** ja arvon ***attribute*** yksiköille, joiden arvot vaikuttavat alusta ***date attribute***.
AikaLopetuspäivästä(*<relationship>*, *<date>*, *<value>*)	Palauttaa yksittäisen ajallisen määritteen ***attribute*** (lähteen ***entity*** tasolta) suhteesta ***relationship*** ja arvon ***attribute*** yksiköille, joiden arvot vaikuttavat päivämäärään ***date attribute*** asti.
AikaAlueesta(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Palauttaa yksittäisen ajallisen määritteen ***attribute*** (lähteen ***entity*** tasolta) suhteesta ***relationship*** ja arvon ***attribute*** yksiköille, joiden arvot vaikuttavat alusta ***date attribute*** (sisällyttävä) loppuun ***date attribute*** (poissulkeva). Arvo on ***uncertain***, jos sen voimassaolo päättyy ennen seuraavaa alkua ***date***.
AikaOnViikonpäivä (*<startdate>*, *<enddate>*)	Palauttaa arvon tosi päivämäärinä, jotka ovat viikonpäiviä, ja epätosi päivämäärinä, jotka ovat viikonloppuja alkupäivämäärästä ***date*** (sisällyttävä) loppupäivämäärään ***date*** (poissulkeva). Palauttaa arvon ***uncertain*** alueen ***date*** ulkopuolella.
AikaKerranKuukaudessa (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Palauttaa arvon tosi, jos päivä on sama kuin kuukaudenpäivä-parametri, ja epätosi kaikkina muina kuukauden päivinä määritetystä alkupäivämäärästä ***date*** (sisällyttävä) loppupäivämäärään ***date*** (poissulkeva). Palauttaa arvon ***uncertain*** päivämäärän ***date*** alueen ulkopuolella. Jos kuukaudenpäivä ylittää tämän hetkisen kuukauden päivien määrän, arvo on tosi kyseisen kuukauden viimeisenä päivänä, joten toiminto palauttaa arvon tosi täsmälleen yhtenä päivänä kuukaudessa.

Tarkistustapahtuman toiminnot(English)

Syntaksi	Kuvaus
Virhe (*<text>*)	Järjestelmä lähettää virhetapahtuman avulla sanoman käyttäjälle ja estää käyttäjiä jatkamasta tutkimusta, kunnes virheen käynnistänyt ehto ei enää ole voimassa.

Syntaksi	Kuvaus
Varoitus (<*text*>)	Varoitustapahtuman avulla lähetetään sanoma käyttäjälle, mutta käyttäjän sallitaan jatkaa varoituksen käynnistäneestä ehdosta huolimatta.

Hylätyt funktiot(English)

Syntaksi	Kuvaus
KutsuRäätälöityFunktio (<*A*>, <*B*>)	Palauttaa koodikirjaston ulkoisen kutsun tuloksen. Koodikirjaston täytyy olla määritysohjelman käytettävissä, jotta räätälöity toiminnon kutsu onnistuu.

Connecteurs logiques(English)

Syntaxe	Description
si	Condition facultative apparaissant à la fin d'une ligne de conclusion qui est suivie d'une preuve
et	Conjonction logique entre deux attributs ***attributes***
ou	Disjonction logique entre deux attributs ***attributes***
soit **un de** **au moins un de la suite est vrai** **n'importe lequel de la suite est satisfait** **n’importe lequel de la suite est satisfait**	Elément de regroupement utilisé avec des disjonctions, où deux attributs ***attributes*** ou plus doivent être regroupés
les deux **tout** **tous** **tous dans la suite sont vrais** **tous dans la suite sont satisfaits**	Elément de regroupement utilisé avec des conjonctions, où deux attributs ***attributes*** ou plus doivent être regroupés
autrement	Condition apparaissant à la fin d'une règle de table pour indiquer la clause autrement
est	Condition utilisée dans une entrée de légende entre l'expression abrégée et le texte ***text*** attribut ***attribute*** complet

Fonctions logiques(English)

Syntaxe	Description
ce n'est pas vrai que *<expr>*	Opérateur utilisé pour renvoyer la valeur Vrai si l'attribut ***attribute*** a une valeur qui est fausse
<var> **est certain** *<var>* **est certaine** **il est certain** [**ou pas**] **que** *<expr>*	Opérateur utilisé pour renvoyer la valeur Vrai si l'attribut ***attribute*** a une valeur qui n'est pas "incertain" ***uncertain***
<var> **est incertain** *<var>* **est incertaine** *<var>* **n'est pas certain** *<var>* **n'est pas certaine** **il est incertain** [**ou pas**] **que** *<expr>* **il n'est pas certain que** *<expr>* **incertain** **incertaine**	Opérateur utilisé pour renvoyer la valeur Vrai si la valeur de l'attribut ***attribute*** est "incertain" ***uncertain***
<var> **est connu** *<var>* **est connue** **il est connu** [**ou pas**] **que** *<expr>* **il est actuellement connu** [**ou pas**] **que** *<expr>*	Opérateur utilisé pour renvoyer la valeur Vrai si l'attribut ***attribute*** a une valeur
<var> **est inconnu** *<var>* **est inconnue** **il est inconnu** [**ou pas**] **que** *<expr>* **inconnu** **inconnue**	Opérateur utilisé pour renvoyer la valeur Vrai si l'attribut ***attribute*** n'a pas de valeur

Constantes logiques(English)

Syntaxe	Description
vrai	Valeur Vrai constante utilisé pour les règles de table.
faux	Valeur Faux constante utilisée pour les règles de table.
incertain	Valeur "incertain" ***uncertain*** constante utilisée pour les règles de table.

Opérateurs de comparaison(English)

Syntaxe	Description
*<lhs>***<***<rhs>* *<lhs>* **est inférieur à** *<rhs>*	Inférieur à Note : Il n'existe pas de langage naturel quand cet opérateur est utilisé avec des valeurs

Syntaxe	Description
<lhs> **est inférieure à** *<rhs>* *<lhs>* **est antérieur à** *<rhs>* *<lhs>* **est antérieure à** *<rhs>*	numériques et de devise.
<lhs> **>** *<rhs>* *<lhs>* **est supérieur à** *<rhs>* *<lhs>* **est supérieure à** *<rhs>* *<lhs>* **est postérieur à** *<rhs>* *<lhs>* **est postérieure à** *<rhs>*	Supérieur à Note : Il n'existe pas de langage naturel quand cet opérateur est utilisé avec des valeurs numériques et de devise.
*<lhs>***<=***<rhs>* *<lhs>* **est inférieur ou égale à** *<rhs>* *<lhs>* **est inférieure ou égale à** *<rhs>*	Inférieur ou égal à
<lhs> **>=** *<rhs>* *<lhs>* **est supérieur ou égale à** *<rhs>* *<lhs>* **est supérieure ou égale à** *<rhs>*	Supérieur ou égal à
*<lhs>***=***<rhs>* *<lhs>* **est égale à** *<rhs>* *<lhs>* **égale** *<rhs>*	Egale
<lhs> **n'est pas égale à** *<rhs>* *<lhs>* **n'égale pas** *<rhs>* *<lhs>* **<>** *<rhs>* *<lhs>* **!=** *<rhs>*	Différent de

Fonctions numériques(English)

Syntaxe	Description
Nombre(*<numText>*)	Convertit la chaîne spécifiée en une valeur numérique
<x> **+** *<y>*	Addition
<x> **-** *<y>*	Soustraction
<lhs> * *<rhs>*	Multiplication
<lhs> **/** *<rhs>*	Division
<lhs> **** *<rhs>*	Division entière
<lhs> **modulo** *<rhs>*	Reste après une division entière

Syntaxe	Description
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **le plus grand de** *<val1>* **et de** *<val2>* **la plus grande de** *<val1>* **et de** *<val2>* **le dernier de** *<val1>* **et** *<val2>* **la dernière** [**date**] **de** *<val1>* **et** *<val2>*	Renvoie la valeur supérieure entre deux valeurs
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **le plus petit de** *<val1>* **et de** *<val2>* **la plus petite de** *<val1>* **et de** *<val2>* **le premier de** *<val1>* **et de** *<val2>* **la première** [**date**] **de** *<val1>* **et de** *<val2>*	Renvoie la valeur inférieure entre deux valeurs
Xy(*<x>*, *<y>*) *<val>* **élevé à la puissance** *<power>*	x à la puissance y
Ex(*<x>*) **e à la puissance** *<log-val>*	Constante e à la puissance x
Abs(*<x>*) **la valeur absolue de** *<val>* **|** *<val>* **|**	Valeur absolue de x
Ln(*<x>*) **le logarithme naturel de** *<log-val>*	Logarithme népérien de x
Log(*<x>*) **le logarithme à base 10 de** *<log-val>*	Logarithme en base 10 de x
RacineCarrée(*<x>*) **la racine carrée de** *<val>*	Racine carrée de x
Arrondi(*<x>*, *<n>*) *<val>* **arrondi à** *<num_places>* **décimale** *<val>* **arrondi à** *<num_places>* **décimales**	Arrondit x à n positions décimales
Troncation(*<x>*, *<n>*) *<val>* **tronqué à** *<num_places>* **décimale** *<val>* **tronqué à** *<num_places>* **décimales**	x tronqué à n positions décimales
Sin(*<x>*)	Sinus de x
Cos(*<x>*)	Cosinus de x
Tan(*<x>*)	Tangente de x
Asin(*<x>*)	Arcsinus de x
Acos(*<x>*)	Arccosinus de x

Syntaxe	Description
Atan(*<x>*)	Alrctangente de x

Fonctions de date(English)

Syntaxe	Description
DateActuelle() **la date actuelle**	Renvoie la date ***date*** actuelle au début de la session.
Date(*<text>*)	Convertit la chaîne spécifiée en une valeur de date ***date***
DéfinirDate(*<year>*, *<month>*, *<day>*)	Renvoie une date ***date*** constituée de l'année, du mois et du jour spécifiés.
ExtraireJour (*<date/datetime>*)	Renvoie le composant jour d'un attribut date/date-heure ***date/datetime attribute***.
ExtraireMois (*<date/datetime>*)	Renvoie le composant mois d'un attribut date/date-heure ***date/datetime attribute***.
ExtraireAnnée(*<date/datetime>*)	Renvoie le composant année d'un attribut date/date-heure ***date/datetime attribute***.
JourSuivantSemaine(*<date/datetime>*, *<day>*) **le lundi** [**suivant**][**au moment où**] **ou après** *<from-date>* **le lundi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le mardi** [**suivant**][**au moment où**] **ou après** *<from-date>* **le mardi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le mercredi** [**suivant**][**au moment où**] **ou après** *<from-date>* **le mercredi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le jeudi** [**suivant**][**au moment où**] **ou après** *<from-*	Renvoie la date ***date*** du jour de semaine suivant à une date ***date*** ou avant/après celle-ci (selon la syntaxe utilisée).

Syntaxe	Description
date> **le jeudi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le vendredi** [**suivant**][**au moment où**] **ou après** *<from-date>* **le vendredi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le samedi** [**suivant**][**au moment où**] **ou après** *<from-date>* **le samedi** [**précédant**][**au moment où**] **ou avant** *<from-date>* **le dimanche** [**suivant**][**au moment où**] **ou après** *<from-date>* **le dimanche** [**précédant**][**au moment où**] **ou avant** *<from-date>*	
JourSuivant(*<date>*, *<day>*, *<month>*)	Renvoie l'instance suivante du jour et du mois donnés après une date ***date***.
AjouterJours(*<date/datetime>*, *<num_days>*) **la date** *<num_days>* **jour après** *<date>* **la date** *<num_days>* **jours après** *<date>* **la date** *<num_days>* **jour avant** *<date>* **la date** *<num_days>* **jours avant** *<date>* **le temps** *<num_days>* **jour après** *<date>* **le temps** *<num_days>* **jours après** *<date>* **le temps** *<num_days>* **jour avant** *<date>* **le temps** *<num_days>* **jours avant** *<date>*	Ajoute un nombre de jours à une valeur de date ***date*** ou soustrait un nombre de jours de cette valeur. En cas d'utilisation d'une forme syntaxique concise, le nombre doit être un nombre entier positif pour ajouter des jours à la valeur de date ***date*** saisie, ou un nombre entier négatif pour soustraire des jours de la valeur de date ***date*** saisie.

Syntaxe	Description
AjouterSemaines(*<date/datetime>*, *<num_weeks>*) **la date** *<num_weeks>* **semaine après** *<date>* **la date** *<num_weeks>* **semaines après** *<date>* **la date** *<num_weeks>* **semaine avant** *<date>* **la date** *<num_weeks>* **semaines avant** *<date>* **le temps** *<num_weeks>* **semaine après** *<date>* **le temps** *<num_weeks>* **semaines après** *<date>* **le temps** *<num_weeks>* **semaine avant** *<date>* **le temps** *<num_weeks>* **semaines avant** *<date>*	Ajoute un nombre de semaines à une date ***date***. Lors de l'utilisation d'une forme syntaxique concise, le nombre doit être un nombre entier positif pour ajouter des semaines à la date ***date*** saisie.
AjouterMois(*<date/datetime>*, *<num_months>*) **la date** *<num_months>* **mois après** *<date>* **la date** *<num_months>* **mois avant** *<date>* **le temps** *<num_months>* **mois après** *<date>* **le temps** *<num_months>* **mois avant** *<date>*	Ajoute un nombre de mois à une date ***date***. Lors de l'utilisation d'une forme syntaxique concise, le nombre doit être un nombre entier positif pour ajouter des mois à la date ***date*** saisie.
AjouterAnnées(*<date/datetime>*, *<num_years>*) **la date** *<num_years>* **an après** *<date>* **la date** *<num_years>* **ans après** *<date>* **la date** *<num_years>* **an avant** *<date>* **la date** *<num_years>* **ans avant** *<date>* **le temps** *<num_years>* **an après** *<date>* **le temps** *<num_years>* **ans après** *<date>* **le temps** *<num_years>* **an**	Ajoute un nombre d'années à une date ***date***. Lors de l'utilisation d'une forme syntaxique concise, le nombre doit être un nombre entier positif pour ajouter des années à la date ***date*** saisie.

Syntaxe	Description
avant *<date>* **le temps** *<num_years>* **ans avant** *<date>*	
NombreJoursSemaine (*<date1>*, *<date2>*) **le nombre de jours de la semaine** [**entiers**] **de** *<date1>* **à** *<date2>* **le nombre de jours de la semaine** [**entiers**] **entre** *<date1>* **et** *<date2>*	Compte le nombre de jours de semaine entre ***date***1 et ***date***2, autrement dit le nombre de jours compris entre le lundi et le vendredi. Note : La première date ***date*** est incluse et la dernière date ***date*** est exclue.
DébutAnnée (*<date/datetime>*) **le premier jour de l'année dans lequel** *<from-date>* **arrive** **le premier jour de l'année dans laquelle** *<from-date>* **arrive**	Renvoie la première date ***date*** de l'année dans laquelle tombe la date ***date***.
FinAnnée(*<date/datetime>*) **le dernier jour de l'année dans lequel** *<from-date>* **arrive** **le dernier jour de l'année dans laquelle** *<from-date>* **arrive**	Renvoie la dernière date ***date*** de l'année dans laquelle tombe la date ***date***.
DifférenceJours(*<date/datetime1>*, *<date/datetime2>*) **le nombre de jours** [**entiers**] **de** *<date1>* **à** *<date2>* **le nombre de jours** [**entiers**] **entre** *<date1>* **et** *<date2>*	Renvoie le nombre de jours complets entre ***date/datetime1*** et ***date/datetime2***. L'ordre des deux dates n'affecte pas le résultat.
DifférenceJoursInclus (*<date/datetime1>*, *<date/datetime2>*) **le nombre de jours** [**entiers**] **(inclusif) de** *<date1>* **à** *<date2>* **le nombre de jours** [**entiers**] **(inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de jours complets entre ***date/datetime1*** et ***date/datetime2*** (inclus). Ce calcul inclut les deux extrémités. Lorsque les dates sont identiques, le résultat est 1. L'ordre des deux dates n'affecte pas le résultat.

Syntaxe	Description
DifférenceJoursExclus(*<date/datetime1>*, *<date/datetime2>*) **le nombre de jours** [**entiers**] **(exclusif) de** *<date1>* **à** *<date2>* **le nombre de jours** [**entiers**] **(exclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de jours complets entre ***date/datetime1*** et ***date/datetime2*** (exclus). Ce calcul exclut les deux extrémités. Lorsque les dates sont identiques, le résultat est 0. L'ordre des deux dates n'affecte pas le résultat.
DifférenceSemaines(*<date/datetime1>*, *<date/datetime2>*) **le nombre de semaines** [**entières**] **de** *<date1>* **à** *<date2>* **le nombre de semaines** [**entières**] **entre** *<date1>* **et** *<date2>*	Renvoie le nombre de semaines complètes écoulées entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2***. L'ordre des deux dates n'affecte pas le résultat.
DifférenceSemainesIncluses(*<date/datetime1>*, *<date/datetime2>*) **le nombre de semaines** [**entières**] **(inclusif) de** *<date1>* **à** *<date2>* **le nombre de semaines** [**entières**] **(inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de semaines complètes écoulées entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** incluses. L'ordre des deux dates n'affecte pas le résultat.
DifférenceSemainesExclues(*<date/datetime1>*, *<date/datetime2>*) **le nombre de semaines** [**entières**] **(exclusif) de** *<date1>* **à** *<date2>* **le nombre de semaines** [**entières**] **(exclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de semaines complètes écoulées entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** exclues. L'ordre des deux dates n'affecte pas le résultat.
DifférenceMois(*<date/datetime1>*, *<date/datetime2>*) **le nombre de mois** [**entiers**] **de** *<date1>* **à** *<date2>* **le nombre de mois** [**entiers**] **entre** *<date1>* **et** *<date2>*	Renvoie le nombre de mois complets écoulés entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2***. L'ordre des deux dates n'affecte pas le résultat.

Syntaxe	Description
DifférenceMoisInclus (*<date/datetime1>*, *<date/datetime2>*) **le nombre de mois** [**entiers**] **(inclusif) de** *<date1>* **à** *<date2>* **le nombre de mois** [**entiers**] **(inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de mois complets écoulés entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** incluses. L'ordre des deux dates n'affecte pas le résultat.
DifférenceMoisExclus (*<date/datetime1>*, *<date/datetime2>*) **le nombre de mois** [**entiers**] **(exclusif) de** *<date1>* **à** *<date2>* **le nombre de mois** [**entiers**] **(exclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de mois complets écoulés entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** exclues. L'ordre des deux dates n'affecte pas le résultat.
DifférenceAnnées(*<date/datetime1>*, *<date/datetime2>*) **le nombre d'années** [**entières**] **de** *<date1>* **à** *<date2>* **le nombre d'années** [**entières**] **entre** *<date1>* **et** *<date2>*	Renvoie le nombre d'années entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2***. L'ordre des deux dates n'affecte pas le résultat.
DifférenceAnnéesIncluses (*<date/datetime1>*, *<date/datetime2>*) **le nombre d'années** [**entières**] **(inclusif) de** *<date1>* **à** *<date2>* **le nombre d'années** [**entières**] **(inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de jours entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** (incluses). L'ordre des deux dates n'affecte pas le résultat.
DifférenceAnnéesExclues (*<date/datetime1>*, *<date/datetime2>*) **le nombre d'années** [**entières**] **(exclusif) de** *<date1>* **à** *<date2>*	Renvoie le nombre de jours entre date/date-heure1 ***date/datetime1*** et date/date-heure2 ***date/datetime2*** (exclues). L'ordre des deux dates n'affecte pas le résultat.

Syntaxe	Description
le nombre d'années [**entières**] **(exclusif) entre** *<date1>* **et** *<date2>*	

Fonctions d'heure du jour(English)

Syntaxe	Description
HeureJour(*<text>*)	Convertit la chaîne donnée en une heure du jour
ExtraireSeconde(*<time/datetime>*)	Renvoie le composant seconde d'un attribut heure du jour/date-heure ***timeofday/datetime attribute***.
ExtraireMinute(*<time/datetime>*)	Renvoie le composant minute d'un attribut heure du jour/date-heure ***timeofday/datetime attribute***.
ExtraireHeure(*<time/datetime>*)	Renvoie le composant heure d'un attribut heure du jour/date-heure ***timeofday/datetime attribute***.

Fonctions de date et heure(English)

Syntaxe	Description
DateHeureActuelles()	Renvoie les date ***date*** et heure actuelles au début de la session.
DateEtHeure(*<text>*)	Convertit la chaîne spécifiée en une valeur de date-heure ***datetime***
DateHeureConcaténées (*<date>*, *<time>*)	Définit la date ***date*** et heure en joignant la date ***date*** et l'heure.
DifférenceSecondes(*<datetime1>*, *<datetime2>*) **DifférenceSecondes** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre de secondes de** *<date1>* **à** *<date2>* **le nombre de secondes entre** *<date1>* **et** *<date2>*	Renvoie le nombre de secondes entre ***datetime1*** et ***datetime2***.
DifférenceSecondesIncluses (*<datetime1>*, *<datetime2>*) **DifférenceSecondesIncluses** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre de secondes**	Renvoie le nombre de secondes entre les dates et heures ***datetime1*** et ***datetime2*** (incluses).

Syntaxe	Description
(inclusif) de *<date1>* **à** *<date2>* **le nombre de secondes (inclusif) entre** *<date1>* **et** *<date2>*	
DifférenceSecondesExclues (*<datetime1>*, *<datetime2>*) **DifférenceSecondesExclues** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre de secondes (exclusif) de** *<date1>* **à** *<date2>* **le nombre de secondes (exclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de secondes entre les dates et heures ***datetime1*** et ***datetime2*** (exclues).
DifférenceMinutes(*<datetime1>*, *<datetime2>*) **DifférenceMinutes** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre de minutes de** *<date1>* **à** *<date2>* **le nombre de minutes entre** *<date1>* **et** *<date2>*	Renvoie le nombre de minutes entre ***datetime1*** et ***datetime2***.
DifférenceMinutesIncluses (*<datetime1>*, *<datetime2>*) **DifférenceMinutesIncluses** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre de minutes (inclusif) de** *<date1>* **à** *<date2>* **le nombre de minutes (inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre de minutes entre les dates et heures ***datetime1*** et ***datetime2*** (incluses).
DifférenceMinutesExclues (*<datetime1>*, *<datetime2>*) **DifférenceMinutesExclues** (*<timeOfDay1>*, *<timeOfDay2>*)	Renvoie le nombre de minutes entre les dates et heures ***datetime1*** et ***datetime2*** (exclues).

Syntaxe	Description
le nombre de minutes (exclusif) de *<date1>* **à** *<date2>* **le nombre de minutes (exclusif) entre** *<date1>* **et** *<date2>*	
DifférenceHeures(*<datetime1>*, *<datetime2>*) **DifférenceHeures** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre d'heures de** *<date1>* **à** *<date2>* **le nombre d'heures entre** *<date1>* **et** *<date2>*	Renvoie le nombre d'heures entre ***datetime1*** et ***datetime2***.
DifférenceHeuresIncluses (*<datetime1>*, *<datetime2>*) **DifférenceHeuresIncluses** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre d'heures (inclusif) de** *<date1>* **à** *<date2>* **le nombre d'heures (inclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre d'heures entre les dates et heures ***datetime1*** et ***datetime2*** (incluses).
DifférenceHeuresExclues (*<datetime1>*, *<datetime2>*) **DifférenceHeuresExclues** (*<timeOfDay1>*, *<timeOfDay2>*) **le nombre d'heures (exclusif) de** *<date1>* **à** *<date2>* **le nombre d'heures (exclusif) entre** *<date1>* **et** *<date2>*	Renvoie le nombre d'heures entre les dates et heures ***datetime1*** et ***datetime2*** (exclues).
ExtraireDate(*<datetime>*)	Extrait la date ***date*** à partir d'un attribut date-heure ***datetime attribute***.
ExtraireHeureJour(*<datetime>*)	Renvoie l'heure du jour d'un attribut date-heure ***datetime attribute***. Peut être utilisé pour définir la valeur d'un attribut heure du jour ***timeofday attribute*** avec l'heure à laquelle la règle est exécutée en extrayant l'heure des date ***date*** et heure actuelles.

Syntaxe	Description
AjouterHeures(*<datetime>*, *<num_hours>*) **AjouterHeures**(*<timeOfDay>*, *<num_hours>*) **le temps** *<num_hours>* **heure après** *<date>* **le temps** *<num_hours>* **heures après** *<date>* **le temps** *<num_hours>* **heure avant** *<date>* **le temps** *<num_hours>* **heures avant** *<date>* **le temps de jour** *<num_hours>* **heure aprés** *<time-of-day>* **le temps de jour** *<num_hours>* **heures aprés** *<time-of-day>* **le temps de jour** *<num_hours>* **heure avant** *<time-of-day>* **le temps de jour** *<num_hours>* **heures avant** *<time-of-day>*	Ajoute un nombre d'heures à une date ***date***.
AjouterMinutes(*<datetime>*, *<num_minutes>*) **AjouterMinutes** (*<timeOfDay>*, *<num_minutes>*) **le temps** *<num_minutes>* **minute après** *<date>* **le temps** *<num_minutes>* **minutes après** *<date>* **le temps** *<num_minutes>* **minute avant** *<date>* **le temps** *<num_minutes>* **minutes avant** *<date>* **le temps de jour** *<num_minutes>* **minute aprés** *<time-of-day>* **le temps de jour** *<num_minutes>* **minutes aprés** *<time-of-day>*	Ajoute un nombre de minutes à une date ***date***.

Syntaxe	Description
le temps de jour *<num_ minutes>* **minute avant** *<time-of-day>* **le temps de jour** *<num_ minutes>* **minutes avant** *<time-of-day>*	
AjouterSecondes (*<datetime>*, *<num_seconds>*) **AjouterSecondes** (*<timeOfDay>*, *<num_ seconds>*) **le temps** *<num_seconds>* **seconde après** *<date>* **le temps** *<num_seconds>* **secondes après** *<date>* **le temps** *<num_seconds>* **seconde avant** *<date>* **le temps** *<num_seconds>* **secondes avant** *<date>* **le temps de jour** *<num_ seconds>* **seconde aprés** *<time-of-day>* **le temps de jour** *<num_ seconds>* **secondes aprés** *<time-of-day>* **le temps de jour** *<num_ seconds>* **seconde avant** *<time-of-day>* **le temps de jour** *<num_ seconds>* **secondes avant** *<time-of-day>*	Ajoute un nombre de secondes à une date ***date***.

Fonctions de texte(English)

Syntaxe	Description
<text1> **&** *<text2>*	Combine la valeur de texte ***text1*** avec la valeur de texte ***text2***, et ainsi de suite, afin de former une valeur de texte ***text*** unique. Note : Vous pouvez utiliser des variables de tout type. Les valeurs sont formatées avec le formateur installé dans la session de règles.
la concatenation de *<text1>* **&** *<text2>*	Combine la valeur de texte ***text1*** avec la valeur de texte ***text2***, et ainsi de suite, afin de former une valeur de texte ***text*** unique. Note : Vous pouvez utiliser des variables de tout type. Les valeurs sont formatées avec le formateur

Syntaxe	Description
	installé dans la session de règles.
Contient(*<text>*, *<substring>*) *<text>* **contient** *<substring>*	Renvoie une valeur booléenne indiquant si la valeur de texte ***text*** donnée contient la sous-chaîne de texte ***text*** donnée. La comparaison ***text*** ne respecte pas la casse.
SeTerminePar (*<text>*, *<substring>*) *<text>* **se termine avec** *<substring>*	Renvoie une valeur booléenne indiquant si la valeur de texte ***text*** donnée se termine par la sous-chaîne de texte ***text*** donnée. La comparaison ***text*** ne respecte pas la casse.
EstNombre(*<text>*) *<text>* **est un nombre**	Renvoie une valeur booléenne indiquant si la valeur de texte ***text*** donnée représente un nombre valide.
Longueur(*<text>*)	Renvoie la longueur en caractères de la valeur de texte ***text*** donnée.
CommencePar (*<text>*, *<substring>*) *<text>* **commence par** *<substring>*	Renvoie une valeur booléenne indiquant si la valeur de texte ***text*** donnée commence par la sous-chaîne de texte ***text*** donnée. La comparaison ***text*** ne respecte pas la casse.
Sous-chaîne (*<text>*, *<offset>*, *<length>*)	Renvoie la sous-chaîne de texte ***text*** commençant à l'emplacement donné, c'est-à-dire la longueur spécifiée en caractères. Si la fin de la chaîne est atteinte, un nombre inférieur de caractères est renvoyé.
Texte(*<number>*) **Texte**(*<date>*) **Texte**(*<datetime>*) **Texte** (*<timeOfDay>*)	Convertit l'attribut ***date attribute*** ou le nombre spécifié ou en une valeur de texte ***text***.

Fonctions d'entité et de relation(English)

Syntaxe	Description
Pour(*<relationship>*, *<Exp>*) **dans le cas de** *<ent>***,** *<attr>* *<val>***, dans le cas où c'est** *<ent>*	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à 1", "n à 1" ou "n à n", lorsqu'il n'y a qu'une condition.

Syntaxe	Description
ChampPour(*<relationship>*, *<alias>*) **ChampPour**(*<relationship>*) **dans le cas de** *<ent>*	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à 1", "n à 1" ou "n à n", lorsqu'il y a une ou plusieurs conditions.
PourTous(*<relationship>*, *<Exp>*) **chacun de** *<ent-attr>* **chacune de** *<ent-attr>* **pour chacun de** *<ent>***,** *<attr>* **pour chacune de** *<ent>***,** *<attr>* **pour tous** *<ent>***,** *<attr>* **pour toutes** *<ent>***,** *<attr>*	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à n" ou "n à n", lorsque vous devez déterminer si tous les membres du groupe d'entités ***entity*** cible doivent satisfaire la règle. Cette forme est utilisée lorsque la règle ne contient qu'une seule condition.
ChampPourTous(*<relationship>*) **ChampPourTous**(*<relationship>*, *<alias>*) **pour tous** *<ent>* **pour toutes** *<ent>* [**pour**] **chacun de** *<ent>* [**pour**] **chacune de** *<ent>* [**pour**] **tous** *<ent>* **(**<alias>**)** [**pour**] **toutes** *<ent>* **(**<alias>**)** [**pour**] **chacun de** *<ent>* **(**<alias>**)** [**pour**] **chacune de** *<ent>* **(**<alias>**)**	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à n" ou "n à n", lorsque vous devez déterminer si tous les membres du groupe d'entités ***entity*** cible doivent satisfaire la règle. Cette forme est utilisée lorsque la règle contient une ou plusieurs conditions.
Existe(*<relationship>*, *<Exp>*) **au moins un de** *<ent-attr>* **au moins une de** *<ent-attr>* **pour au moins un de** *<ent>***,** *<attr>* **pour au moins une de** *<ent>***,** *<attr>*	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à n" ou "n à n", lorsque vous devez déterminer si certains membres du groupe d'entités ***entity*** cible doivent satisfaire la règle. Cette forme est utilisée lorsque la règle ne contient qu'une seule condition.

Syntaxe	Description
ChampExiste(*<relationship>*) **ChampExiste**(*<relationship>*, *<alias>*) [**pour**] **au moins un de** *<ent>* [**pour**] **au moins une de** *<ent>* [**pour**] **au moins un de** *<ent>* **(**<alias>**)** [**pour**] **au moins une de** *<ent>* **(**<alias>**)**	Utilisé pour une référence d'une entité ***entity*** à une autre entité ***entity*** dans une relation ***relationship*** "1 à n" ou "n à n", lorsque vous devez déterminer si certains membres du groupe d'entités ***entity*** cible doivent satisfaire la règle. Cette forme est utilisée lorsque la règle contient une ou plusieurs conditions.
EstMembreDe(*<target>*, *<relationship>*) **EstMembreDe**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **es un miembro de** *<ent>* <ent-target> **(**<alias>**) es un miembro de** *<ent>* **IsMemberOf(**<ent-target>**,** *<ent>***)** **IsMemberOf(**<ent-target>**,** <alias>**,** *<ent>***)**	Utilisé en tant que conclusion pour inférer qu'une instance d'entité ***entity*** est un membre de la relation ***relationship***. Utilisé comme condition pour tester si une instance d'entité ***entity*** est la cible d'une relation ***relationship*** dont une deuxième instance d'entité ***entity*** est la source.
PasMembreDe(*<target>*, *<relationship>*)	Utilisé comme condition pour tester si une instance d'entité ***entity*** n'est pas la cible d'une relation ***relationship*** dont une deuxième instance d'entité ***entity*** est la source.
InstancesNombre(*<relationship>*) **le nombre de** *<ent>*	Compte le nombre d'instances existant pour une entité ***entity***.
InstancesNombreSi(*<relationship>*, *<Exp>*) **le nombre de** *<ent>* **dans le cas où** *<condition>*	Compte le nombre d'instances d'une entité ***entity*** pour laquelle un attribut de niveau entité ***entity-level attribute*** donné a une valeur particulière.
InstancesMaximum (*<relationship>*, *<number-attr>*) **InstancesMaximum** (*<relationship>*, *<date-attr>*)	Obtient la valeur la plus élevée/la plus récente d'une variable de niveau entité ***entity-level*** pour toutes les instances de l'entité ***entity***.

Syntaxe	Description
InstancesMaximum (*<relationship>*, *<datetime-attr>*) **InstancesMaximum** (*<relationship>*, *<time-attr>*) **le plus grand de** *<attr>* **pour tous** *<ent>* **la plus grande de** *<attr>* **pour tous** *<ent>* **le plus grand de** *<attr>* **pour toutes** *<ent>* **la plus grande de** *<attr>* **pour toutes** *<ent>* **le dernier de** *<attr>* **pour tous** *<ent>* **la dernière** [**date**] **de** *<attr>* **pour tous** *<ent>* **le dernier de** *<attr>* **pour toutes** *<ent>* **la dernière** [**date**] **de** *<attr>* **pour toutes** *<ent>* *<attr>*[**qui**] **est le dernier pour tous** *<ent>* *<attr>*[**qui**] **est la dernière** [**date**] **pour tous** *<ent>* *<attr>*[**qui**] **est le dernier pour toutes** *<ent>* *<attr>*[**qui**] **est la dernière** [**date**] **pour toutes** *<ent>* *<attr>*[**qui**] **est le plus grand pour tous** *<ent>* *<attr>*[**qui**] **est la plus grande pour tous** *<ent>* *<attr>*[**qui**] **est le plus grand pour toutes** *<ent>* *<attr>*[**qui**] **est la plus grande pour toutes** *<ent>*	

Syntaxe	Description
InstancesMaximumSi (*<relationship>*, *<number-attr>*, *<condition>*) **InstancesMaximumSi** (*<relationship>*, *<date-attr>*, *<condition>*) **InstancesMaximumSi** (*<relationship>*, *<datetime-attr>*, *<condition>*) **InstancesMaximumSi** (*<relationship>*, *<time-attr>*, *<condition>*) **le plus grand de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **la plus grande de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **le plus grand de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* **la plus grande de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* **le dernier de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **la dernière** [**date**] **de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **le dernier de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* **la dernière** [**date**] **de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le dernier pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la dernière** [**date**] **pour tous** *<ent>* **dans le cas où** *<ent-test>*	Obtient la valeur la plus élevée/la plus récente d'une variable de niveau entité ***entity-level*** pour toutes les instances de l'entité ***entity*** pour laquelle un attribut de niveau entité ***entity-level attribute*** donné a une valeur particulière.

Syntaxe	Description
<attr>[**qui**] **est le dernier pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la dernière** [**date**] **pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le plus grand pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la plus grande pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le plus grand pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la plus grande pour toutes** *<ent>* **dans le cas où** *<ent-test>*	
InstancesMinimum(*<relationship>*, *<number-attr>*) **InstancesMinimum**(*<relationship>*, *<date-attr>*) **InstancesMinimum**(*<relationship>*, *<datetime-attr>*) **InstancesMinimum**(*<relationship>*, *<time-attr>*) **le plus petit de** *<attr>* **pour tous** *<ent>* **la plus petite de** *<attr>* **pour tous** *<ent>* **le plus petit de** *<attr>* **pour toutes** *<ent>* **la plus petite de** *<attr>* **pour toutes** *<ent>* **le premier de** *<attr>* **pour tous** *<ent>* **la première** [**date**] **de** *<attr>* **pour tous** *<ent>* **le premier de** *<attr>* **pour toutes** *<ent>*	Obtient la valeur la plus basse/la moins récente d'une variable de niveau entité ***entity-level*** pour toutes les instances de l'entité ***entity***.

Syntaxe	Description
la première [**date**] **de** *<attr>* **pour toutes** *<ent>* *<attr>*[**qui**] **est le premier pour tous** *<ent>* *<attr>*[**qui**] **est la première** [**date**] **pour tous** *<ent>* *<attr>*[**qui**] **est le premier pour toutes** *<ent>* *<attr>*[**qui**] **est la première** [**date**] **pour toutes** *<ent>* *<attr>*[**qui**] **est le plus petit pour tous** *<ent>* *<attr>*[**qui**] **est la plus petite pour tous** *<ent>* *<attr>*[**qui**] **est le plus petit pour toutes** *<ent>* *<attr>*[**qui**] **est la plus petite pour toutes** *<ent>*	
InstancesMinimumSi (*<relationship>*, *<number-attr>*, *<condition>*) **InstancesMinimumSi** (*<relationship>*, *<date-attr>*, *<condition>*) **InstancesMinimumSi** (*<relationship>*, *<datetime-attr>*, *<condition>*) **InstancesMinimumSi** (*<relationship>*, *<time-attr>*, *<condition>*) **le plus petit de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **la plus petite de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **le plus petit de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>*	Obtient la valeur la plus basse/la moins récente d'une variable de niveau entité ***entity-level*** pour toutes les instances de l'entité ***entity*** pour laquelle un attribut de niveau entité ***entity-level attribute*** donné a une valeur particulière.

Syntaxe	Description
la plus petite de *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* **le premier de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **la première** [**date**] **de** *<attr>* **pour tous** *<ent>* **dans le cas où** *<ent-test>* **le premier de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* **la première** [**date**] **de** *<attr>* **pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le premier pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la première** [**date**] **pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le premier pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la première** [**date**] **pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le plus petit pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la plus petite pour tous** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est le plus petit pour toutes** *<ent>* **dans le cas où** *<ent-test>* *<attr>*[**qui**] **est la plus petite pour toutes** *<ent>* **dans le cas où** *<ent-test>*	
InstancesSomme(*<relationship>*, *<number-attr>*)	Obtient la somme de toutes les instances d'une variable de niveau entité ***entity-level***.

Syntaxe	Description
le total pour tous *<ent>***,** *<attr>* **le total pour toutes** *<ent>***,** *<attr>* **la quantité totale pour tous** *<ent>***,** *<attr>* **la quantité totale pour toutes** *<ent>***,** *<attr>* *<attr>* **totalisé pour tous** *<ent>* *<attr>* **totalisé pour toutes** *<ent>*	
InstancesSommeSi(*<relationship>*, *<number-attr>*, *<condition>*) **le total pour tous** *<ent>***,** *<attr>* **dans le cas où** *<condition>* **le total pour toutes** *<ent>***,** *<attr>* **dans le cas où** *<condition>* **la quantité totale pour tous** *<ent>***,** *<attr>* **dans le cas où** *<condition>* **la quantité totale pour toutes** *<ent>***,** *<attr>* **dans le cas où** *<condition>* *<attr>* **totalisé pour tous** *<ent>* **dans le cas où** *<condition>* *<attr>* **totalisé pour toutes** *<ent>* **dans le cas où** *<condition>*	Obtient la somme de toutes les instances d'une variable de niveau entité ***entity-level*** pour laquelle il est vrai pour l'entité ***entity*** qu'un attribut ***attribute*** booléen de niveau entité ***entity-level*** est vrai.
InstanceValeurSi(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceValeurSi**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanceValeurSi**(*<relationship>*, *<date-attr>*, *<con-*	Obtient une valeur d'une instance ***entity*** unique, identifiée à partir des instances ***entity*** cible d'une ***relationship*** par une condition. • Si la condition identifie une instance de l'instance ***entity*** cible unique, la valeur est la valeur calculée par rapport à cette instance ***entity***. • Si plusieurs instances cible satisfont la condition, ***uncertain*** est renvoyé. • Si aucune instance cible ne satisfait la condition et que la ***relationship*** est connue, la valeur est ***uncertain***.

Syntaxe	Description
dition>) **InstanceValeurSi**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceValeurSi**(*<relationship>*, *<time-attr>*, *<condition>*)	
InstanceEgale (*<instance1>*, *<instance2>*)	Détermine si deux instances d'une entité ***entity*** sont la même instance.
InstanceDifférente (*<instance1>*, *<instance2>*)	Détermine si deux instances d'une entité ***entity*** ne sont pas la même instance.
InférerInstance(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** (**existent** | **existe**)	Utilisé comme conclusion pour inférer qu'une instance de ***entity*** existe et est membre d'une relation ***relationship***.

Fonctions de raisonnement temporel(English)

Syntaxe	Description
IntervalleNombreDistinct(*<start-date>*, *<end-date>*, *<variable>*) **IntervalleNombreDistinct**(*<start-date>*, *<end-date>*, *<condition>*)	Compte le nombre de valeurs distinctes connues pour la variable, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue).
IntervalleNombreDistinctSi (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Compte le nombre de valeurs distinctes connues pour la variable, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue), uniquement lorsqu'un filtre booléen est vrai.
IntervalleSommeQuotidienne (*<start-date>*, *<end-date>*, *<number-attr>*)	Calcule la somme d'une variable devise ou numérique, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue). L'attribut ***attribute*** est supposé être une quantité par jour.
IntervalleSommeQuotidienneSsi (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Calcule la somme de toutes les valeurs journalières pour une variable devise ou numérique, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue), uniquement lorsqu'une condition est vraie.
IntervalleMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalleMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalleMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*)	Sélectionne la valeur maximum d'une variable dans l'intervalle compris entre une date ***date*** de début (incluse) et une date ***date*** de fin (exclue).

Syntaxe	Description
IntervalleMaximum(*<start-date>*, *<end-date>*, *<time-attr>*)	
IntervalleMaximumSi(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalleMaximumSi**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalleMaximumSi**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalleMaximumSi**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Sélectionne la valeur maximum d'une variable dans l'intervalle compris entre une date ***date*** de début (incluse) et une date ***date*** de fin (exclue), uniquement lorsqu'une condition est vraie.
IntervalleMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalleMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalleMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalleMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Sélectionne la valeur minimum d'une variable dans l'intervalle compris entre une date ***date*** de début (incluse) et une date ***date*** de fin (exclue).
IntervalleMinimumSi(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalleMinimumSi**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalleMinimumSi**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalleMinimumSi**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Sélectionne la valeur minimum d'une variable dans l'intervalle compris entre une date ***date*** de début (incluse) et une date ***date*** de fin (exclue), uniquement lorsqu'une condition est vraie.
IntervalleMoyennePondérée (*<start-date>*, *<end-date>*, *<number-attribute>*)	Calcule la valeur moyenne d'une variable devise ou numérique, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue), pondérée par le laps de temps durant lequel chaque valeur s'applique.
IntervalleMoyennePondéréeSi (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Calcule la valeur moyenne d'une variable devise ou numérique, dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue), uniquement lorsqu'une condition booléenne est vraie (pondérée par le laps de temps durant lequel chaque valeur s'applique et lorsque le filtre est vrai).

Syntaxe	Description
IntervalleToujours(*<start-date>*, *<end-date>*, *<condition>*)	Renvoie Vrai si et seulement si une condition booléenne est toujours vraie à tout moment dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue).
TemporelJoursAuMoins(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Renvoie Vrai si et seulement si une condition booléenne est vraie pendant au moins le nombre de jours spécifié (pas nécessairement consécutifs) dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue).
IntervalleJoursConsécutifs (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Renvoie Vrai si et seulement si une condition booléenne est vraie pendant au moins un nombre de jours consécutifs donné dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue).
IntervalleParfois(*<start-date>*, *<end-date>*, *<condition>*)	Renvoie Vrai si et seulement si une condition booléenne est toujours vraie dans l'intervalle compris entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue).
ValeurA(*<date>*, *<value>*)	Renvoie la valeur de l'attribut ***attribute*** donné à la date ***date*** spécifiée.
QuandDernier(*<date>*, *<condition>*)	Renvoie la dernière date ***date*** à laquelle une condition booléenne a été vraie, en remontant dans le temps à partir d'une date ***date*** (incluse) spécifiée.
QuandSuivant(*<date>*, *<condition>*)	Renvoie la prochaine date ***date*** à laquelle une condition booléenne sera vraie, à partir d'une date ***date*** (incluse) spécifiée.
Dernier()	Renvoie une valeur de date ***date*** équivalente à la dernière date ***date*** possible - c'est-à-dire une date ***date*** garantie comme étant postérieure à toute autre date ***date*** pouvant être prise par un attribut date ***date attribute*** ou résulter de l'évaluation d'une expression.
Premier()	Renvoie une valeur de date ***date*** équivalente à la première date ***date*** possible - c'est-à-dire une date ***date*** garantie comme étant antérieure à toute autre date ***date*** pouvant être prise par un attribut date ***date attribute*** ou résulter de l'évaluation d'une expression.
TemporelJoursEcoulés(*<date>*, *<end-date>*)	Renvoie une variable numérique qui varie chaque jour et correspond au nombre de jours complets depuis la date ***date***.
TemporelSemainesEcoulées (*<date>*, *<end-date>*)	Renvoie une variable numérique qui varie chaque semaine et correspond au nombre de semaines complètes depuis la date ***date***.
TemporelMoisEcoulés(*<date>*, *<end-date>*)	Renvoie une variable numérique qui varie chaque mois et correspond au nombre de mois complets depuis la date ***date***. Note : Si la date ***date*** indiquée est postérieure au 28e jour du mois et que le mois suivant a moins de jours que le mois indiqué, le point de modification pour le mois anniversaire est créé sur le dernier jour de ce mois. Par exemple, si la date ***date*** indiquée est le 28, 29, 30 ou le 31 janvier 2007, le premier point de modification est le 28 février 2007.
TemporelAnnéesEcoulées (*<date>*, *<end-date>*)	Renvoie une variable numérique qui varie chaque année et correspond au nombre d'années complètes depuis la date ***date***.

Syntaxe	Description
TemporelJoursToujours(*<days>*, *<condition>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai si et seulement si une condition booléenne est vraie pour un nombre de jours précédents donné, sans inclure le jour actuel.
TemporelJoursConsécutifs (*<minDays>*, *<days>*, *<condition>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai si et seulement si une condition booléenne est vraie pour au moins un nombre minimum de jours consécutifs à tout moment dans le nombre défini de jours précédents, sans inclure le jour actuel.
TemporelJoursParfois(*<days>*, *<condition>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai si et seulement si une condition booléenne est toujours vraie pour un nombre de jours précédents donné, sans inclure le jour actuel.
TemporelAprès(*<date>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai après une date ***date*** et faux à cette date et avant.
TemporelAvant(*<date>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai avant une date ***date*** et faux à cette date et après.
TemporelEnCours(*<date>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai à une date ***date***, et faux avant et après cette date.
TemporelEnCoursOuAprès (*<date>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai à une date ***date*** ou après, et faux avant cette date.
TemporelEnCoursOuAvant (*<date>*)	Renvoie un attribut ***attribute*** booléen qui varie dans le temps et est vrai à une date ***date*** ou avant, et faux après cette date.
TemporelDateDepuisDébut(*<relationship>*, *<date>*, *<value>*)	Renvoie un attribut ***attribute*** temporel unique (au niveau entité ***entity*** source) à partir d'une relation ***relationship*** et d'un attribut ***attribute*** de valeur sur les entités, les valeurs prenant effet à compter d'un attribut date ***date attribute*** de début.
TemporelDateDepuisFin(*<relationship>*, *<date>*, *<value>*)	Renvoie un attribut ***attribute*** temporel unique (au niveau entité ***entity*** source) à partir d'une relation ***relationship*** et d'un attribut ***attribute*** de valeur sur les entités, les valeurs prenant effet jusqu'à un attribut date ***date attribute*** de fin.
TemporelAPartirPlage(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Renvoie un attribut ***attribute*** temporel unique (au niveau entité ***entity*** source) à partir d'une relation ***relationship*** et d'un attribut ***attribute*** de valeur sur les entités, les valeurs prenant effet entre un attribut date ***date attribute*** de début (incluse) et un attribut date ***date attribute*** de fin (exclue). La valeur est "incertain" ***uncertain*** en cas d'expiration avant la date ***date*** de début suivante.
TemporelEstJoursSemaine (*<startdate>*, *<enddate>*)	Renvoie vrai pour les dates correspondant à des jours de semaine et faux pour les dates correspondant à des jours de week-end entre la date ***date*** de début spécifiée (incluse) et la date ***date*** de fin (exclue). Renvoie ***uncertain*** en dehors de la plage de dates ***date***.
TemporelUneFoisParMois (*<startdate>*, *<enddate>*, *<day-*	Renvoie Vrai si le jour est égal au paramètre du jour du mois et Faux pour tous les

Syntaxe	Description
ofmonth>)	autres jours du mois entre la date ***date*** de début (incluse) et la date ***date*** de fin (exclue). Renvoie "incertain" ***uncertain*** si en dehors de la fourchette de valeurs de date ***date***. Lorsque le jour du mois dépasse le nombre de jours du mois actuel, la valeur est Vrai pour le dernier jour de ce mois, afin que la fonction renvoie une valeur Vrai un jour par mois.

Fonctions de l'événement de validation(English)

Syntaxe	Description
Erreur(*<text>*)	Un événement d'erreur sert à transmettre un message à l'utilisateur et à l'empêcher de poursuivre une investigation tant que la condition qui a été à l'origine de l'erreur reste vraie.
Avertissement (*<text>*)	Un événement d'avertissement sert à transmettre un message à l'utilisateur ; il ne l'empêche pas de poursuivre bien que la condition à l'origine de l'avertissement persiste.

Fonctions en phase d'abandon(English)

Syntaxe	Description
FonctionAppelPersonnalisée (*<A>*, *<B>*)	Renvoie le résultat d'un appel externe à une bibliothèque de code. La bibliothèque de code doit être indiquée au moteur de déterminations pour assurer le succès de la fonction personnalisée.

Logische Connectors(English)

Syntax	Beschreibung
wenn **falls**	Optionaler Ausdruck, der am Ende einer Konklusionszeile angezeigt werden kann, auf die eine Prüfung folgt
und	Logische Verknüpfung zwischen zwei ***attributes***
oder	Logische Disjunktion zwischen zwei ***attributes***
entweder **eine von** **einer von** **beliebige** **mindestens eine der folgenden Bedingungen ist wahr** **eine der folgenden Bedingungen ist erfüllt**	Gruppierungselement, das mit Disjunktionen verwendet wird, bei denen mindestens zwei ***attributes*** gruppiert werden müssen
beide	Gruppierungselement, das mit Verknüpfungen verwendet wird, bei denen mindestens

Syntax	Beschreibung
alle **alle Folgenden sind wahr** **alle Folgenden sind erfüllt**	zwei ***attributes*** gruppiert werden müssen
anderenfalls	Ausdruck, der am Ende einer Tabellenregel angezeigt wird und die Klausel "Anderenfalls" angibt
ist	Ausdruck, der in einem Legendeneintrag zwischen dem abgekürzten Begriff und dem vollständigen ***attribute text*** verwendet wird

Logische Funktionen(English)

Syntax	Beschreibung
es ist nicht wahr, dass *<expr>*	Operator, der True zurückgibt, wenn ***attribute*** einen Wert hat, der falsch ist
<var> **ist sicher** **es ist sicher,** [**ungeachtet dessen**] **ob** *<expr>*	Operator, der True zurückgibt, wenn ***attribute*** einen Wert aufweist, der nicht ***uncertain*** ist
<var> **ist unsicher** *<var>***ist nicht sicher** **es ist unsicher,** [**ungeachtet dessen**] **ob** *<expr>* **es ist unsicher, dass** *<expr>* **es ist nicht sicher, dass** *<expr>* **unsicher**	Operator, der True zurückgibt, wenn der Wert für ***attribute uncertain*** ist
<var> **ist bekannt** **es ist bekannt,** [**ungeachtet dessen**] **ob** *<expr>*	Operator, der True zurückgibt, wenn ***attribute*** einen beliebigen Wert aufweist
<var> **ist unbekannt** **es ist unbekannt,** [**ungeachtet dessen**] **ob** *<expr>* **unbekannt**	Operator, der True zurückgibt, wenn ***attribute*** keinen Wert aufweist

Logische Konstanten(English)

Syntax	Beschreibung
wahr	Konstanter True-Wert, der für Tabellenregeln verwendet wird.
falsch	Konstanter False-Wert, der für Tabellenregeln verwendet wird.
unbestimmt	Konstanter Wert ***uncertain***, der für Tabellenregeln verwendet wird.

Vergleichsoperatoren(English)

Syntax	Beschreibung
*<lhs>***<***<rhs>* *<lhs>* **ist weniger als** *<rhs>* *<lhs>* **ist früher als** *<rhs>*	Kleiner als Hinweis: Es gibt keine natürliche Sprachform, wenn dieser Operator mit numerischen oder Währungswerten verwendet wird.
<lhs> **>** *<rhs>* *<lhs>* **ist größer als** *<rhs>* *<lhs>* **ist später als** *<rhs>*	Größer als Hinweis: Es gibt keine natürliche Sprachform, wenn dieser Operator mit numerischen oder Währungswerten verwendet wird.
*<lhs>***<=***<rhs>* *<lhs>* **ist weniger als oder gleicht** *<rhs>* *<lhs>* **ist früher als oder gleicht** *<rhs>*	Kleiner/gleich
<lhs> **>=** *<rhs>* *<lhs>* **ist größer als oder gleicht** *<rhs>* *<lhs>* **ist später als oder gleicht** *<rhs>*	Größer/gleich
*<lhs>***=***<rhs>* *<lhs>* **ist** *<rhs>* **gleich** *<lhs>* **gleicht** *<rhs>*	Gleich
<lhs> **ist** *<rhs>* **nicht gleich** *<lhs>* **gleicht** *<rhs>* **nicht** *<lhs>* **<>** *<rhs>* *<lhs>* **!=** *<rhs>*	Ungleich

Numerische Funktionen(English)

Syntax	Beschreibung
Zahl(*<numText>*)	Konvertiert die angegebene Zeichenfolge in einen Zahlenwert
<x> **+** *<y>*	Addition
<x> **-** *<y>*	Subtraktion
<x> * *<y>*	Multiplikation
<x> **/** *<y>*	Division
<x> \ *<y>*	Ganzzahlendivision
<x> **modulo** *<y>*	Rest nach Ganzzahlendivision

Syntax	Beschreibung
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) (**der** | **die** | **das**) **größere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **höhere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **spätere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **letzte von** *<val1>* **und** *<val2>*	Gibt den größeren von zwei Werten zurück
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) (**der** | **die** | **das**) **geringere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **kleinere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **frühere von** *<val1>* **und** *<val2>* (**der** | **die** | **das**) **erste von** *<val1>* **und** *<val2>*	Gibt den kleineren von zwei Werten zurück
Xy(*<x>*, *<y>*)	x hoch y
Ex(*<x>*)	Konstante e hoch x
Absolut(*<x>*)	Absoluter Wert von x
Natürl. Logarithmus(*<x>*)	Natürlicher Logarithmus von x
Logarithmus(*<x>*)	Logarithmus von x zur Basis 10
Wurzel(*<x>*) **die Quadartwurzel von** *<val>*	Quadratwurzel von x
Runden(*<x>*, *<n>*) *<val>* **abgerundet zu** *<num_places>*(**Dezimalstelle** | **Dezimalstellen**)	Rundet x auf n Dezimalstellen
Gekürzt(*<x>*, *<n>*)	x gekürzt auf n Dezimalstellen
Sinus(*<x>*)	Sinus von x
Kosinus(*<x>*)	Kosinus von x
Tangens(*<x>*)	Tangens von x
Arkussinus(*<x>*)	Arkussinus von x
Arkuskosinus(*<x>*)	Arkuskosinus von x
Arkustangens(*<x>*)	Arkustangens von x

Datumsfunktionen(English)

Syntax	Beschreibung
AktuellesDatum()	Gibt das aktuelle ***date*** zu Beginn der Session zurück.

Syntax	Beschreibung
das gegenwärtige Datum	
Datum(*<text>*)	Konvertiert die angegebene Zeichenfolge in einen ***date***-Wert
DatumFestlegen(*<year>*, *<month>*, *<day>*)	Gibt ein ***date*** zurück, das aus den angegebenen Werten für Jahr, Monat und Tag gebildet wird.
TagExtrahieren(*<date/datetime>*)	Gibt die Tageskomponente eines Attributs des Typs ***date/datetime attribute*** zurück.
MonatExtrahieren(*<date/datetime>*)	Gibt die Monatskomponente eines Attributs des Typs ***date/datetime attribute*** zurück.
JahrExtrahieren(*<date/datetime>*)	Gibt die Jahreskomponente eines Attributs des Typs ***date/datetime attribute*** zurück.
NächsterWochentag(*<date/datetime>*, *<day>*) **der** [**nächste**] **Montag am oder nach** *<from-date>* **der** [**nächste**] **Dienstag am oder nach** *<from-date>* **der** [**nächste**] **Mittwoch am oder nach** *<from-date>* **der** [**nächste**] **Donnerstag am oder nach** *<from-date>* **der** [**nächste**] **Freitag am oder nach** *<from-date>* **der** [**nächste**] **Samstag am oder nach** *<from-date>* **der** [**nächste**] **Sonntag am oder nach** *<from-date>*	Gibt das ***date*** des nächsten Wochentages an oder vor einem ***date*** zurück (abhängig von der verwendeten Syntax).
NächstesDatum(*<date>*, *<day>*, *<month>*)	Gibt die nächste Instanz des angegebenen Tages und Monats nach einem ***date*** zurück.
TageHinzufügen(*<date/datetime>*, *<num_days>*) **das Datum** *<num_days>*(**Tag** | **Tage**) **nach** *<date>* **die Zeit** *<num_days>*(**Tag** | **Tage**) **nach** *<date>*	Fügt einem ***date*** eine Anzahl von Tagen hinzu bzw. zieht sie davon ab. Wenn die syntaktische Kurzform verwendet wird, muss die Zahl entweder eine positive Ganzzahl sein, damit die Tage dem eingegebenen ***date*** hinzugefügt werden können, oder eine negative Zahl, um die Tage vom eingegebenen ***date*** abzuziehen.
WochenHinzufügen(*<date/datetime>*, *<num_weeks>*) **das Datum** *<num_weeks>* (**Woche** | **Wochen**) **nach**	Fügt einem ***date*** eine Anzahl von Wochen hinzu. Wenn die syntaktische Kurzform verwendet wird, muss die Zahl eine positive Ganzzahl sein, damit die Wochen dem eingegebenen ***date*** hinzugefügt werden können.

Syntax	Beschreibung
<date> **die Zeit** *<num_weeks>*(**Woche** | **Wochen**) **nach** *<date>*	
MonateHinzufügen(*<date/datetime>*, *<num_months>*) **das Datum** *<num_months>* (**Monat** | **Monate**) **nach** *<date>* **die Zeit** *<num_months>*(**Monat** | **Monate**) **nach** *<date>*	Fügt einem ***date*** eine Anzahl von Monaten hinzu. Wenn die syntaktische Kurzform verwendet wird, muss die Zahl eine positive Ganzzahl sein, damit die Monate dem eingegebenen ***date*** hinzugefügt werden können.
JahreHinzufügen(*<date/datetime>*, *<num_years>*) **das Datum** *<num_years>*(**Jahr** | **Jahre**) **nach** *<date>* **die Zeit** *<num_years>*(**Jahr** | **Jahre**) **nach** *<date>*	Fügt einem ***date*** eine Anzahl von Jahren hinzu. Wenn die syntaktische Kurzform verwendet wird, muss die Zahl eine positive Ganzzahl sein, damit die Jahre dem eingegebenen ***date*** hinzugefügt werden können.
AnzahlWochentage(*<date1>*, *<date2>*) **die Anzahl der** [**ganzen**] **Werktage von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Werktage zwischen** *<date1>* **und** *<date2>*	Zählt die Anzahl der Wochentage zwischen ***date***1 und ***date***2, d.h. die Anzahl der Tage zwischen Montag und Freitag. Hinweis: Das frühere ***date*** ist einschließlich, und das spätere ***date*** ist ausschließlich.
JahrBeginn(*<date/datetime>*) **der erste Tag im Jahr, in dem** *<from-date>* **fällt**	Gibt das erste ***date*** in dem Jahr zurück, in das ein ***date*** fällt.
JahrEnde(*<date/datetime>*) **der letzte Tag im Jahr, in dem** *<from-date>* **fällt**	Gibt das letzte ***date*** in dem Jahr zurück, in das ein ***date*** fällt.
TagDifferenz (*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**]**Tage von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**]**Tage zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der ganzen Tage zwischen ***date/datetime1*** und ***date/datetime2*** zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
TagDifferenzEinschließlich (*<date/datetime1>*, *<date/d-*	Gibt die Anzahl der ganzen Tage zwischen ***date/datetime1*** und ***date/datetime2*** (einschließlich) zurück. Diese Berechnung schließt beide Endpunkte ein. Wenn die Datumswerte identisch sind, ist das Ergebnis 1. Die Reihenfolge der beiden Datumswerte wirkt

Syntax	Beschreibung
atetime2>) **die Anzahl der** [**ganzen**] **Tage (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Tage (inklusiv) zwischen** *<date1>* **und** *<date2>*	sich nicht auf das Ergebnis aus.
TagDifferenzOhne(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Tage (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Tage (exklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der ganzen Tage zwischen ***date/datetime1*** und ***date/datetime2*** (ausschließlich) zurück. Diese Berechnung schließt beide Endpunkte aus. Wenn die Datumswerte identisch sind, ist das Ergebnis 0. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
WocheDifferenz(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Wochen von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Wochen zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der ganzen verstrichenen Wochen zwischen ***date/datetime1*** und ***date/datetime2*** zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
WocheDifferenzEinschließlich(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Wochen (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Wochen (inklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der ganzen verstrichenen Wochen zwischen ***date/datetime1*** und ***date/datetime2*** (einschließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
WocheDifferenzOhne(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Wochen (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Wochen (exklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der ganzen verstrichenen Wochen zwischen ***date/datetime1*** und ***date/datetime2*** (ausschließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
MonatDifferenz(*<date/d-*	Gibt die Anzahl der verstrichenen ganzen Monate zwischen ***date/datetime1*** und ***date/d-***

Syntax	Beschreibung
atetime1>, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Monate von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Monate zwischen** *<date1>* **und** *<date2>*	***atetime2*** zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
MonatDifferenzEinschließlich (*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Monate (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Monate (inklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der verstrichenen ganzen Monate zwischen ***date/datetime1*** und ***date/datetime2*** (einschließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
MonatDifferenzOhne(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Monate (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Monate (exklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der verstrichenen ganzen Monate zwischen ***date/datetime1*** und ***date/datetime2*** (ausschließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
JahrDifferenz(*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Jahre von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Jahre zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Jahre zwischen ***date/datetime1*** und ***date/datetime2*** zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
JahrDifferenzEinschließlich (*<date/datetime1>*, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Jahre (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Jahre (inklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Jahre zwischen ***date/datetime1*** und ***date/datetime2*** (einschließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.
JahrDifferenzOhne(*<date/d-*	Gibt die Anzahl der Jahre zwischen ***date/datetime1*** und ***date/datetime2*** (aus-

Syntax	Beschreibung
atetime1>, *<date/datetime2>*) **die Anzahl der** [**ganzen**] **Jahre (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der** [**ganzen**] **Jahre (exklusiv) zwischen** *<date1>* **und** *<date2>*	schließlich) zurück. Die Reihenfolge der beiden Datumswerte wirkt sich nicht auf das Ergebnis aus.

Uhrzeitfunktionen(English)

Syntax	Beschreibung
Uhrzeit(*<text>*)	Konvertiert die angegebene Zeichenfolge in eine Uhrzeit
SekundeExtrahieren(*<time/datetime>*)	Gibt die Sekundenkomponente eines Attributs des Typs ***timeofday/datetime attribute*** zurück.
MinuteExtrahieren(*<time/datetime>*)	Gibt die Minutenkomponente eines Attributs des Typs ***timeofday/datetime attribute*** zurück.
StundeExtrahieren(*<time/datetime>*)	Gibt die Stundenkomponente eines Attributs des Typs ***timeofday/datetime attribute*** zurück.

Datums- und Uhrzeitfunktionen(English)

Syntax	Beschreibung
AktuellesDatumUhrzeit()	Gibt das aktuelle ***date*** und die aktuelle Uhrzeit zu Beginn der Session zurück.
DatumUhrzeit(*<text>*)	Konvertiert die angegebene Zeichenfolge in einen ***datetime***-Wert
DatumUhrzeitVerknüpfen (*<date>*, *<time>*) *<date>* **um** *<time-of-day>* *<time-of-day>* **am** *<date>*	Legt die Uhrzeit an einem ***date*** fest, indem ***date*** und Uhrzeit miteinander verknüpft werden.
SekundeDifferenz(*<datetime1>*, *<datetime2>*) **SekundeDifferenz** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Sekunden von** *<date1>* **bis zu** *<date2>* **die Anzahl der Sekunden zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Sekunden zwischen ***datetime1*** und ***datetime2*** zurück.
SekundeDifferenzEinschließlich	Gibt die Anzahl der Sekunden zwischen ***datetime1*** und ***datetime2*** (einschließlich)

Syntax	Beschreibung
(*<datetime1>*, *<datetime2>*) **SekundeDifferenzEinschließlich** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Sekunden (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der Sekunden (inklusiv) zwischen** *<date1>* **und** *<date2>*	zurück.
SekundeDifferenzOhne(*<datetime1>*, *<datetime2>*) **SekundeDifferenzOhne** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Sekunden (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der Sekunden (exklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Sekunden zwischen ***datetime1*** und ***datetime2*** (ausschließlich) zurück.
MinuteDifferenz(*<datetime1>*, *<datetime2>*) **MinuteDifferenz**(*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Minuten von** *<date1>* **bis zu** *<date2>* **die Anzahl der Minuten zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Minuten zwischen ***datetime1*** und ***datetime2*** zurück.
MinuteDifferenzEinschließlich (*<datetime1>*, *<datetime2>*) **MinuteDifferenzEinschließlich** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Minuten (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der Minuten (inklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Minuten zwischen ***datetime1*** und ***datetime2*** (einschließlich) zurück.
MinuteDifferenzOhne(*<datetime1>*, *<datetime2>*) **MinuteDifferenzOhne** (*<timeOfDay1>*, *<timeOfDay2>*)	Gibt die Anzahl der Minuten zwischen ***datetime1*** und ***datetime2*** (ausschließlich) zurück.

Syntax	Beschreibung
die Anzahl der Minuten (exklusiv) von *<date1>* **bis zu** *<date2>* **die Anzahl der Minuten (exklusiv) zwischen** *<date1>* **und** *<date2>*	
StundeDifferenz(*<datetime1>*, *<datetime2>*) **StundeDifferenz**(*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Stunden von** *<date1>* **bis zu** *<date2>* **die Anzahl der Stunden zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Stunden zwischen ***datetime1*** und ***datetime2*** zurück.
StundeDifferenzEinschließlich (*<datetime1>*, *<datetime2>*) **StundeDifferenzEinschließlich** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Stunden (inklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der Stunden (inklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Stunden zwischen ***datetime1*** und ***datetime2*** (einschließlich) zurück.
StundeDifferenzOhne(*<datetime1>*, *<datetime2>*) **StundeDifferenzOhne** (*<timeOfDay1>*, *<timeOfDay2>*) **die Anzahl der Stunden (exklusiv) von** *<date1>* **bis zu** *<date2>* **die Anzahl der Stunden (exklusiv) zwischen** *<date1>* **und** *<date2>*	Gibt die Anzahl der Stunden zwischen ***datetime1*** und ***datetime2*** (ausschließlich) zurück.
DatumExtrahieren(*<datetime>*)	Extrahiert das ***date*** aus einem Attribut des Typs ***datetime attribute***.
TageszeitExtrahieren(*<datetime>*)	Extrahiert die Tageszeit aus einem Attribut des Typs ***datetime attribute***. Kann verwendet werden, um den Wert einer ***timeofday attribute*** auf die Uhrzeit der Regelausführung zu setzen, indem die Uhrzeit aus dem aktuellen ***date*** und der aktuellen Uhrzeit extrahiert wird.
StundenHinzufügen	Fügt einer Uhrzeit an einem ***date*** eine Anzahl von Stunden hinzu.

Syntax	Beschreibung
(*<datetime>*, *<num_hours>*) **StundenHinzufügen** (*<timeOfDay>*, *<num_hours>*) **die Zeit** *<num_hours>*(**Stunde** | **Stunden**) **nach** *<date>* **die Tageszeit** *<num_hours>* (**Stunde** | **Stunden**) **nach** *<time-of-day>*	
MinutenHinzufügen (*<datetime>*, *<num_minutes>*) **MinutenHinzufügen** (*<timeOfDay>*, *<num_minutes>*) **die Zeit** *<num_minutes>*(**Minute** | **Minuten**) **nach** *<date>* **die Tageszeit** *<num_minutes>* (**Minute** | **Minuten**) **nach** *<time-of-day>*	Fügt einer Uhrzeit an einem ***date*** eine Anzahl von Minuten hinzu.
SekundenHinzufügen(*<datetime>*, *<num_seconds>*) **SekundenHinzufügen** (*<timeOfDay>*, *<num_seconds>*) **die Zeit** *<num_seconds>* (**Sekunde** | **Sekunden**) **nach** *<date>* **die Tageszeit** *<num_seconds>* (**Sekunde** | **Sekunden**) **nach** *<time-of-day>*	Fügt einer Uhrzeit an einem ***date*** eine Anzahl von Sekunden hinzu.

Textfunktionen(English)

Syntax	Beschreibung
<text1> **&** *<text2>*	Kombiniert ***text1*** mit ***text2*** usw. zu einem einzelnen ***text***-Wert. Hinweis: Sie können Variablen jedes beliebigen Typs verwenden. Die Werte werden mit dem in der Regel-Session installierten Formatierer formatiert.
die Verkettung von *<text1>* **&** *<text2>*	Kombiniert ***text1*** mit ***text2*** usw. zu einem einzelnen ***text***-Wert. Hinweis: Sie können Variablen jedes beliebigen Typs verwenden. Die Werte werden mit dem in der Regel-Session installierten Formatierer formatiert.
Enthält(*<text>*, *<substring>*) *<text>* **enthält** *<sub-*	Gibt einen booleschen Wert zurück, der angibt, ob ein bestimmter ***text***-Wert die angegebene ***text***-Teilzeichenfolge enthält. Beim Vergleich von ***text*** ist die Groß-/Kleinschreibung irrelevant.

Syntax	Beschreibung
string>	
EndetMit(*<text>*, *<substring>*) *<text>* **mit** *<substring>* **endet**	Gibt einen booleschen Wert zurück, der angibt, ob ein bestimmter ***text***-Wert mit der angegebenen ***text***-Teilzeichenfolge endet. Beim Vergleich von ***text*** ist die Groß-/Kleinschreibung irrelevant.
IstZahl(*<text>*) *<text>* **ist eine Zahl**	Gibt einen booleschen Wert zurück, der angibt, ob ein bestimmter ***text***-Wert eine gültige Zahl darstellt.
Länge(*<text>*)	Gibt die Zeichenlänge des angegebenen ***text***-Werts zurück.
BeginntMit(*<text>*, *<substring>*) *<text>* **mit** *<substring>* **beginnt**	Gibt einen booleschen Wert zurück, der angibt, ob ein bestimmter ***text***-Wert mit der angegebenen ***text***-Teilzeichenfolge beginnt. Beim Vergleich von ***text*** ist die Groß-/Kleinschreibung irrelevant.
Teilzeichenfolge (*<text>*, *<offset>*, *<length>*)	Gibt die Teilzeichenfolge von ***text*** zurück, die beim angegebenen Offset beginnt, d.h. nach der angegebenen Zeichenlänge. Bei Erreichen des Endes der Zeichenfolge werden weniger Zeichen zurückgegeben.
Text(*<number>*) **Text**(*<date>*) **Text**(*<datetime>*) **Text**(*<timeOfDay>*)	Konvertiert die angegebene Zahl oder das angegebene ***date attribute*** in einen ***text***-Wert.

Entity- und Beziehungsfunktionen(English)

Syntax	Beschreibung
Für(*<relationship>*, *<Exp>*) **im Fall von** *<ent>*, *<attr>* *<val>*, **im Fall von** *<ent>*	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des Typs "1:n", "n:1" oder "m:m" verwendet, wenn nur eine Bedingung vorhanden ist.
FürGeltungsbereich(*<relationship>*, *<alias>*) **FürGeltungsbereich**(*<relationship>*) **im Fall von** *<ent>* **im Fall von** *<ent>* **(**<alias>**)**	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des Typs "1:n", "n:1" oder "m:m" verwendet, wenn mindestens eine Bedingung vorhanden ist.
FürAlle(*<relationship>*, *<Exp>*) **für jedes** *<ent>*, *<attr>*	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des Typs "1:n" oder "m:m" verwendet, wenn Sie festlegen müssen, ob alle Mitglieder der Ziel-***entity***-Gruppe die Regel erfüllen müssen. Diese Form wird verwendet, wenn die Regel nur eine Bedingung aufweist.
FürAlleGeltungsbereiche	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des

Syntax	Beschreibung
(*<relationship>*) **FürAlleGeltungsbereiche** (*<relationship>*, *<alias>*) **für** *<ent>* **für alle** *<ent>* [**für**] **jedes** *<ent>* **für** *<ent>* **(**<alias>**)** **für alle** *<ent>* **(**<alias>**)** [**für**] **jedes** *<ent>* **(**<alias>**)**	Typs "1:n" oder "m:m" verwendet, wenn Sie festlegen müssen, ob alle Mitglieder der Ziel-***entity***-Gruppe die Regel erfüllen müssen. Diese Form wird verwendet, wenn die Regel mindestens eine Bedingung aufweist.
Vorhanden(*<relationship>*, *<Exp>*) **für mindestens eins von** *<ent>***,** *<attr>*	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des Typs "1:n" oder "m:m" verwendet, wenn Sie festlegen müssen, ob Mitglieder der Ziel-***entity***-Gruppe die Regel erfüllen müssen. Diese Form wird verwendet, wenn die Regel nur eine Bedingung aufweist.
VorhandenGeltungsbereich (*<relationship>*) **VorhandenGeltungsbereich** (*<relationship>*, *<alias>*) **mindestens eins von** *<ent>* **mindestens eins von** *<ent>* **(**<alias>**)**	Wird für den Verweis von einer ***entity*** auf eine andere ***entity*** in einer ***relationship*** des Typs "1:n" oder "m:m" verwendet, wenn Sie festlegen müssen, ob Mitglieder der Ziel-***entity***-Gruppe die Regel erfüllen müssen. Diese Form wird verwendet, wenn die Regel mindestens eine Bedingung aufweist.
IstMitgliedVon(*<target>*, *<relationship>*) **IstMitgliedVon**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **ist ein Mitglied** *<relationship>* <ent-target> **(**<alias>**) ist ein Mitglied** *<relationship>*	Wird in einer Konklusion verwendet, um abzuleiten, dass eine ***entity***-Instanz Mitglied einer ***relationship*** ist. Wird als Bedingung verwendet, um zu testen, ob eine ***entity***-Instanz Ziel einer ***relationship*** ist, deren Quelle eine zweite ***entity***-Instanz ist.
IstKeinMitgliedVon(*<target>*, *<relationship>*) <ent-target> **ist nicht ein Mitglied der** *<relationship>*	Wird als Bedingung für den Test verwendet, ob eine ***entity***-Instanz kein Ziel einer ***relationship*** ist, deren Quelle eine zweite ***entity***-Instanz ist.
AnzahlInstanzen(*<relationship>*) **die Anzahl** *<ent>*	Zählt die Anzahl der vorhandenen Instanzen für eine ***entity***.
AnzahlInstanzenFalls(*<relationship>*, *<Exp>*) **die Anzahl** *<ent>***, für den es der Fall ist, dass** *<condition>*	Zählt die Anzahl der für eine ***entity*** vorhandenen Instanzen, für die ein bestimmtes ***entity-level attribute*** einen bestimmten Wert aufweist.

Syntax	Beschreibung
InstanzMaximum(*<relationship>*, *<number-attr>*) **InstanzMaximum**(*<relationship>*, *<date-attr>*) **InstanzMaximum**(*<relationship>*, *<datetime-attr>*) **InstanzMaximum**(*<relationship>*, *<time-attr>*) (**der** | **die** | **das**) **größte** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **höchste** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **späteste** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **letzte** *<attr>* **für alle** *<ent>* *<attr>* **das für alle** *<ent>* **am größten ist** *<attr>* **das für alle** *<ent>* **am höchsten ist** *<attr>* **das für alle** *<ent>* **am spätesten ist** *<attr>* **das für alle** *<ent>* **am letzten ist**	Ermittelt den höchsten/aktuellen Wert einer ***entity-level***-Variable für alle Instanzen der ***entity***.
InstanzMaximumFalls(*<relationship>*, *<number-attr>*, *<condition>*) **InstanzMaximumFalls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanzMaximumFalls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanzMaximumFalls**(*<relationship>*, *<time-attr>*, *<condition>*) (**der** | **die** | **das**) **größte** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* (**der** | **die** | **das**) **höchste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>*	Ermittelt den höchsten/aktuellen Wert einer ***entity-level***-Variable für alle Instanzen der ***entity***, für die ein bestimmtes ***entity-level attribute*** einen bestimmten Wert aufweist.

Syntax	**Beschreibung**
(**der** | **die** | **das**) **späteste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* (**der** | **die** | **das**) **letzte** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am größten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am höchsten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am spätesten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am letzten ist und für den es der Fall ist, dass** *<ent-test>*	
InstanzMinimum(*<relationship>*, *<number-attr>*) **InstanzMinimum**(*<relationship>*, *<date-attr>*) **InstanzMinimum**(*<relationship>*, *<datetime-attr>*) **InstanzMinimum**(*<relationship>*, *<time-attr>*) (**der** | **die** | **das**) **kleinste** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **geringste** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **frühste** *<attr>* **für alle** *<ent>* (**der** | **die** | **das**) **erste** *<attr>* **für alle** *<ent>* *<attr>* **das für alle** *<ent>* **am geringsten ist** *<attr>* **das für alle** *<ent>* **am kleinsten ist** *<attr>* **das für alle** *<ent>* **am frühsten ist** *<attr>* **das für alle** *<ent>* **am ersten ist**	Ermittelt den niedrigsten/ältesten Wert einer ***entity-level***-Variable für alle Instanzen der ***entity***.

Syntax	Beschreibung
InstanzMinimumFalls(*<relationship>*, *<number-attr>*, *<condition>*) **InstanzMinimumFalls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanzMinimumFalls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanzMinimumFalls**(*<relationship>*, *<time-attr>*, *<condition>*) (**der** | **die** | **das**) **geringste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* (**der** | **die** | **das**) **kleinste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* (**der** | **die** | **das**) **frühste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* (**der** | **die** | **das**) **erste** *<attr>* **für alle** *<ent>* **, für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am geringsten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am kleinsten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am frühsten ist und für den es der Fall ist, dass** *<ent-test>* *<attr>* **das für alle** *<ent>* **am ersten ist und für den es der Fall ist, dass** *<ent-test>*	Ermittelt den niedrigsten/ältesten Wert einer ***entity-level***-Variable für alle Instanzen der ***entity***, für die ein bestimmtes ***entity-level attribute*** einen bestimmten Wert aufweist.
InstanzSumme(*<relationship>*, *<number-attr>*) *<attr>* **summiert für alle** *<ent>*	Ermittelt die Summe aller Instanzen einer ***entity-level***-Variable.
InstanzSummeFalls(*<relationship>*, *<number-attr>*, *<con-*	Ermittelt die Summe aller Instanzen einer ***entity-level***-Variable, für die ein boolesches ***attribute*** auf einer bestimmten ***entity-level*** für die ***entity*** wahr ist.

Syntax	Beschreibung
condition>) *<attr>* **summiert für alle** *<ent>*[**, nur dort**] **, wo** *<condition>* *<attr>* **summiert für alle** *<ent>* **, für den es der Fall ist, dass** *<condition>*	
InstanzWertFalls(*<relationship>*, *<number-attr>*, *<condition>*) **InstanzWertFalls**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanzWertFalls**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanzWertFalls**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanzWertFalls**(*<relationship>*, *<time-attr>*, *<condition>*)	Ermittelt einen Wert aus einer eindeutigen ***entity***-Instanz anhand der Ziel-***entity***-Instanzen einer ***relationship*** mittels einer Bedingung. • Wenn die Bedingung für eine einzelne Ziel-***entity***-Instanz gilt, ist der Wert der für diese ***entity***-Instanz berechnete Wert. • Wenn mehrere Zielinstanzen die Bedingung erfüllen, wird ***uncertain*** zurückgegeben. • Wenn keine Zielinstanzen die Bedingung erfüllen und die ***relationship*** bekannt ist, ist der Wert ***uncertain***.
InstanzGleich(*<instance1>*, *<instance2>*)	Legt fest, ob zwei Instanzen einer ***entity*** identisch sind.
InstanzUngleich(*<instance1>*, *<instance2>*)	Legt fest, ob zwei Instanzen einer ***entity*** nicht identisch sind.
InstanzAbleiten (*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** (**existieren** | **existiert**)	Wird in einer Konklusion verwendet, um abzuleiten, dass eine ***entity***-Instanz vorhanden und Mitglied einer ***relationship*** ist.

Zeitbasierte Funktionen(English)

Syntax	Beschreibung
IntervallAnzahlAbweichend(*<start-date>*, *<end-date>*, *<variable>*) **IntervallAnzahlAbweichend**(*<start-date>*, *<end-date>*, *<condition>*)	Zählt die Anzahl der bekannten eindeutigen Werte für die Variable im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich).
IntervallAnzahlAbweichendFalls (*<start-date>*, *<end-date>*, *<variable>*,	Zählt die Anzahl der bekannten eindeutigen Werte für die Variable im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich)

Syntax	Beschreibung
<condition>)	und berücksichtigt dabei nur die Zeiten, zu denen ein boolescher Filter wahr ist.
IntervallTäglicheSumme(*<start-date>*, *<end-date>*, *<number-attr>*)	Berechnet die Summe einer Währungs- oder Zahlenvariable im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich). Für das ***attribute*** wird von einer täglichen Menge ausgegangen.
IntervallTäglicheSummeFalls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Berechnet die Summe aller täglichen Werte für eine Währungs- oder Zahlenvariable im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich) und berücksichtigt dabei nur die Zeiten, zu denen eine Bedingung wahr ist.
IntervallMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervallMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervallMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervallMaximum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Wählt den maximalen Wert einer Variable im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich).
IntervallMaximumFalls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervallMaximumFalls**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervallMaximumFalls**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervallMaximumFalls**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Wählt den maximalen Wert einer Variable im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich) und berücksichtigt dabei nur die Zeiten, zu denen eine Bedingung wahr ist.
IntervallMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervallMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervallMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervallMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Wählt den Mindestwert einer Variable im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich).
IntervallMinimumFalls(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervallMinimumFalls**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervallMinimumFalls**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervallMinimumFalls**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Wählt den Mindestwert einer Variable im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich) und berücksichtigt dabei nur die Zeiten, zu denen eine Bedingung wahr ist.

Syntax	Beschreibung
IntervallGewichteterDurchschnitt (*<start-date>*, *<end-date>*, *<number-attribute>*)	Berechnet den Durchschnittswert einer Währungs- oder Zahlenvariablen im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich), gewichtet nach dem für den jeweiligen Wert geltenden Zeitraum.
IntervallGewichteterDurchschnittFalls (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Berechnet den Durchschnittswert einer Währungs- oder Zahlenvariablen im Intervall zwischen einem Start-***date*** (einschließlich) und einem End-***date*** (ausschließlich) und berücksichtigt dabei nur die Zeiten, zu denen eine boolesche Bedingung wahr ist (gewichtet nach dem für den jeweiligen Wert geltenden Zeitraum, für den der Filter wahr ist).
IntervallImmer(*<start-date>*, *<end-date>*, *<condition>*)	Gibt nur dann True zurück, wenn eine boolesche Bedingung immer im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich) wahr ist.
IntervallMindestensTage(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Gibt nur dann True zurück, wenn eine boolesche Bedingung für mindestens die angegebene Anzahl von Tagen (nicht unbedingt aufeinanderfolgend) im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich) wahr ist.
IntervallAufeinanderfolgendeTage (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Gibt nur dann True zurück, wenn eine boolesche Bedingung für mindestens eine bestimmte Anzahl an aufeinanderfolgenden Tagen im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich) wahr ist.
IntervallManchmal(*<start-date>*, *<end-date>*, *<condition>*)	Gibt nur dann True zurück, wenn eine boolesche Bedingung jemals im Intervall zwischen dem Start-***date*** (einschließlich) und dem End-***date*** (ausschließlich) wahr ist.
WertBei(*<date>*, *<value>*)	Gibt den Wert für das angegebene ***attribute*** am angegebenen ***date*** zurück.
WennLetzte(*<date>*, *<condition>*)	Gibt das ***date*** zurück, an dem eine boolesche Bedingung letztmals wahr war. Dabei wird ausgehend von einem angegebenen ***date*** (einschließlich) rückwärts gerechnet.
WennNächste(*<date>*, *<condition>*)	Gibt das ***date*** zurück, an dem eine boolesche Bedingung zum nächsten Mal wahr sein wird. Dabei wird ausgehend von einem angegebenen ***date*** (einschließlich) vorwärts gerechnet.
Spätestens()	Gibt einen ***date***-Wert zurück, der dem letztmöglichen ***date*** entspricht. Dies ist ein späteres ***date*** als jedes andere für ein ***date attribute*** mögliche oder für einen Ausdruck auswertbare ***date***.
Frühestens()	Gibt einen ***date***-Wert zurück, der dem frühestmöglichen ***date*** entspricht. Dies ist ein früheres ***date*** als jedes andere für ein ***date attribute*** mögliche oder für einen Ausdruck auswertbare ***date***.
ZeitlichTageSeit(*<date>*, *<end-date>*)	Gibt eine Zahlenvariable zurück, die sich täglich ändert und die die Anzahl der vollständigen Tage seit dem ***date*** darstellt.
ZeitlichWochenSeit(*<date>*, *<end-*	Gibt eine Zahlenvariable zurück, die sich wöchentlich ändert und die Anzahl der

Syntax	Beschreibung
date>)	vollständigen Wochen seit dem ***date*** darstellt.
ZeitlichMonateSeit(*<date>*, *<end-date>*)	Gibt eine Zahlenvariable zurück, die sich monatlich ändert und die Anzahl der vollständigen Monate seit dem ***date*** darstellt. Hinweis: Wenn das angegebene ***date*** nach dem 28. des Monats liegt und ein darauffolgender Monat weniger Tage als der angegebene Monat hat, wird der Änderungspunkt für den wiederkehrenden Monat am letzten Tag dieses Monats erstellt. Beispiel: Wenn das angegebene ***date*** der 28., 29., 30. oder 31. Januar 2007 ist, ist der erste Änderungspunkt der 28. Februar 2007.
ZeitlichJahreSeit(*<date>*, *<end-date>*)	Gibt eine Zahlenvariable zurück, die sich jährlich ändert und die Anzahl der vollständigen Jahre seit dem ***date*** darstellt.
ZeitlichImmerTage(*<days>*, *<condition>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und nur dann wahr ist, wenn eine boolesche Bedingung für alle Tage einer bestimmten Anzahl vorausgehender Tage mit Ausnahme des aktuellen Tages wahr ist.
ZeitlichAufeinanderfolgendeTage (*<minDays>*, *<days>*, *<condition>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und nur dann wahr ist, wenn eine boolesche Bedingung jederzeit für mindestens eine Mindestanzahl aufeinanderfolgender Tage innerhalb der vorausgehenden, festgelegten Anzahl an Tagen mit Ausnahme des aktuellen Tages wahr ist.
ZeitlichManchmalTage(*<days>*, *<condition>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und nur dann wahr ist, wenn eine boolesche Bedingung jemals innerhalb einer bestimmten Anzahl vorhergehender Tage mit Ausnahme des aktuellen Tages wahr ist.
ZeitlichNach(*<date>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und nach einem ***date*** wahr und an oder vor diesem Datum falsch ist.
ZeitlichVor(*<date>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und vor einem ***date*** wahr und an oder nach diesem Datum falsch ist.
ZeitlichAm(*<date>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und an einem ***date*** wahr und vor oder nach diesem Datum falsch ist.
ZeitlichAmOderNach(*<date>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und an oder nach einem ***date*** wahr und vor diesem Datum falsch ist.
ZeitlichAmOderVor(*<date>*)	Gibt ein boolesches ***attribute*** zurück, das sich im Laufe der Zeit ändert und an oder vor einem ***date*** wahr und nach diesem Datum falsch ist.
ZeitlichVonAnfangsdatum(*<relationship>*, *<date>*, *<value>*)	Gibt ein einzelnes zeitliches ***attribute*** (auf Ebene der Quell-***entity***) aus einer ***relationship*** und einem Wert-***attribute*** für die Entitys mit Werten zurück, die ab einem Start-***date attribute*** wirksam sind.
ZeitlichVonEnddatum(*<relationship>*, *<date>*, *<value>*)	Gibt ein einzelnes zeitliches ***attribute*** (auf Ebene der Quell-***entity***) aus einer ***relationship*** und einem Wert-***attribute*** für die Entitys mit Werten zurück, die bis zu einem End-***date attribute*** wirksam sind.

Syntax	Beschreibung
ZeitlichVonZeitraum(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Gibt ein einzelnes zeitliches ***attribute*** (auf Ebene der Quell-***entity***) aus einer ***relationship*** und einem Wert-***attribute*** für die Entitys mit Werten zurück, die von einem Start-***date attribute*** (einschließlich) bis zu einem End-***date attribute*** (ausschließlich) wirksam sind. Bei Ablauf vor dem nächsten Start-***date*** ist der Wert ***uncertain***.
ZeitlichIstWochentag(*<startdate>*, *<enddate>*)	Gibt zwischen dem angegebenen Start-***date*** (einschließlich) und dem angegebenen End-***date*** (ausschließlich) True für Datumswerte zurück, die Wochentage sind, und False für Datumswerte, die Wochenenden darstellen. Gibt außerhalb des ***date***-Bereichs ***uncertain*** zurück.
ZeitlichEinmalProMonat(*<startdate>*, *<enddate>*, *<dayofmonth>*)	Gibt zwischen dem angegebenen Start-***date*** (einschließlich) und dem angegebenen End-***date*** (ausschließlich) True zurück, wenn der Tag dem Parameter "Kalendertag" entspricht, und gibt an allen anderen Tagen des Monats False zurück. Gibt außerhalb des ***date***-Bereichs ***uncertain*** zurück. Wenn der Tag des Monats die angegebene Anzahl an Tagen im aktuellen Monat übersteigt, ist der Wert am letzten Tag dieses Monats wahr. Auf diese Weise gibt die Funktion exakt für einen Tag pro Monat den Wert True zurück.

Funktionen für Validierungsereignis(English)

Syntax	Beschreibung
Fehler (*<text>*)	Ein Fehlereignis übergibt eine Meldung an den Benutzer und verhindert solange die Fortsetzung einer Überprüfung, bis die Bedingung, die den Fehler ausgelöst hat, nicht mehr gültig ist.
Warnung (*<text>*)	Ein Warnereignis übergibt eine Meldung an den Benutzer, gestattet jedoch die Fortsetzung ungeachtet der Bedingung, die die Warnung ausgelöst hat.

Verworfene Funktionen(English)

Syntax	Beschreibung
BenutzerdefinierteFunktionAufrufen (*<A>*, *<B>*)	Gibt das Ergebnis eines externen Aufrufs einer Codebibliothek zurück. Die Codebibliothek muss für die Determinations Engine bereitgestellt werden, damit die benutzerdefinierte Funktion erfolgreich aufgerufen wird.

מחברים לוגיים(English)

תחביר	תיאור
אם **if**	מונח אופציונלי שיכול להופיע בסוף שורת מסקנה שמכילה את ההוכחה הבאה
וגם	צירוף לוגי בין שתי ***attributes***

תיאור	תחביר
	and
הפרדה לוגית בין שתי ***attributes***	**או** **or**
מרכיב הקבצה עם הפרדות כשיש צורך לקבץ שתי ***attributes*** או יותר	**זה או זה** **אחד מ** **לפחות אחד מ** **לפחות אחד מהבאים נכון** **לפחות** **לפחות אחד מהבאים** **כל אחד מהבאים הנו משביע רצון**
מרכיב הקבצה עם צירופים כשיש צורך לקבץ שתי ***attributes*** או יותר	**שניהם** **הכל** **כל הבאים נכונים** **כל הבאים משביעי רצון** **כל התנאים הבאים מתקיימים**
'מונח שמופיע בסוף של כלל טבלה לציון משפט 'אחרת	**אחר** **otherwise**
מונח שנעשה בו שימוש ברשומת המקרא שבין הביטוי המקוצר ל-***attribute text*** המלא	**הנו** **is**

פונקציות לוגיות(English)

תיאור	תחביר
מפעיל המשמש להחזרת 'אמת' אם ל-***attribute*** יש ערך 'שקר'	**לא נכון ש** *<expr>*
מפעיל המשמש להחזרת 'אמת' אם ל-***attribute*** ערך שאינו שווה ל-***uncertain***	*<var>* **הוא בטוח** *<var>* **הוא ודאי** *<var>* **היא בטוחה** *<var>* **היא ודאית** *<var>* **הוא ללא ספק** *<var>* **היא ללא ספק** **זה בטוח האם [או לא]** *<expr>* **זה ודאי האם [או לא]** *<expr>* **אין ספק האם [או לא]** *<expr>*
מפעיל המשמש להחזרת 'אמת' אם הערך של ***attribute*** הוא ***uncertain***	*<var>* **לא ודאי** *<var>* **לא בטוח** *<var>* **הוא בספק** *<var>* **לא ודאית**

תיאור	תחביר
	לא בטוחה *<var>* **היא בספק** *<var>* *<expr>* **זה לא בטוח האם** *<expr>* **זה לא ודאי האם** *<expr>* **יש ספק האם** **לא ודאי** **לא בטוח** **יש ספק**
מכילה ערך ***attribute*** מפעיל המשמש להחזרת 'אמת' אם	**ידוע** *<var>* **ידועה** *<var>* *<expr>*[**ידוע האם** [**או לא**
אין ערך ***attribute***-מפעיל המשמש להחזרת 'אמת' אם ל	**לא ידוע** *<var>* **לא ידועה** *<var>* *<expr>*[**לא ידוע האם** [**או לא** **לא ידוע**

קבועים לוגיים(English)

תיאור	תחביר
ערך 'אמת' קבוע המשמש לכללי טבלה.	**נכון** **אמת**
ערך 'שקר' קבוע המשמש לכללי טבלה.	**לא נכון** **שקר**
ערך ***uncertain*** קבוע המשמש לכללי טבלה.	**לא בטוח**

מפעילי השוואה(English)

תיאור	תחביר
קטן מ- שים לב: אין צורה של שפה טבעית כאשר משתמשים באופרטור זה עם ערכים מספריים או ערכי מטבע.	*<lhs>***<***<rhs>* *<lhs>***קטן מ***<rhs>* *<lhs>***קטנה מ***<rhs>* *<lhs>* **לפני** *<rhs>*
גדול מ- שים לב: אין צורה של שפה טבעית כאשר משתמשים באופרטור זה עם ערכים מספריים או ערכי מטבע.	*<lhs>***>***<rhs>* *<lhs>***גדול מ-***<rhs>* *<lhs>***גדולה מ***<rhs>* *<lhs>***גדול מ***<rhs>* *<lhs>***גדולה מ-***<rhs>* *<lhs>* **אחרי** *<rhs>*

תיאור	תחביר
קטן או שווה ל-	*<lhs>***<=***<rhs>* *<lhs>* **קטן או שווה ל***<rhs>*
גדול או שווה ל-	*<lhs>***>=***<rhs>* *<lhs>* **גדול או שווה ל-** *<rhs>*
שווה	*<lhs>***=***<rhs>* *<lhs>* **שווה ל***<rhs>* *<lhs>* **שווה** *<rhs>*
לא שווה	*<lhs>***<>***<rhs>* *<lhs>* **שונה מ***<rhs>*

פונקציות מספריות(English)

תיאור	תחביר
ממירה את המחרוזת שצוינה לערך מספרי	**מספר**(*<numText>*)
חיבור מתמטי	*<x>* **+** *<y>*
חיסור מתמטי	*<x>* **-** *<y>*
כפל מתמטי	
חילוק מתמטי	
חילוק של מספר חיובי שלם	
השארית אחרי חילוק מספר שלם	
מחזירה את הערך הגדול מבין השניים	**מקסימום**(*<x>*, *<y>*) **מקסימום**(*<date/time/datetime1>*, *<date/time/datetime2>*) **הערך הגבוה בין** *<val1>* **לבין** *<val2>* **המאוחר בין** *<val1>* **לבין** *<val2>*
מחזירה את הערך הקטן מבין השניים	**מינימום**(*<x>*, *<y>*) **מינימום**(*<date/time/datetime1>*, *<date/time/datetime2>*) **הערך הנמוך בין** *<val1>* **לבין** *<val2>* **המוקדם בין** *<val1>* **לבין** *<val2>*
x בחזקת y	**Xy**(*<x>*, *<y>*)
ערך e של לחזקה קבוע x	**Ex**(*<x>*)
ערך מוחלט של x	**ערך-מוחלט**(*<x>*)
לוגריתם טבעי של x	**אקספוננט**(*<x>*)

תיאור	תחביר
לוגריתם על בסיס 10 של x	**לוגריתם**(*<x>*)
שורש ריבועי של x	**שורש-ריבועי**(*<x>*) **the square root of** *<val>*
מעגלת את x למספר עשרוני	**עיגול**(*<x>*, *<n>*) *<val>* **rounded to** *<num_places>* **decimal place** *<val>* **rounded to** *<num_places>* **decimal places**
ה-x קוצץ ל-n מקומות עשרוניים	**קיצור**(*<x>*, *<n>*)
סינוס (Sine) של x	**סינוס**(*<x>*)
קוסינוס (Cosine) של x	**קוסינוס**(*<x>*)
טנגנס (Tangent) של x	**טנגנס**(*<x>*)
ארק-סינוס (Arcsine) של x	**ארק-סינוס**(*<x>*)
ארק-קוסינוס (Arccosine) של x	**ארק-קוסינוס**(*<x>*)
ארק-טנגנס (Arctangent) של x	**ארק-טנגנס**(*<x>*)

פונקציות של תאריך(English)

תיאור	תחביר
מחזיר את ה-***date*** הנוכחי בתחילת מושב העבודה.	**התאריך-היום**() **התאריך היום**
ממירה את המחרוזת שצוינה לערך של ***date***	**תאריך**(*<text>*)
מחזירה ***date*** שנוצר מן השנה, החודש והיום שצוינו.	**צור-תאריך**(*<year>*, *<month>*, *<day>*)
מחזירה את רכיב היום של ***date/datetime attribute***.	**יום-מתוך**(*<date/datetime>*)
מחזירה את רכיב החודש של ***date/datetime attribute***.	**חודש-מתוך**(*<date/datetime>*)
מחזירה את רכיב השנה של ***date/datetime attribute***.	**שנה-מתוך**(*<date/datetime>*)
מחזירה את ה-***date*** של היום הבא בשבוע או שלפני/אחרי ***date*** (בהתאם לתחביר שבו נעשה שימוש).	**היום-הבא-בשבוע**(*<date/datetime>*, *<day>*) **the next Friday on or after** *<from-date>* **the next Monday on or after** *<from-date>*

תחביר	תיאור
the next Saturday on or after *<from-date>* **the next Sunday on or after** *<from-date>* **the next Thursday on or after** *<from-date>* **the next Tuesday on or after** *<from-date>* **the next Wednesday on or after** *<from-date>*	
התאריך-הבא(*<date>*, *<day>*, *<month>*) **the previous UK tax year start date on or before** *<from-date>*	מחזירה את המופע הבא של היום והחודש הנתונים אחרי ***date***.
הוסף-ימים (*<date/datetime>*, *<num_days>*) **התאריך שחל** *<num_days>* **ימים אחרי** *<datetime>* **הזמן שחל** *<num_days>* **ימים אחרי** *<datetime>*	כשמשתמשים בצורה תחבירית מקוצרת המספר .***date***-מוסיפה או מחסירה מספר ימים מ של הקלט, או מספר שלילי כדי לחסר ***date*** צריך להיות מספר שלם חיובי כדי להוסיף ימים על ימים מה-***date*** של הקלט.
הוסף-שבועות(*<date/datetime>*, *<num_weeks>*) **התאריך שחל** *<num_weeks>* **weeks after** *<datetime>* **the time** *<num_weeks>* **weeks after** *<datetime>*	כשמשתמשים בצורה תחבירית מקוצרת המספר צריך להיות ***date***. מוסיפה מספר שבועות על של הקלט ***date*** שלם וחיובי כדי להוסיף שבועות על.
הוסף-חודשים(*<date/datetime>*, *<num_months>*) **התאריך שחל** *<num_months>* **חודשים אחרי** *<datetime>* **הזמן שחל** *<num_months>* **חודשים אחרי** *<datetime>*	כשמשתמשים בצורה התחבירית המקוצרת, המספר צריך ***date***. מוסיפה מספר חודשים על של הקלט ***date*** להיות שלם וחיובי כדי להוסיף חודשים על.
הוסף-שנים(*<date/datetime>*, *<num_years>*) **התאריך שחל** *<num_years>* **years after** *<datetime>* **the time** *<num_years>*	כשמשתמשים בצורה תחבירית מקוצרת המספר צריך להיות ***date***. מוסיפה מספר שנים על של הקלט ***date*** שלם וחיובי כדי להוסיף שנים על.

תיאור	תחביר
	years after *<datetime>*
כלומר, מספר הימים שבין יום שני ויום ששי. .***date*2**-ל ***date*1** מונה את מספר ימי השבוע שבין המאוחר יותר. ***date***-המוקדם יותר הוא כולל וה ***date***-הערה: ה	**ספירת-ימי-השבוע**(*<date1>*, *<date2>*) **the number of weekdays (inclusive) between** *<date1>* **and** *<date2>*
.חל ***date*** הראשון בשנה שבה ***date***-מחזירה את ה	**תחילת-שנה**(*<date/datetime>*) **היום הראשון בשנה עבור** *<from-date>*
.חל ***date*** האחרון בשנה שבה ***date***-מחזירה את ה	**סוף-שנה**(*<date/datetime>*) **היום האחרון בשנה עבור** *<from-date>*
סדר ההופעה .***date/datetime2***-ל ***date/datetime1*** מחזירה את מספר הימים השלמים בין .של שני התאריכים לא משפיע על התוצאה	**הפרש-בימים**(*<date/datetime1>*, *<date/datetime2>*) **מספר הימים בין** *<date1>* **לבין** *<date2>*
חישוב .***date/datetime2***-ל ***date/datetime1*** מחזירה את מספר הימים השלמים (כולל) בין זה כולל את שתי נקודות הקצה. כשהתאריכים זהים התוצאה היא 1. סדר ההופעה של שני .התאריכים לא משפיע על התוצאה	**משך-בימים**(*<date/datetime1>*, *<date/datetime2>*) **מספר הימים בין (כולל) בין** *<date1>* **לבין** *<date2>*
.***date/datetime2***-ל ***date/datetime1*** מחזירה את מספר הימים השלמים (לא כולל) בין חישוב זה אינו כולל את שתי נקודות הקצה. כשהתאריכים זהים התוצאה היא אפס (0). סדר .ההופעה של שני התאריכים לא משפיע על התוצאה	**פער-בימים**(*<date/datetime1>*, *<date/datetime2>*) **מספר הימים בין (לא כולל) בין** *<date1>* **לבין** *<date2>*
.***date/datetime2***-לס ***date/datetime1*** מחזירה את מספר השבועות השלמים שחלפו בין .סדר ההופעה של שני התאריכים לא משפיע על התוצאה	**הפרש-בשבועות**(*<date/datetime1>*, *<date/datetime2>*) **מספר השבועות בין** *<date1>* **לבין** *<date2>*
-***date/d***ו ***date/datetime1*** מחזירה את המספר הכולל של השבועות השלמים שחלפו בין .סדר ההופעה של שני התאריכים לא משפיע על התוצאה .***atetime2***	**משך-בשבועות**(*<date/datetime1>*, *<date/datetime2>*) **מספר השבועות (כולל) בין** *<date1>* **לבין** *<date2>*

תיאור	תחביר
מחזירה את המספר הלא-כולל של השבועות השלמים שחלפו בין ***date/datetime1*** ו-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**פער-בשבועות**(*<date/datetime1>*, *<date/datetime2>*) **מספר השבועות (לא כולל) בין** *<date1>* **לבין** *<date2>*
מחזירה את מספר החודשים השלמים שחלפו בין ***date/datetime1*** ל-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**הפרש-בחודשים**(*<date/datetime1>*, *<date/datetime2>*) **מספר החודשים בין** *<date1>* **לבין** *<date2>*
מחזירה את המספר הכולל של חודשים שחלפו בין ***date/datetime1*** ל-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**משך-בחודשים**(*<date/datetime1>*, *<date/datetime2>*) **מספר החודשים (לא כולל) בין** *<date1>* **לבין** *<date2>*
מחזירה את המספר הלא-כולל של החודשים השלמים שחלפו בין ***date/datetime1*** ל-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**פער-בחודשים**(*<date/datetime1>*, *<date/datetime2>*) **מספר החודשים (כולל) בין** *<date1>* **לבין** *<date2>*
מחזירה את מספר השנים שבין ***date/datetime1*** ו-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**הפרש-בשנים**(*<date/datetime1>*, *<date/datetime2>*) **מספר השנים בין** *<date1>* **לבין** *<date2>*
מחזירה את המספר הכולל של שנים שבין ***date/datetime1*** ו-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**משך-בשנים**(*<date/datetime1>*, *<date/datetime2>*) **מספר השנים (כולל) בין** *<date1>* **לבין** *<date2>*
מחזירה את המספר הלא-כולל של שנים שבין ***date/datetime1*** ו-***date/datetime2***. סדר ההופעה של שני התאריכים לא משפיע על התוצאה.	**פער-בשנים**(*<date/datetime1>*, *<date/datetime2>*) **מספר השנים (לא כולל) בין** *<date1>* **לבין** *<date2>*

פונקציות של השעה ביום(English)

תיאור	תחביר
ממירה את המחרוזת הנתונה לשעה ביום	**שעה**(*<text>*)
מחזירה את רכיב השנייה של ***timeofday/datetime attribute***.	**שנייה-מתוך**(*<time/datetime>*)
מחזירה את רכיב הדקה של ***timeofday/datetime attribute***.	**דקה-מתוך**(*<time/datetime>*)
מחזירה את רכיב השעה של ***timeofday/datetime attribute***.	**שעה-מתוך**(*<time/datetime>*)

פונקציות של תאריך ושעה(English)

תיאור	תחביר
מחזיר את ה-***date*** ואת השעה הנוכחיים בתחילת מושב העבודה.	**הזמן-כעת**()
ממירה את המחרוזת שצוינה לערך של ***datetime***	**תאריך-שעה**(*<text>*)
מגדירה את השעה ב-***date*** באמצעות צירוף יחד של השעה והיום של ***date***.	**תאריך-ושעה**(*<date>*, *<time>*) *<date>* **בשעה** *<time-of-day>* *<time-of-day>* **בתאריך** *<date>*
מחזיר את מספר השניות שבין ***datetime1*** ו-***datetime2***.	**הפרש-בשניות**(*<datetime1>*, *<datetime2>*) **הפרש-בשניות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר השניות בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הכולל של השניות שבין ***datetime1*** ו-***datetime2***.	**משך-בשניות**(*<datetime1>*, *<datetime2>*) **משך-בשניות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר השניות (כולל) בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הלא-כולל של השניות שבין ***datetime1*** ו-***datetime2***.	**פער-בשניות**(*<datetime1>*, *<datetime2>*) **פער-בשניות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר השניות (לא כולל) בין** *<date1>* **לבין**

תיאור	תחביר
	<date2>
מחזיר את מספר הדקות שבין ***datetime1*** ו-***datetime2***.	**הפרש-בדקות**(*<datetime1>*, *<datetime2>*) **הפרש-בדקות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר הדקות בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הכולל של הדקות שבין ***datetime1*** ו-***datetime2***.	**משך-בדקות**(*<datetime1>*, *<datetime2>*) **משך-בדקות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר הדקוח (לא כולל) בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הלא-כולל של הדקות שבין ***datetime1*** ו-***datetime2***.	**הפער-בדקות**(*<datetime1>*, *<datetime2>*) **הפער-בדקות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר הדקוח (כולל) בין** *<date1>* **לבין** *<date2>*
מחזיר את מספר השעות שבין ***datetime1*** ו-***datetime2***.	**הפרש-בשעות**(*<datetime1>*, *<datetime2>*) **הפרש-בשעות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר השעות בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הכולל של השעות שבין ***datetime1*** ו-***datetime2***.	**משך-בשעות**(*<datetime1>*, *<datetime2>*) **משך-בשעות** (*<timeOfDay1>*, *<timeOfDay2>*) **מספר השעות (כולל) בין** *<date1>* **לבין** *<date2>*
מחזיר את המספר הלא-כולל של השעות שבין ***datetime1*** ו-***datetime2***.	**פער-בשעות**(*<dat-*

תיאור	תחביר
	etime1>, *<datetime2>*) **פער-בשעות**(*<timeOfDay1>*, *<timeOfDay2>*) **מספר השעות (לא כולל) בין** *<date1>* **לבין** *<date2>*
מחלצת את ***date*** מ-***datetime attribute***.	**תאריך-מתוך**(*<datetime>*)
מחלצת את השעה ביום מתוך ***datetime attribute***. ניתן להשתמש בה כדי להגדיר את הערך של ***timeofday attribute*** לשעת הביצוע של הכלל, באמצעות חילוץ השעה מתוך ה-***date*** והשעה הנוכחיים.	**זמן-מתוך**(*<datetime>*)
מוסיפה מספר שעות על ***date*** של שעה.	**הוסף-שעות**(*<datetime>*, *<num_hours>*) **הוסף-שעות**(*<timeOfDay>*, *<num_hours>*) **הזמן שחל** *<num_hours>* **שעות אחרי** *<datetime>*
מוסיפה מספר דקות על ***date*** של שעה.	**הוסף-דקות**(*<datetime>*, *<num_minutes>*) **הוסף-דקות**(*<timeOfDay>*, *<num_minutes>*) **הזמן שחל** *<num_minutes>* **דקות אחרי** *<datetime>*
מוסיפה מספר שניות על ***date*** של שעה.	**הוסף-שניות**(*<datetime>*, *<num_seconds>*) **הוסף-שניות**(*<timeOfDay>*, *<num_seconds>*) **הזמן שחל** *<num_seconds>* **שניות אחרי** *<datetime>*

פונקציות של טקסט(English)

תיאור	תחביר
משלבת את **text1** עם **text2** וכן הלאה, ליצירת ערך **text** בודד. שים לב: באפשרותך להשתמש במשתנים מכל סוג. הערכים נוצרים באמצעות המפרמט המותקן במושב העבודה של הכלל.	*<text1>* **&** *<text2>*
משלבת את **text1** עם **text2** וכן הלאה, ליצירת ערך **text** בודד. שים לב: באפשרותך להשתמש במשתנים מכל סוג. הערכים נוצרים באמצעות המפרמט המותקן במושב העבודה של הכלל.	**שרשור של** *<text1>* **&** *<text2>*
מחזיר ערך בוליאני שמציין אם הערך הנתון של **text** מכיל את תת-המחרוזת הנתונה של **text**. השוואת **text** רגישה לאותיות רישיות.	**מכיל**(*<text>*, *<substring>*)
מחזיר ערך בוליאני שמציין אם הערך הנתון של **text** מסתיים בתת-המחרוזת הנתונה של **text**. השוואת **text** רגישה לאותיות רישיות.	**מסתיים-עם**(*<text>*, *<substring>*)
מחזירה ערך בוליאני שמציין אם הערך הנתון של **text** מייצג מספר תקף.	**הוא-מספר**(*<text>*)
מחזירה את האורך בתווים של ערך **text** הנתון.	**אורך**(*<text>*)
מחזיר ערך בוליאני שמציין אם הערך הנתון של **text** מתחיל בתת-המחרוזת הנתונה של **text**. השוואת **text** רגישה לאותיות רישיות.	**מתחיל-עם**(*<text>*, *<substring>*)
מחזירה את תת-המחרוזת של **text** שמתחילה בהיסט הנתון שהוא האורך שצוין בתווים. אם מגיעים לסוף המחרוזת יוחזרו פחות תווים.	**קטע-ממחרוזת**(*<text>*, *<offset>*, *<length>*)
ממירה את המספר שצוין או את **date attribute** לערך של **text**.	**טקסט**(*<number>*) **טקסט**(*<date>*) **טקסט**(*<datetime>*) **טקסט**(*<timeOfDay>*)

פונקציות של ישות וקשר(English)

תיאור	תחביר
משמשת להפניה מ-**entity** אחת ל-**entity** אחרת ב-**relationship** "יחיד ליחיד", "רבים ליחיד" או "רבים לרבים", כשיש רק תנאי אחד.	**מתקיים-בקשר**(*<relationship>*, *<Exp>*) **בקשר** *<ent>* **מתקיים** *<attr>* **מתקיים בקשר** *<ent>*,*<attr>* *<val>*, **עבור הקשר** *<ent>*
משמשת להפניה מ-**entity** אחת ל-**entity** אחרת ב-**relationship** "יחיד ליחיד", "רבים ליחיד" או "רבים לרבים", כשיש תנאי אחד או כמה תנאים.	**עבור-הקשר**(*<relationship>*, *<alias>*) **עבור-הקשר**(*<relationship>*) **עבור הקשר** *<ent>* **עבור הקשר** *<ent>* **(**<alias>**)**
משמשת להפניה מ-**entity** אחת ל-**entity** אחרת ב-**relationship** "יחיד לרבים" או "רבים לרבים", במקרה שיש צורך לקבוע אם כל החברים בקבוצת **entity** היעד צריכים להתאים לכלל הזה.	**בכל-הקשר**(*<relationship>*, *<Exp>*)

תיאור	תחביר
בצורה זו נעשה שימוש כשהכלל מכיל רק תנאי אחד.	**עבור כל מופע בקשר** *<ent>*, *<attr>* **עבור כל מופע בקשר** *<ent>* **מתקיים** *<attr>* **עבור כל מופע** *<ent-attr>*
משמשת להפניה מ-***entity*** אחת ל-***entity*** אחרת ב-***relationship*** "יחיד לרבים" או "רבים לרבים", במקרה שיש צורך לקבוע שכל החברים בקבוצת ***entity*** היעד צריכים להתאים לכלל הזה. בצורה זו נעשה שימוש כשהכלל מכיל תנאי אחד או כמה תנאים.	**בכל-מופע-בקשר**(*<relationship>*) **בכל-מופע-בקשר**(*<relationship>*, *<alias>*) **עבור כל מופע בקשר** *<ent>* **בקשר** *<ent>* **כל מופע מקיים** **עבור כל מופע בקשר** *<ent>* **(**<alias>**)** **בקשר** *<ent>* **כל מופע מקיים** **(**<alias>**)**
משמשת להפניה מ-***entity*** אחת ל-***entity*** אחרת ב-***relationship*** "יחיד לרבים" או "רבים לרבים", במקרה שיש צורך לקבוע אם יש חבר כלשהו בקבוצת ***entity*** היעד שצריך להתאים לכלל הזה. בצורה זו נעשה שימוש כשהכלל מכיל רק תנאי אחד.	**קיים-בקשר**(*<relationship>*, *<Exp>*) **בקשר** *<ent>* **יש לפחות מופע אחד המקיים** *<attr>* **לפחות מופע אחד בקשר** *<ent>*,*<attr>*
משמשת להפניה מ-***entity*** אחת ל-***entity*** אחרת ב-***relationship*** "יחיד לרבים" או "רבים לרבים", במקרה שיש צורך לקבוע אם יש חבר כלשהו בקבוצת ***entity*** היעד שצריך להתאים לכלל הזה. בצורה זו נעשה שימוש כשהכלל מכיל תנאי אחד או כמה תנאים.	**קיים-מופע-בקשר**(*<relationship>*) **קיים-מופע-בקשר**(*<relationship>*, *<alias>*) **יש לפחות מופע אחד המקיים בקשר** *<ent>* **בקשר** *<ent>* **יש לפחות מופע אחד המקיים** **יש לפחות מופע אחד המקיים בקשר** *<ent>* **(**<alias>**)** **בקשר** *<ent>* **יש לפחות מופע אחד המקיים (**<alias>**)**
משמש כתנאי שבודק אם מופע של ***entity*** הוא יעד של ***relationship*** שמקורו במופע שני של ***entity***. משמש כדי להסיק שמופע ***entity*** הוא איבר ב-***relationship***.	**חבר-בקשר**(*<target>*, *<relationship>*) **חבר-בקשר**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **(**<alias>**) חבר בקשר** *<ent>* <ent-target> **(**<alias>**) חברה בקשר** *<ent>* <ent-target> **חבר בקשר** *<ent>* <ent-target> **חברה בקשר** *<ent>*

תיאור	תחביר
שמקורו **relationship** אינו יעד של **entity** משמש כתנאי המאפשר לבדוק שמופע במופע שני של **entity**.	**אינו-חבר-בקשר**(*<target>*, *<relationship>*)
מונה את מספר המופעים הקיימים של **entity**.	**ספירת-מופעים**(*<relationship>*) **כמות מופעים בקשר** *<ent>*
מסוימת עם **entity-level attribute** שמכילה **entity** מונה את מספר המופעים של ערך מסוים.	**ספירת-מופעים-אם**(*<relationship>*, *<Exp>*) **כמות מופעים בקשר** *<ent>* **המקיימים** *<condition>*
לכל המופעים של **entity-level** מביאה את הערך הגבוה ביותר או האחרון של משתנה ה-**entity**.	**ערך-מופע-מקסימאלי**(*<relationship>*, *<number-attr>*) **ערך-מופע-מקסימאלי**(*<relationship>*, *<date-attr>*) **ערך-מופע-מקסימאלי**(*<relationship>*, *<datetime-attr>*) **ערך-מופע-מקסימאלי**(*<relationship>*, *<time-attr>*) **בקשר** *<attr>* **הערך הגבוה עבור** *<ent>* **בקשר** *<attr>* **הערך המירבי עבור** *<ent>* **הערך המקסימאלי עבור** *<attr>* **בקשר** *<ent>* *<ent>* **המאוחר בקשר** *<date-attr>* **שהינו הערך המירבי** *<max-attr>* **בקשר** *<ent>* **שהינו הערך הגבוה** *<max-attr>* **בקשר** *<ent>* **שהינו הערך** *<max-attr>* **המקסימאלי בקשר** *<ent>*
לכל המופעים של **entity-level**, מביאה את הערך הגבוה ביותר או האחרון של משתנה ה-**entity** שמכילה **entity-level attribute** מסוימת עם ערך מסוים.	**ערך-מופע-מקסימאלי-אם**(*<relationship>*, *<number-attr>*, *<condition>*) **ערך-מופע-מקסימאלי-אם**(*<relationship>*, *<date-attr>*, *<condition>*) **ערך-מופע-מקסימאלי-אם**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ערך-מופע-מקסימאלי-אם**(*<relationship>*, *<time-attr>*, *<condition>*) **בקשר** *<attr>* **הערך הגבוה עבור**

תיאור	תחביר
	<ent> **כאשר מתקיים** *<ent-test>* **הערך המירבי עבור** *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* **הערך המקסימאלי עבור** *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<date-attr>* **המאוחר בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<max-attr>* **שהינו הערך המירבי בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<max-attr>* **שהינו הערך הגבוה בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<max-attr>* **שהינו הערך המקסימאלי בקשר** *<ent>* **כאשר מתקיים** *<ent-test>*
לכל המופעים ***entity-level*** מביאה את הערך הנמוך ביותר או הישן ביותר של משתנה של ה-***entity***.	**ערך-מופע-מינימאלי**(*<relationship>*, *<number-attr>*) **ערך-מופע-מינימאלי**(*<relationship>*, *<date-attr>*) **ערך-מופע-מינימאלי**(*<relationship>*, *<datetime-attr>*) **ערך-מופע-מינימאלי**(*<relationship>*, *<time-attr>*) **הערך הנמוך עבור** *<attr>* **בקשר** *<ent>* **הערך המזערי עבור** *<attr>* **בקשר** *<ent>* **הערך המינימאלי עבור** *<attr>* **בקשר** *<ent>* *<date-attr>* **המוקדם בקשר** *<ent>*
לכל המופעים ***entity-level*** מביאה את הערך הנמוך ביותר או הישן ביותר של משתנה של ה-***entity*** שמכילה ***entity-level attribute*** מסוימת עם ערך מסוים.	**ערך-מופע-מינימאלי-אם**(*<relationship>*, *<number-attr>*, *<condition>*) **ערך-מופע-מינימאלי-אם**(*<relationship>*, *<date-attr>*, *<condition>*) **ערך-מופע-מינימאלי-אם**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ערך-מופע-מינימאלי-אם**(*<relationship>*, *<time-attr>*, *<condition>*)

תיאור	תחביר
	הערך הנמוך עבור *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* **הערך המזערי עבור** *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* **הערך המינימאלי עבור** *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<date-attr>* **המוקדם בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<num-attr>* **שהינו הערך המזערי בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<num-attr>* **שהינו הערך הנמוך בקשר** *<ent>* **כאשר מתקיים** *<ent-test>* *<num-attr>* **שהינו הערך המינימאלי בקשר** *<ent>* **כאשר מתקיים** *<ent-test>*
מביאה את סכום כל המופעים של משתנה ***entity-level***.	**סכום-מופעים**(*<relationship>*, *<number-attr>*) **סיכום כולל של כל** *<attr>* **בקשר** *<ent>* *<num-attr>* **מסוכם עבור הקשר** *<ent>*
מביאה את סכום כל המופעים של משתנה ***entity-level*** שנקבע כ'אמת', בעבור ה-***entity*** שיש לה ***attribute*** בוליאנית ***entity-level*** שנקבעה כ'אמת'.	**סכום-מופעים-אם**(*<relationship>*, *<number-attr>*, *<condition>*) **סיכום כולל של כל** *<attr>* **בקשר** *<ent>* **כאשר מתקיים** *<condition>* *<num-attr>* **מסוכם עבור הקשר** *<ent>* **כאשר מתקיים** *<ent-test>*
מביאה ערך ממופע ייחודי של ***entity*** היעד של ***relationship***, שזוהה מתוך מופעי ***entity*** באמצעות תנאי. • אם התנאי מזהה מופע בודד של ***entity*** היעד, אז הערך הוא הערך המחושב כנגד מופע זה של ***entity***. • אם יש יותר ממופע יעד אחד שמתאים לתנאי, אז מחזירה את ***uncertain***. • אם אין אף מופע יעד שמתאים לתנאי, וה-***relationship*** ידועה, אז הערך הוא ***uncertain***.	**ערך-מופע-אם**(*<relationship>*, *<number-attr>*, *<condition>*) **ערך-מופע-אם**(*<relationship>*, *<text-attr>*, *<condition>*) **ערך-מופע-אם**(*<relationship>*, *<date-attr>*, *<condition>*) **ערך-מופע-אם**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ערך-מופע-אם**(*<relationship>*, *<time-attr>*, *<condition>*)
קובעת אם שני מופעים של ***entity*** הם למעשה אחד.	**מופע-שווה**(*<instance1>*,

תיאור	תחביר
	<instance2>)
קובעת אם שני מופעים של ***entity*** הם למעשה אחד או לאו.	**מופע-שונה**(*<instance1>*, *<instance2>*)
משמש כתנאי שמסיק שמופע ***entity*** קיים והוא חבר ב-***relationship***.	**InferInstance**(*<relationship>*, *<identity>*)

פונקציות של הנמקה זמנית(English)

תיאור	תחביר
מונה את מספר הערכים הבדידים הידועים של המשתנה, במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**ספירה-מבדלת-בתקופה**(*<start-date>*, *<end-date>*, *<variable>*) **ספירה-מבדלת-בתקופה**(*<start-date>*, *<end-date>*, *<condition>*)
מונה את מספר הערכים הבדידים הידועים במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). ספירה זו כוללת רק מקרים שבהם מסנן בוליאני הוא 'אמת'.	**ספירה-מבדלת-בתקופה-אם**(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)
מחשבת את הסכום של משתנה מטבע או משתנה מספר, במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). בהנחה ש-***attribute*** היא כמות יומית.	**סיכום-בתקופה** (*<start-date>*, *<end-date>*, *<number-attr>*)
מחשבת את הסכום של כל הערכים היומיים למשתנה מטבע או משתנה מספר, במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). החישוב כולל רק את הפעמים שבהן תנאי הוא 'אמת'.	**סיכום-בתקופה-אם** (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)
בוחרת את הערך המרבי של משתנה במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מקסימום-בתקופה** (*<start-date>*, *<end-date>*, *<number-attr>*) **מקסימום-בתקופה** (*<start-date>*, *<end-date>*, *<date-attr>*) **מקסימום-בתקופה** (*<start-date>*, *<end-date>*, *<datetime-attr>*)

תיאור	תחביר
	מקסימום-בתקופה (*<start-date>*, *<end-date>*, *<time-attr>*)
בוחרת את הערך המרבי של משתנה במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). החישוב כולל רק את הפעמים שבהן תנאי הוא 'אמת'.	**מקסימום-בתקופה-אם**(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **מקסימום-בתקופה-אם**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **מקסימום-בתקופה-אם**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **מקסימום-בתקופה-אם**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)
בוחרת את הערך המזערי של משתנה במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מינימום-בתקופה** (*<start-date>*, *<end-date>*, *<number-attr>*) **מינימום-בתקופה** (*<start-date>*, *<end-date>*, *<date-attr>*) **מינימום-בתקופה** (*<start-date>*, *<end-date>*, *<datetime-attr>*) **מינימום-בתקופה** (*<start-date>*, *<end-date>*, *<time-attr>*)
בוחרת את הערך המזערי של משתנה במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). החישוב כולל רק את הפעמים שבהן תנאי הוא 'אמת'.	**מינימום-בתקופה-אם** (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **מינימום-בתקופה-אם** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **מינימום-בתקופה-אם**

תיאור	תחביר
	(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **מינימום-בתקופה-אם** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)
מחשבת את הערך הממוצע של משתנה מטבע או משתנה מספר במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). הערך הממוצע משוקלל לפי טווח הזמן שבו חל כל ערך.	**ממוצע-משוקלל-בתקופה**(*<start-date>*, *<end-date>*, *<number-attribute>*)
מחשבת את הערך הממוצע של משתנה מטבע או משתנה מספר במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל). החישוב כולל רק את הפעמים שבהן תנאי הוא 'אמת' (כלומר, הערך הממוצע משוקלל לפי טווח הזמן שבו חל כל ערך ונערך בכל פעם שהמסנן היה 'אמת').	**ממוצע-משוקלל-בתקופה-אם**(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)
מחזירה 'אמת' אם ורק כאשר תנאי בוליאני הוא אמת בכל הפעמים במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מתקיים-בתקופה-תמיד**(*<start-date>*, *<end-date>*, *<condition>*)
מחזירה 'אמת' אם ורק כאשר תנאי בוליאני הוא אמת לפחות במספר הימים שצוין (לא בהכרח בסדר עוקב) במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מתקיים-בתקופה-לפחות-ימים**(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)
מחזירה 'אמת' אם ורק כאשר תנאי בוליאני הוא אמת לפחות במספר נתון של ימים עוקבים במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מתקיים-בתקופה-ברצף-ימים**(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)
מחזירה 'אמת' אם ורק כאשר תנאי בוליאני הוא תמיד אמת במרווח שבין ***date*** ההתחלתי (כולל) עד ***date*** האחרון (לא כולל).	**מתקיים-בתקופה-לעיתים**(*<start-date>*, *<end-date>*, *<condition>*)
מחזירה את הערך של ה-***attribute*** שצוין ב-***date*** הנתונה.	**ערך-בתאריך** (*<date>*, *<value>*)
מחזירה את ה-***date*** שבו תנאי בוליאני היה 'אמת' בפעם האחרונה, במבט אחורה מ-***date*** (וכולל) שצוין.	**התקיים-בתאריך-האחרון**(*<date>*, *<condition>*)

תחביר	תיאור
מתקיים-בתאריך-הבא(*<date>*, *<condition>*)	מחזירה את ה-***date*** שבו תנאי בוליאני יהיה 'אמת' בפעם הבאה, במבט אחורה מ-***date*** (וכולל) שצוין.
תאריך-גג()	מחזירה ערך ***date*** שהוא שווה לערך ***date*** המאוחר ביותר האפשרי, קרי ***date***, שלבטח הוא יותר מאוחר מכל ***date*** שעשוי להיות ל-***date attribute*** אחר או לביטוי.
תאריך-ריצפה()	מחזירה ערך ***date*** שהוא שווה לערך ***date*** המוקדם ביותר האפשרי - קרי ***date***, שלבטח הוא יותר מוקדם מכל ***date*** שעשוי להיות ל-***date attribute*** אחר או לביטוי.
ימים-מתאריך (*<date>*, *<end-date>*)	מחזירה משתנה מספר שמשתנה כל יום והוא מבטא את מספר הימים המלאים מאז ***date***.
שבועות-מתאריך (*<date>*, *<end-date>*)	מחזירה משתנה מספר שמשתנה כל שבוע והוא מבטא את מספר השבועות המלאים מאז ***date***.
חודשים-מתאריך (*<date>*, *<end-date>*)	מחזירה משתנה מספר שמשתנה כל חודש ומבטא את מספר החודשים המלאים החל מ-***date***. שים לב: אם ה-***date*** הנתון חל אחרי היום ה-28 לחודש ואילו החודש שאחריו מכיל פחות ימים מהחודש ***date*** הנתון - אזי נקודת השינוי לחודש השנתי תיווצר ביום האחרון של החודש שאחריו. לדוגמה, אם הנתון הוא 28, 29, 30 או 31 בינואר 2007, אז נקודת השינוי הראשונה תהיה 28 בפברואר 2007.
שנים-מתאריך (*<date>*, *<end-date>*)	מחזירה משתנה מספר שמשתנה כל שנה והוא מבטא את מספר השנים המלאות מאז ***date***.
מתקיים-בכל-רצף-הימים(*<days>*, *<condition>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' רק אם תנאי בוליאני הוא 'אמת' לכל המספר הנתון של הימים הקודמים, לא כולל את היום הנוכחי.
מתקיים-ברצף-ימים-בתקופה(*<minDays>*, *<days>*, *<condition>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' רק אם תנאי בוליאני הוא 'אמת' בעבור מספר מזערי לפחות של ימים עוקבים בכל עת בטווח מספר נתון של ימים קודמים, לא כולל את היום הנוכחי.
מתקיים-ברצף-ימים (*<days>*, *<condition>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' אם ורק כאשר תנאי בוליאני הוא תמיד אמת בטווח המספר שצוין של ימים קודמים, לא כולל את היום הנוכחי.
מתרחש-לאחר-התאריך(*<date>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' אחרי ***date*** אך 'שקר' בו ולפניו.
מתרחש-לפני-התאריך(*<date>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' לפני ***date*** אך 'שקר' בו ואחריו.
מתרחש-בתאריך (*<date>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' ב-***date*** אך 'שקר' לפניו ואחריו.
מתרחש-בתאריך-או-לאחריו(*<date>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' ב-***date*** או אחריו אך 'שקר' לפניו.
מתרחש-בתאריך-או-לפניו(*<date>*)	מחזירה ***attribute*** בוליאנית שמשתנה במשך הזמן והיא 'אמת' ב-***date*** ולפניו אך 'שקר' אחריו.

תיאור	תחביר
ערך של **attribute** עם **relationship**-מ (**entity** זמנית בודדת (ברמת המקור של **attribute** מחזירה מ החל בתוקף שהיו הערכים עם ,זאת .הישויות-**date attribute** התחלתי.	**ערך-מתאריך-ההתחלה**(*<relationship>*, *<date>*, *<value>*)
ערך של **attribute** עם **relationship**-מ (**entity** זמנית בודדת (ברמת המקור של **attribute** מחזירה עד בתוקף שיהיו הערכים עם ,זאת .הישויות **date attribute** הסיום תאריך של.	**ערך-עד-תאריך-סיום**(*<relationship>*, *<date>*, *<value>*)
ערך של **attribute** עם **relationship**-מ (**entity** זמנית בודדת (ברמת המקור של **attribute** מחזירה מ בתוקף שהיו הערכים עם ,זאת .הישויות-**date attribute** עד (כולל) התחלתי **date attribute** יהיה הערך .(לא כולל) אחרון **uncertain** לפני פג תוקפו אם **date** הבא ההתחלתי.	**ערך-בטווח-תאריכים**(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)
מ החל ,שבוע סופי שהם בתאריכים 'שקר' ומחזירה השבוע ימי שהם בתאריכים 'אמת' מחזירה-**date** עד (כולל) התחלתי **date** מחזירה .(לא כולל) אחרון **uncertain** לטווח מחוץ **date**.	**ימי-השבוע-זמניים**(*<startdate>*, *<end-date>*)
מחזירה 'אמת' אם היום שווה לפרמטר 'היום בחודש' ומחזירה 'שקר' בכל שאר הימים בחודש, מחוץ **uncertain** האחרון (לא כולל). מחזירה **date** ההתחלה שצוין (כולל) ועד **date** בתקופה שבין כשהיום בחודש עובר את מספר הימים בחודש הנוכחי, הערך הוא 'אמת' ביום האחרון **date**. לטווח של חודש זה כך שהפונקציה מחזירה ערך שהוא אמת בדיוק פעם אחת בחודש.	**זמני-פעם-בחודש**(*<startdate>*, *<end-date>*, *<dayofmonth>*)

פונקציות אירוע אימות(English)

תיאור	תחביר
אירוע שגיאה משמש להעברת הודעה למשתמש, ומונע ממנו להמשיך בחקירה עד שהתנאי שהזניק את השגיאה אינו חל עוד.	**שגיאה**(*<text>*)
אירוע אזהרה משמש להעברת הודעה למשתמש, אבל מאפשר לו להמשיך למרות התנאי שהזניק אזהרה זו.	**אזהרה**(*<text>*)

פונקציות מוחלפות(English)

תיאור	תחביר
מחזירה את התוצאה של קריאה חיצונית לספריית קודים. על מנת שהקריאה לפונקצית הקסטומיזציה תצליח, יש לספק את קוד הספריה למנוע ההגדרות (Oracle Determinations Engine).	**פונקציה-מקומית**(*<A>*, *<B>*)

Connettori logici(English)

Sintassi	Descrizione
se	Termine opzionale che può apparire alla fine di una riga conclusiva cui segue una prova
e	Congiunzione logica tra due valori ***attributes***
o	Disgiunzione logica tra due valori ***attributes***
oppure **uno di** **qualsiasi** **almeno uno dei seguenti è vero** **uno qualsiasi dei seguenti è soddisfatto**	Elemento di raggruppamento utilizzato con disgiunzioni per raggruppare due o più ***attributes***
entrambi **tutti** **tutti i seguenti sono veri** **tutti i seguenti sono soddisfatti**	Elemento di raggruppamento utilizzato con congiunzioni per raggruppare due o più ***attributes***
altrimenti	Termine che appare alla fine di una regola tabella per indicare la clausola altrimenti
è	Termine utilizzato in una voce legenda tra la frase abbreviata e il valore ***attribute text*** completo

Funzioni logiche(English)

Sintassi	Descrizione
non è vero che *<expr>*	Operatore utilizzato per restituire vero se ***attribute*** contiene un valore che è falso
<var> **è certo** *<var>* **è certa** *<var>* **sono certe** *<var>* **sono certi** **è certo se** *<expr>*	Operatore utilizzato per restituire vero se ***attribute*** contiene un valore diverso da ***uncertain***
<var> **non è certo** *<var>* **non è certa** *<var>* **non sono certe** *<var>* **non sono certi** **non è certo se** *<expr>* **non è certo se** *<expr>* **non è certo che** *<expr>* **non certo**	Operatore utilizzato per restituire vero se il valore di ***attribute*** è ***uncertain***
<var> **è noto** *<var>* **è nota**	Operatore utilizzato per restituire vero se ***attribute*** contiene un valore

Sintassi	Descrizione
<var> **sono note** *<var>* **sono noti** **è noto se** *<expr>*	
<var> **non è noto** *<var>* **non è nota** *<var>* **non sono note** *<var>* **non sono noti** **non è noto se** *<expr>* **non noto**	Operatore utilizzato per restituire vero se ***attribute*** non contiene alcun valore

Costanti logiche(English)

Sintassi	Descrizione
vero	Valore vero costante utilizzato per le regole tabella.
falso	Valore falso costante utilizzato per le regole tabella.
non certo	Valore ***uncertain*** costante utilizzato per le regole tabella.

Operatori di confronto(English)

Sintassi	Descrizione
*<x>***<***<y>* *<x>* **minore** (**di** | **del** | **dell'** | **dell’** | **della** | **delle** | **dei** | **degli**)*<y>*	Minore di Nota: non esiste una forma di linguaggio naturale in cui tale operatore viene utilizzato con valori numerici o di valuta.
<x> **>** *<y>* *<x>* **maggiore** (**di** | **del** | **dell'** | **dell’** | **della** | **delle** | **dei** | **degli**)*<y>*	Maggiore di Nota: non esiste una forma di linguaggio naturale in cui tale operatore viene utilizzato con valori numerici o di valuta.
*<x>***<=***<y>* *<x>* **minore o uguale** (**a** | **al** | **all'** | **all’** | **alla** | **alle** | **di** | **del** | **dell'** | **dell’** | **della** | **delle** | **dei** | **degli**)*<y>*	Minore di o uguale a
<x> **>=** *<y>* *<x>* **maggiore o uguale** (**a** | **al** | **all'** | **all’** | **alla** | **alle** | **di** | **del** | **dell'** | **dell’** | **della** | **delle** | **dei** | **degli**)*<y>*	Maggiore di o uguale a
*<x>***=***<y>* *<x>* **uguale** (**a** | **al** | **all'** | **all’** | **alla** | **alle**)*<y>*	Uguale
<x> **<>** *<y>* *<x>* **diverso** (**da** | **dal** | **dall'** | **dall’** | **della** | **delle**)*<y>*	Diverso

Funzioni numeriche(English)

Sintassi	Descrizione
Numero(*<numText>*)	Converte la stringa specificata in un valore numerico
<x> **+** *<y>*	Addizione matematica
<x> **-** *<y>*	Sottrazione matematica
<x> * *<y>*	Moltiplicazione matematica
<x> **/** *<y>*	Divisione matematica
<x> \ *<y>*	Divisione di un intero
<x> **modulo** *<y>*	Resto rimanente dalla divisione di un intero
Massimo(*<x>*, *<y>*) **Massimo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Restituisce il più grande di due valori
Minimo(*<x>*, *<y>*) **Minimo**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Restituisce il più piccolo di due valori
Xy(*<x>*, *<y>*)	x elevata alla potenza di y
Ex(*<x>*)	Costante e elevata alla potenza di x
Ass(*<x>*) **|***<val>***|**	Valore assoluto di x
Ln(*<x>*)	Logaritmo naturale di x
Log(*<x>*)	Logaritmo in base 10 di x
Radq(*<x>*)	Radice quadrata di x
Arrotonda(*<x>*, *<n>*)	Arrotonda a n posizioni decimali
Tronca(*<x>*, *<n>*)	x troncata a n posizioni decimali
Sen(*<x>*)	Seno di x
Cos(*<x>*)	Coseno di x
Tan(*<x>*)	Tangente di x
Arcsen(*<x>*)	Arcoseno di x
Arccos(*<x>*)	Arcocoseno di x
Arctan(*<x>*)	Arcotangente di x

Funzioni data(English)

Sintassi	Descrizione
DataAttuale() **data corrente**	Restituisce il valore ***date*** corrente all'inizio della sessione.
Data(*<text>*)	Converte la stringa specificata in un valore ***date***
CalcolaData(*<year>*, *<month>*, *<day>*)	Restituisce un valore ***date*** formato dall'anno, dal mese e dal giorno specificati.
EstraiGiorno(*<date/datetime>*)	Restituisce il componente giorno di un valore ***date/datetime attribute***.
EstraiMese(*<date/datetime>*)	Restituisce il componente mese di un valore ***date/datetime attribute***.
EstraiAnno(*<date/datetime>*)	Restituisce il componente anno di un valore ***date/datetime attribute***.
GiornoSuccessivoDellaSettimana(*<date/datetime>*, *<day>*)	Restituisce la data ***date*** del giorno feriale successivo corrispondente o anteriore/successivo a ***date*** (in base alla sintassi utilizzata).
DataSuccessiva(*<date>*, *<day>*, *<month>*)	Restituisce l'istanza successiva del giorno e del mese specificati dopo un valore ***date***.
AggiungiGiorni(*<date/datetime>*, *<num_days>*)	Aggiunge/sottrae un numero di giorni a un valore ***date***. Quando si usa la forma sintattica concisa, il numero deve essere un intero positivo per aggiungere giorni al valore ***date*** di input, oppure un numero negativo per sottrarre giorni al valore ***date*** di input.
AggiungiSettimane(*<date/datetime>*, *<num_weeks>*)	Aggiunge un numero di settimane a un valore ***date***. Quando si usa la forma sintattica concisa, il numero deve essere un intero positivo per aggiungere settimane al valore ***date*** di input.
AggiungiMesi(*<date/datetime>*, *<num_months>*)	Aggiunge un numero di mesi a un valore ***date***. Quando si usa la forma sintattica concisa, il numero deve essere un intero positivo per aggiungere mesi al valore ***date*** di input.
AggiungiAnni(*<date/datetime>*, *<num_years>*)	Aggiunge un numero di anni a un valore ***date***. Quando si usa la forma sintattica concisa, il numero deve essere un intero positivo per aggiungere anni al valore ***date*** di input.
NumeroGiorniFeriali(*<date1>*, *<date2>*)	Conteggia il numero di giorni feriali tra ***date***1 e ***date***2, ovvero il numero di giorni compresi tra lunedì e venerdì. Nota: il primo valore ***date*** è incluso, l'ultimo ***date*** escluso.
InizioAnno(*<date/datetime>*)	Restituisce il primo valore ***date*** dell'anno in cui cade il valore ***date***.
FineAnno(*<date/datetime>*)	Restituisce l'ultimo valore ***date*** dell'anno in cui cade il valore ***date***.
DifferenzaGiorni(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di giorni interi tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaGiorniInclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di giorni interi (in modalità di inclusione) compresi tra ***date/datetime1*** e ***date/datetime2***. Il calcolo include il giorno di inizio e il giorno di fine. Se le date coincidono, il risultato è uguale a uno. L'ordine delle due date non influenza il

Sintassi	Descrizione
	risultato.
DifferenzaGiorniEsclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di giorni interi (in modalità di esclusione) compresi tra ***date/datetime1*** e ***date/datetime2***. Il calcolo esclude il giorno di inizio e il giorno di fine. Se le date coincidono, il risultato è uguale a zero. L'ordine delle due date non influenza il risultato.
DifferenzaSettimane(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di settimane intere trascorse tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato
DifferenzaSettimaneInclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di settimane intere trascorse (in modalità di esclusione) tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaSettimaneEsclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di settimane intere trascorse (in modalità di esclusione) tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato
DifferenzaMesi(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di mesi interi trascorsi tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaMesiInclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di mesi interi trascorsi (in modalità di inclusione) compresi tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaMesiEsclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di mesi interi trascorsi (in modalità di esclusione) tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaAnni(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di anni tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato
DifferenzaAnniInclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di anni (in modalità di inclusione) tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.
DifferenzaAnniEsclusa(*<date/datetime1>*, *<date/datetime2>*)	Restituisce il numero di anni (in modalità di esclusione) tra ***date/datetime1*** e ***date/datetime2***. L'ordine delle due date non influenza il risultato.

Funzioni ora del giorno(English)

Sintassi	Descrizione
OraDelGiorno(*<text>*)	Converte la stringa specificata in un'ora del giorno.
EstraiSecondo(*<time/datetime>*)	Restituisce il componente secondo di un valore ***timeofday/datetime attribute***.
EstraiMinuto(*<time/datetime>*)	Restituisce il componente minuto di un valore ***timeofday/datetime attribute***.
EstraiOra(*<time/datetime>*)	Restituisce il componente ora di un valore ***timeofday/datetime attribute***.

Funzioni data e ora(English)

Sintassi	Descrizione
DataOraAttuale()	Restituisce il valore ***date*** e ora corrente all'inizio della sessione.
DataOra(*<text>*)	Converte la stringa specificata in un valore ***datetime***
ConcatenaDataOra (*<date>*, *<time>*)	Imposta il valore ***date*** e ora collegando il valore ***date*** all'ora del giorno.
DifferenzaSecondi(*<datetime1>*, *<datetime2>*) **DifferenzaSecondi** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di secondi tra ***datetime1*** e ***datetime2***.
DifferenzaSecondiInclusa (*<datetime1>*, *<datetime2>*) **DifferenzaSecondiInclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di secondi (in modalità di inclusione) tra ***datetime1*** e ***datetime2***.
DifferenzaSecondiEsclusa (*<datetime1>*, *<datetime2>*) **DifferenzaSecondiEsclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di secondi (in modalità di esclusione) tra ***datetime1*** e ***datetime2***.
DifferenzaMinuti(*<datetime1>*, *<datetime2>*) **DifferenzaMinuti** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di minuti tra ***datetime1*** e ***datetime2***.
DifferenzaMinutiInclusa (*<datetime1>*, *<datetime2>*) **DifferenzaMinutiInclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di minuti (in modalità di inclusione) tra ***datetime1*** e ***datetime2***.
DifferenzaMinutiEsclusa (*<datetime1>*, *<datetime2>*) **DifferenzaMinutiEsclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di minuti (in modalità di esclusione) tra ***datetime1*** e ***datetime2***.
DifferenzaOre(*<datetime1>*, *<datetime2>*) **DifferenzaOre** (*<timeOfDay1>*,	Restituisce il numero di ore tra ***datetime1*** e ***datetime2***.

Sintassi	Descrizione
<timeOfDay2>)	
DifferenzaOreInclusa (*<datetime1>*, *<datetime2>*) **DifferenzaOreInclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di ore (in modalità di inclusione) tra ***datetime1*** e ***datetime2***.
DifferenzaOreEsclusa (*<datetime1>*, *<datetime2>*) **DifferenzaOreEsclusa** (*<timeOfDay1>*, *<timeOfDay2>*)	Restituisce il numero di ore (in modalità di esclusione) tra ***datetime1*** e ***datetime2***.
EstraiData(*<datetime>*)	Estrae il valore ***date*** da un valore ***datetime attribute***.
EstraiOraDelGiorno(*<datetime>*)	Estrae l'ora del giorno da un valore ***datetime attribute***. La funzione può essere utilizzata per impostare un valore ***timeofday attribute*** sull'ora di esecuzione della regola estraendo l'ora dal valore ***date*** e ora attuali.
AggiungiOre(*<datetime>*, *<num_hours>*) **AggiungiOre** (*<timeOfDay>*, *<num_hours>*)	Aggiunge un numero di ore a un valore ***date*** e ora.
AggiungiMinuti(*<datetime>*, *<num_minutes>*) **AggiungiMinuti** (*<timeOfDay>*, *<num_minutes>*)	Aggiunge un numero di minuti a un valore ***date*** e ora.
AggiungiSecondi(*<datetime>*, *<num_seconds>*) **AggiungiSecondi** (*<timeOfDay>*, *<num_seconds>*)	Aggiunge un numero di secondi a un valore ***date*** e ora.

Funzioni testo(English)

Sintassi	Descrizione
<text1> **&** *<text2>*	Combina ***text1*** e ***text2*** in modo da formare un valore ***text*** singolo. Nota: è possibile utilizzare variabili di qualsiasi tipo. I valori vengono formattati utilizzando il formatter installato nella sessione delle regole.
	Combina ***text1*** e ***text2*** in modo da formare un valore ***text*** singolo.

Sintassi	Descrizione
	Nota: è possibile utilizzare variabili di qualsiasi tipo. I valori vengono formattati utilizzando il formatter installato nella sessione delle regole.
Contiene(*<text>*, *<substring>*)	Restituisce un valore booleano che indica se il valore ***text*** specificato contiene la sottostringa ***text*** specificata. Per il confronto ***text*** non è rilevante la distinzione tra maiuscole e minuscole.
TerminaCon (*<text>*, *<substring>*)	Restituisce un valore booleano che indica se il valore ***text*** specificato finisce con la sottostringa ***text*** specificata. Per il confronto ***text*** non è rilevante la distinzione tra maiuscole e minuscole.
ÈUnNumero (*<text>*)	Restituisce un valore booleano che indica se il valore ***text*** specificato rappresenta un numero valido.
Lunghezza(*<text>*)	Restituisce la lunghezza di caratteri del valore ***text*** specificato.
IniziaCon(*<text>*, *<substring>*)	Restituisce un valore booleano che indica se il valore ***text*** specificato inizia con la sottostringa ***text*** specificata. Per il confronto ***text*** non è rilevante la distinzione tra maiuscole e minuscole.
Sottostringa (*<text>*, *<offset>*, *<length>*)	Restituisce la sottostringa del valore ***text*** che inizia all'offset specificato, vale a dire la lunghezza specificata in caratteri. Viene restituito un numero inferiore di caratteri se viene raggiunta la fine della stringa.
Testo(*<number>*) **Testo**(*<date>*) **Testo**(*<datetime>*) **Testo** (*<timeOfDay>*)	Converte il numero o il valore ***date attribute*** specificato in un valore ***text***.

Funzioni entità e relazione(English)

Sintassi	Descrizione
Per(*<relationship>*, *<Exp>*) **nel caso del** *<relationship>***,** *<attr> <val>***, nel caso del** *<relationship>*	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-uno, molti-a-uno o molti-a-molti dove esiste una sola condizione.
PerAmbito(*<relationship>*, *<alias>*) **PerAmbito**(*<relationship>*) **nel caso** (**del** | **di**)*<relationship>* **nel caso del** *<rela-*	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-uno, molti-a-uno o molti-a-molti dove esistono una o più condizioni.

Sintassi	Descrizione
tionship> **(**<alias>**)**	
PerTutto (*<relationship>*, *<Exp>*)	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-molti o molti-a-molti, quando si desidera stabilire se tutti i membri del gruppo ***entity*** di destinazione devono soddisfare la regola. Questa forma viene utilizzata quando nella regola è presente una sola condizione.
PerTuttoAmbito(*<relationship>*) **PerTuttoAmbito**(*<relationship>*, *<alias>*)	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-molti o molti-a-molti, quando si desidera stabilire se tutti i membri del gruppo ***entity*** di destinazione devono soddisfare la regola. Questa forma viene utilizzata quando nella regola sono presenti una o più condizioni.
Esiste(*<relationship>*, *<Exp>*)	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-molti o molti-a-molti, quando si desidera stabilire se un membro del gruppo ***entity*** di destinazione deve soddisfare la regola. Questa forma viene utilizzata quando nella regola è presente una sola condizione.
EsisteAmbito(*<relationship>*) **EsisteAmbito**(*<relationship>*, *<alias>*)	Funzione utilizzata per creare un riferimento da una ***entity*** a un'altra ***entity*** in una ***relationship*** uno-a-molti o molti-a-molti, quando si desidera stabilire se un membro del gruppo ***entity*** di destinazione deve soddisfare la regola. Questa forma viene utilizzata quando nella regola sono presenti una o più condizioni.
ÈMembroDi(*<target>*, *<relationship>*) **ÈMembroDi**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **è un membro** (**di** | **dei** | **del**)*<relationship>* <ent-target> **(**<alias>**)** **è un membro** (**di** | **dei** | **del**)*<relationship>*	Utilizzato come conclusione per dedurre che un'istanza di ***entity*** è membro di una relazione ***relationship***. Utilizzato come condizione per verificare se un'istanza di ***entity*** è destinazione di una relazione ***relationship*** che ha come origine una seconda istanza di ***entity***.
NonÈMembroDi(*<target>*, *<relationship>*)	Funzione utilizzata come condizione per verificare che una istanza di ***entity*** non sia una destinazione di una ***relationship*** per cui una seconda istanza di ***entity*** rappresenta l'origine.
NumeroIstanze(*<relationship>*)	Conteggia il numero di istanze esistenti per una ***entity***.
NumeroIstanzeSe (*<relationship>*, *<Exp>*)	Conteggia il numero di istanze esistenti per una ***entity*** per cui un ***entity-level attribute*** particolare contiene un valore particolare.
NumeroMaxIstanze (*<relationship>*, *<number-attr>*) **NumeroMaxIstanze** (*<relationship>*, *<date-*	Ottiene il valore massimo/più recente di una variabile ***entity-level*** per tutte le istanze di una ***entity***.

Sintassi	Descrizione
attr>) **NumeroMaxIstanze** (*<relationship>*, *<datetime-attr>*) **NumeroMaxIstanze** (*<relationship>*, *<time-attr>*)	
NumeroMaxIstanzeSe (*<relationship>*, *<number-attr>*, *<condition>*) **NumeroMaxIstanzeSe** (*<relationship>*, *<date-attr>*, *<condition>*) **NumeroMaxIstanzeSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **NumeroMaxIstanzeSe** (*<relationship>*, *<time-attr>*, *<condition>*)	Ottiene il valore massimo/più recente di una variabile ***entity-level*** per tutte le istanze di una ***entity*** per cui un ***entity-level attribute*** particolare contiene un valore particolare.
NumeroMinIstanze (*<relationship>*, *<number-attr>*) **NumeroMinIstanze** (*<relationship>*, *<date-attr>*) **NumeroMinIstanze** (*<relationship>*, *<datetime-attr>*) **NumeroMinIstanze** (*<relationship>*, *<time-attr>*)	Ottiene il valore minimo/meno recente di una variabile ***entity-level*** per tutte le istanze di una ***entity***.
NumeroMinIstanzeSe (*<relationship>*, *<number-attr>*, *<condition>*) **NumeroMinIstanzeSe** (*<relationship>*, *<date-attr>*, *<condition>*) **NumeroMinIstanzeSe** (*<relationship>*, *<datetime-attr>*, *<condition>*) **NumeroMinIstanzeSe** (*<relationship>*, *<time-*	Ottiene il valore minimo/meno recente di una variabile ***entity-level*** per tutte le istanze di una ***entity*** per cui un ***entity-level attribute*** particolare contiene un valore particolare.

Sintassi	Descrizione
attr>, *<condition>*)	
SommaIstanze(*<relationship>*, *<number-attr>*)	Ottiene la somma di tutte le istanze di una variabile ***entity-level***.
SommaIstanzeSe (*<relationship>*, *<number-attr>*, *<condition>*)	Ottiene la somma di tutte le istanze di una variabile ***entity-level*** per cui è vero un ***attribute*** booleano ***entity-level*** specifico di una ***entity***.
ValoreIstanzaSe(*<relationship>*, *<number-attr>*, *<condition>*) **ValoreIstanzaSe**(*<relationship>*, *<text-attr>*, *<condition>*) **ValoreIstanzaSe**(*<relationship>*, *<date-attr>*, *<condition>*) **ValoreIstanzaSe**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ValoreIstanzaSe**(*<relationship>*, *<time-attr>*, *<condition>*)	Ottiene un valore da una istanza ***entity*** univoca, identificata dalle istanze ***entity*** di destinazione di una ***relationship*** in base a una condizione. • Se la condizione identifica una istanza ***entity*** di destinazione singola, il valore corrisponde al valore calcolato a fronte di quella istanza ***entity***. • Se più di una istanza di destinazione soddisfa la condizione, viene restituito ***uncertain***. • Se nessuna istanza di destinazione soddisfa la condizione e la ***relationship*** è nota, il valore è ***uncertain***.
IstanzeUguali (*<instance1>*, *<instance2>*)	Determina se due istanze di una ***entity*** coincidono.
IstanzeDiverse (*<instance1>*, *<instance2>*)	Determina se due istanze di una ***entity*** non coincidono.
DeduciIstanza(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** **esiste**	Funzione utilizzata come conclusione per dedurre che esiste un'istanza ***entity*** ed è membro di una ***relationship***.

Funzioni ragionamento temporale(English)

Sintassi	Descrizione
NumeroIntervalloDistinto (*<start-date>*, *<end-date>*, *<variable>*)	Conteggia il numero di valori distinti noti per la variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.

Sintassi	Descrizione
NumeroIntervalloDistinto(*<start-date>*, *<end-date>*, *<condition>*)	
NumeroIntervalloDistintoSe(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Conteggia il numero di valori distinti noti per la variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso, includendo solo le volte in cui è vero un filtro booleano.
SommaGiornalieraIntervallo(*<start-date>*, *<end-date>*, *<number-attr>*)	Calcola la somma di una variabile valuta o numerica nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso. Si suppone che il valore ***attribute*** corrisponda a una quantità giornaliera.
SommaGiornalieraIntervalloSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Calcola la somma di tutti i valori giornalieri di una variabile valuta o numerica nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso, includendo solo le volte in cui una condizione è vera.
IntervalloMassimo(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalloMassimo**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalloMassimo**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalloMassimo**(*<start-date>*, *<end-date>*, *<time-attr>*)	Seleziona il valore massimo di una variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.
IntervalloMassimoSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalloMassimoSe**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalloMassimoSe**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalloMassimoSe**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Seleziona il valore massimo di una variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso, includendo solo le volte in cui una condizione è vera.
IntervalloMinimo(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalloMinimo**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalloMinimo**(*<start-date>*, *<end-date>*, *<datetime-attr>*)	Seleziona il valore minimo di una variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.

Sintassi	Descrizione
IntervalloMinimo(*<start-date>*, *<end-date>*, *<time-attr>*)	
IntervalloMinimoSe(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalloMinimoSe**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalloMinimoSe**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalloMinimoSe**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Seleziona il valore minimo di una variabile nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso, includendo solo le volte in cui una condizione è vera.
MediaPonderataIntervallo (*<start-date>*, *<end-date>*, *<number-attribute>*)	Calcola il valore medio di una variabile valuta o numerica nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso ponderato in base all'intervallo di tempo a cui si applica il valore.
MediaPonderataIntervalloSe (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Calcola il valore medio di una variabile valuta o numerica nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso includendo solo le volte in cui una condizione booleana è vera, ponderato in base all'intervallo di tempo a cui si applica il valore e laddove il filtro è vero.
IntervalloSempre(*<start-date>*, *<end-date>*, *<condition>*)	Restituisce vero se e solo se è tutte le volte vera una condizione booleana nell'intervallo compreso tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.
GiorniIntervalloAlmeno(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Restituisce vero se e solo se è vera una condizione booleana per almeno il numero di giorni non necessariamente consecutivi specificato nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.
GiorniConsecutiviIntervallo (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Restituisce vero se e solo se è vera una condizione booleana per almeno un numero di giorni consecutivi specificato nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.
IntervalloAVolte(*<start-date>*, *<end-date>*, *<condition>*)	Restituisce vero se e solo se è sempre vera una condizione booleana nell'intervallo tra il valore ***date*** iniziale incluso e il valore ***date*** finale escluso.
ValoreAl(*<date>*, *<value>*)	Restituisce il valore di un dato ***attribute*** in corrispondenza del valore ***date*** specificato.
QuandoUltimo(*<date>*, *<condition>*)	Restituisce l'ultimo valore ***date*** in cui una condizione booleana è stata vera, procedendo all'indietro a partire da un valore ***date*** specificato (incluso).
QuandoSuccessivo(*<date>*, *<condition>*)	Restituisce il prossimo valore ***date*** in cui una condizione booleana sarà vera, procedendo in avanti a partire da un valore ***date*** specificato (incluso).
Ultimo()	Restituisce un valore ***date*** equivalente all'ultimo valore ***date*** possibile, vale a dire un

Sintassi	Descrizione
	valore ***date*** sicuramente successivo a qualunque altro valore ***date*** che un valore ***date attribute*** potrebbe assumere o un'espressione potrebbe restituire.
Primo()	Restituisce un valore ***date*** equivalente al primo valore ***date*** possibile, vale a dire un valore ***date*** sicuramente precedente a qualunque altro valore ***date*** che un valore ***date attribute*** potrebbe assumere o un'espressione potrebbe restituire.
TemporaleGiorniDa(*<date>*, *<end-date>*)	Restituisce una variabile numerica che varia ogni giorno e che corrisponde al numero di giorni interi dal valore ***date***.
TemporaleSettimaneDa (*<date>*, *<end-date>*)	Restituisce una variabile numerica che varia ogni settimana e che corrisponde al numero di settimane intere dal valore ***date***.
TemporaleMesiDa(*<date>*, *<end-date>*)	Restituisce una variabile numerica che varia ogni mese e che corrisponde al numero di mesi interi dal valore ***date***. Nota: se il valore ***date*** specificato è successivo al ventottesimo giorno del mese e il numero di giorni del mese successivo è inferiore a quello del mese specificato, il punto di svolta per il mese anniversario verrà creato l'ultimo giorno di quel mese. Ad esempio, se il valore ***date*** specificato corrisponde al 28, 29, 30 o 31 gennaio 2007, il primo punto di svolta sarà il 28 febbraio 2007.
TemporaleAnniDa(*<date>*, *<end-date>*)	Restituisce una variabile numerica che varia ogni anno e che corrisponde al numero di anni interi dal valore ***date***.
TemporaleGiorniSempre (*<days>*, *<condition>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero se e solo se è vera una condizione booleana per l'intero numero di giorni precedenti specificato, giorno corrente escluso.
TemporaleGiorniConsecutivi (*<minDays>*, *<days>*, *<condition>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero se e solo se è vera una condizione booleana per almeno un numero minimo di giorni consecutivi all'interno del numero precedente di giorni impostato, giorno corrente escluso.
TemporaleGiorniAVolte (*<days>*, *<condition>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero se e solo se è sempre vera una condizione booleana all'interno di un numero di giorni precedenti specificato, giorno corrente escluso.
TemporaleDopo(*<date>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero dopo il valore ***date*** e falso prima e alla data del valore.
TemporalePrima(*<date>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero prima del valore ***date*** e falso alla data del valore e successivamente.
TemporaleIl(*<date>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero in corrispondenza del valore ***date*** e falso prima e successivamente.
TemporaleIlODopo(*<date>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero in corrispondenza di o successivamente al valore ***date*** e falso prima.
TemporaleIlOPrima(*<date>*)	Restituisce un ***attribute*** booleano che varia nel tempo ed è vero in corrispondenza di e prima del valore ***date*** e falso successivamente.

Sintassi	Descrizione
TemporaleDaDataInizio(*<relationship>*, *<date>*, *<value>*)	Restituisce un ***attribute*** temporale singolo a livello di ***entity*** di origine da una ***relationship*** e un ***attribute*** valore sulle entità, con valori che diventano effettivi da un valore ***date attribute*** di inizio.
TemporaleDaDataFine(*<relationship>*, *<date>*, *<value>*)	Restituisce un ***attribute*** temporale singolo a livello di ***entity*** di origine da una ***relationship*** e un ***attribute*** valore sulle entità, con valori effettivi fino a un valore ***date attribute*** di fine.
TemporaleDaIntervallo(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Restituisce un ***attribute*** temporale singolo a livello di ***entity*** di origine da una ***relationship*** e un ***attribute*** valore sulle entità, con valori che diventano effettivi da un valore ***date attribute*** inizio incluso fino a un valore ***date attribute*** fine escluso. Il valore è ***uncertain*** se scade prima del valore ***date*** di inizio successivo.
TemporaleÈGiorniFeriali (*<startdate>*, *<enddate>*)	Restituisce vero se le date corrispondono a giorni feriali e falso se le date corrispondono a giorni festivi a partire dal valore ***date*** inizio specificato incluso fino al valore ***date*** fine escluso. Restituisce ***uncertain*** all'esterno dell'intervallo di valori ***date***.
TemporaleUnaVoltaAlMese (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Restituisce vero se il giorno corrisponde al parametro giorno del mese e falso tutti gli altri giorni del mese a partire dal valore ***date*** inizio specificato incluso fino al valore ***date*** fine escluso. Restituisce ***uncertain*** all'esterno dell'intervallo del valore ***date***. Se il valore di giorno del mese supera il numero di giorni del mese corrente, il valore è vero l'ultimo giorno di quel mese, pertanto la funzione restituisce un valore che è vero esattamente un giorno al mese.

Funzioni evento di convalida(English)

Sintassi	Descrizione
Errore (*<text>*)	Viene utilizzato un evento di errore per passare all'utente un messaggio ed evitare che continui ad indagare fino a quando la condizione che ha attivato l'errore non sarà più valida.
Avvertenza (*<text>*)	Viene utilizzato un evento di avvertenza per passare un messaggio all'utente, che può però continuare nonostante la condizione che ha attivato l'avvertenza.

Funzioni obsolete(English)

Sintassi	Descrizione
ChiamaFunzionePersonalizzata (*<A>*, *<B>*)	Restituisce il risultato di una chiamata esterna a una libreria di codice. È necessario fornire la libreria di codice al motore delle determinazioni perché la chiamata di funzione personalizzata abbia esito positivo.

論理 コネクタ(English)

構文	摘要
もし もし	次のプルーフが含まれる結論行の最後に指定可能なオプションの条件
および **および** **また** **かつ** **それで** **そして** **又** **且つ** **及び**	2つの***attributes***の論理積
または **それとも** **もしくは** **あるいは** **又は** **若しくは** **或いは**	2つの***attributes***の論理和
いずれか **次のうち 1 つ** **任意** **次の少なくとも 1 つが真** **次のいずれかを満たす** **もしくは** **あるいは**	2つ以上の***attributes***のグループ化が必要な場合に、論理和とあわせて使用するグループ化要素
両方 **すべて** **次のすべてが真** **次のすべてを満たす** **また** **かつ**	2つ以上の***attributes***のグループ化が必要な場合に、論理積とともに使用するグループ化要素
それ以外	表ルールの最後に出現する、それ以外句を示す条件
=	凡例エントリの短縮句と短縮していない***attribute text***の間で使用する条件

論理関数(English)

構文	摘要
次の条件が真ではない *<expr>*	演算子で、使用すると***attribute***の値が偽の場合に真を返します

構文	摘要
<var> **は明確** **次の条件を満たすかどうかが明確** *<expr>*	演算子で、使用すると***attribute***の値が***uncertain***でない場合に真を返します
<var> **は不明瞭** **次の条件が不明瞭** *<expr>* **次の条件を満たすかどうかが不明瞭** *<expr>* **次の条件が不明確** *<expr>* **不明瞭**	演算子で、使用すると***attribute***の値が***uncertain***の場合に真を返します
<var> **は既知** **次の条件を満たすかどうかが既知** *<expr>*	演算子で、使用すると***attribute***に値が含まれている場合に真を返します
<var> **は不明** **次の条件を満たすかどうか不明** *<expr>* **不明**	演算子で、使用すると***attribute***に値が含まれていない場合に真を返します

論理定数(English)

構文	摘要
真	表ルールで使用する真の定数値です。
偽	表ルールで使用する偽の定数値
不明瞭	表ルールで使用する***uncertain***の定数値

比較演算子(English)

構文	摘要
*<x>***<***<y>*	次より小さい 注意: この演算子が数値および通貨の値とともに使用される場合、自然言語フォームはありません。
<x> **>** *<y>*	次より大きい 注意: この演算子が数値および通貨の値とともに使用される場合、自然言語フォームはありません。
*<x>***<=***<y>*	次以下
<x> **>=** *<y>*	次以上
*<x>***=***<y>*	等しい
<x> **<>** *<y>*	等しくない

数値関数(English)

構文	摘要
番号(<numText>)	指定した文字列を数値に変換します
<x> **+** <y>	数値の加算
<x> **-** <y>	数値の減算
<x> * <y>	数値の乗算
<x> **/** <y>	数値の除算
<x> \ <y>	整数の除算
<x> **modulo** <y>	整数を除算した余り
最大値(<x>, <y>) 最大値(<date/time/datetime1>, <date/time/datetime2>)	2つの値のうち大きい方を返します
最小値(<x>, <y>) 最小値(<date/time/datetime1>, <date/time/datetime2>)	2つの値のうち小さい方を返します
Xのy乗(<x>, <y>)	xのy乗
e指数(<x>)	定数eのx乗
絶対値(<x>) **|**<val>**|**	xの絶対値
自然対数(<x>)	xの自然対数
常用対数(<x>)	10を底とするxの対数
平方根(<x>)	xの平方根
四捨五入(<x>, <n>)	xを小数点以下n桁に四捨五入します
切り捨て(<x>, <n>)	小数点以下n桁に切り捨てたx
正弦(<x>)	xの正弦
余弦(<x>)	xの余弦
正接(<x>)	xの正接
逆正弦(<x>)	xの逆正弦
逆余弦(<x>)	xの逆余弦
逆正接(<x>)	xの逆正接

日付関数(English)

構文	摘要
現在の日付()	セッションの開始時に現在の***date***を返します。
日付(*<text>*)	指定した文字列を***date***値に変換します
日付の作成(*<year>*, *<month>*, *<day>*)	指定した年、月および日から作成した***date***を返します。
抽出日(*<date/datetime>*)	***date/datetime attribute***の日付コンポーネントを返します。
抽出月(*<date/datetime>*)	***date/datetime attribute***の月コンポーネントを返します。
抽出年(*<date/datetime>*)	***date/datetime attribute***の年コンポーネントを返します。
次の曜日(*<date/datetime>*, *<day>*)	***date***(***date***以前/以降の次の平日)を返します(使用する構文によります)。
次の日付(*<date>*, *<day>*, *<month>*)	***date***の指定日数後および指定月数後の次のインスタンスを返します。
日付の追加(*<date/datetime>*, *<num_days>*)	***date***に対して日数を加算または減算します。簡潔な構文形式を使用する場合は、入力***date***に日数を加算するには数値が正の整数である必要があり、または入力***date***から日数を減算するには数値が負の数値である必要があります。
週の追加(*<date/datetime>*, *<num_weeks>*)	***date***に週数を加算します。簡潔な構文形式を使用する場合は、入力***date***に週数を加算するために、数値が正の整数である必要があります。
月の追加(*<date/datetime>*, *<num_months>*)	***date***に月数を加算します。簡潔な構文形式を使用する場合は、入力***date***に月数を加算するために、数値が正の整数である必要があります。
年の追加(*<date/datetime>*, *<num_years>*)	***date***に年数を加算します。簡潔な構文形式を使用する場合は、入力***date***に年数を加算するために、数値が正の整数である必要があります。
平日の日数(*<date1>*, *<date2>*)	***date***1と***date***2の間の平日の日数をカウントします。つまり、月曜日から金曜日までの曜日の日数です。 注意: 古い***date***は含まれますが、新しい***date***は除外されます。
年の開始(*<date/datetime>*)	最初の***date***(***date***が含まれる年の)を返します。
年の終了(*<date/datetime>*)	最後の***date***(***date***が含まれる年の)を返します。
日単位の差異(*<date/datetime1>*, *<date/d-*	***date/datetime1***と***date/datetime2***の間の、満で数えた日数を返します。2つの日付の順序は、結果に影響しません。

構文	摘要
atetime2>)	
日単位の差異(包含)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた日数(包含)を返します。この計算では、両方のエンドポイントが含まれます。日付が同じ場合、結果は1になります。2つの日付の順序は、結果に影響しません。
日単位の差異(除外)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた日数(除外)を返します。この計算では、両方のエンドポイントは含まれません。日付が同じ場合、結果は0になります。2つの日付の順序は、結果に影響しません。
週単位の差異(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過週数を返します。2つの日付の順序は、結果に影響しません。
週単位の差異(包含)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過週数(包含)を返します。2つの日付の順序は、結果に影響しません。
週単位の差異(除外)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過週数(除外)を返します。2つの日付の順序は、結果に影響しません。
月単位の差異(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過月数を返します。2つの日付の順序は、結果に影響しません。
月単位の差異(包含)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過月数(包含)を返します。2つの日付の順序は、結果に影響しません。
月単位の差異(除外)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の、満で数えた経過月数(除外)を返します。2つの日付の順序は、結果に影響しません。
年単位の差異(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の年数を返します。2つの日付の順序は、結果に影響しません。
年単位の差異(包含)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の年数(包含)を返します。2つの日付の順序は、結果に影響しません。
年単位の差異(除外)(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1***と***date/datetime2***の間の年数(除外)を返します。2つの日付の順序は、結果に影響しません。

時刻関数(English)

構文	摘要
時刻(*<text>*)	指定した文字列を時刻に変換します
抽出秒(*<time/datetime>*)	***timeofday/datetime attribute***の秒コンポーネントを返します。
抽出分(*<time/datetime>*)	***timeofday/datetime attribute***の分コンポーネントを返します。
抽出時(*<time/datetime>*)	***timeofday/datetime attribute***の時間コンポーネントを返します。

日時関数(English)

構文	摘要
現在の日時()	セッションの開始時に現在の***date***および時刻を返します。
日時(*<text>*)	指定した文字列を***datetime***値に変換します
日時の連結(*<date>*, *<time>*)	***date***の時間の設定(***date***と時刻との結合による)を行います。
秒単位の差異(*<datetime1>*, *<datetime2>*) **秒単位の差異(*<timeOfDay1>*, *<timeOfDay2>*)**	***datetime1***と***datetime2***の間の秒数を返します。
秒単位の差異(包含)(*<datetime1>*, *<datetime2>*) **秒単位の差異(包含)(*<timeOfDay1>*, *<timeOfDay2>*)**	***datetime1***と***datetime2***の間の秒数(包含)を返します。
秒単位の差異(除外)(*<datetime1>*, *<datetime2>*) **秒単位の差異(除外)(*<timeOfDay1>*, *<timeOfDay2>*)**	***datetime1***と***datetime2***の間の秒数(除外)を返します。
分単位の差異(*<datetime1>*, *<datetime2>*) **分単位の差異(*<timeOfDay1>*, *<timeOfDay2>*)**	***datetime1***と***datetime2***の間の分数を返します。
分単位の差異(包含)(*<datetime1>*, *<datetime2>*) **分単位の差異(包含)(*<timeOfDay1>*, *<timeOfDay2>*)**	***datetime1***と***datetime2***の間の分数(包含)を返します。

構文	摘要
分単位の差異(除外)(*<datetime1>*, *<datetime2>*) **分単位の差異(除外)**(*<timeOfDay1>*, *<timeOfDay2>*)	***datetime1***と***datetime2***の間の分数(除外)を返します。
時単位の差異(*<datetime1>*, *<datetime2>*) **時単位の差異**(*<timeOfDay1>*, *<timeOfDay2>*)	***datetime1***と***datetime2***の間の時間数を返します。
時単位の差異(包含)(*<datetime1>*, *<datetime2>*) **時単位の差異(包含)**(*<timeOfDay1>*, *<timeOfDay2>*)	***datetime1***と***datetime2***の間の時間数(包含)を返します。
時単位の差異(除外)(*<datetime1>*, *<datetime2>*) **時単位の差異(除外)**(*<timeOfDay1>*, *<timeOfDay2>*)	***datetime1***と***datetime2***の間の時間数(除外)を返します。
抽出日(*<datetime>*)	***datetime attribute***から***date***を抽出します。
抽出時刻(*<datetime>*)	***datetime attribute***から時刻を抽出します。現在の***date***と時刻から時刻を抽出してルールを実行する時刻に***timeofday attribute***の値を設定する場合に使用できます。
時間の追加(*<datetime>*, *<num_hours>*) **時間の追加**(*<timeOfDay>*, *<num_hours>*)	***date***の時刻に時間数を加算します。
分の追加(*<datetime>*, *<num_minutes>*) **分の追加**(*<timeOfDay>*, *<num_minutes>*)	***date***の時刻に分数を加算します。
秒の追加(*<datetime>*, *<num_seconds>*) **秒の追加**(*<timeOfDay>*, *<num_seconds>*)	***date***の時刻に秒数を加算します。

テキスト関数(English)

構文	摘要
<text1> **&** *<text2>*	***text1***を***text2***と結合し、単一の***text***値を形成します。 注意: 任意のタイプの変数を使用できます。値は、ルール・セッションにインストールされているフォーマッタを使用して書式設定されます。
	text1を***text2***と結合し、単一の***text***値を形成します。 注意: 任意のタイプの変数を使用できます。値は、ルール・セッションにインストールされているフォーマッタを使用して書式設定されます。
含む(*<text>*, *<substring>*)	指定した***text***値に、指定した***text***のサブ文字列が含まれているかどうかを示すブール値を返します。***text***比較では大文字小文字は区別されません。
最後(*<text>*, *<substring>*)	指定した***text***値が、指定した***text***のサブ文字列で終わっているかどうかを示すブール値を返します。***text***比較では大文字小文字は区別されません。
数値(*<text>*)	指定した***text***の値が、有効な数値を表しているかどうかを示すブール値を返します。
長さ(*<text>*)	指定した***text***値の文字の長さを返します。
先頭(*<text>*, *<substring>*)	指定した***text***値が、指定した***text***のサブ文字列で始まっているかどうかを示すブール値を返します。***text***比較では大文字小文字は区別されません。
サブ文字列(*<text>*, *<offset>*, *<length>*)	指定したオフセット、つまり指定した文字の長さで始まる***text***のサブ文字列を返します。文字列の終わりに達した場合は、指定より少ない文字が返されます。
テキスト(*<number>*) テキスト(*<date>*) テキスト(*<datetime>*) テキスト(*<timeOfDay>*)	指定した数値または***date attribute***を***text***値に変換します。

エンティティおよび関係関数(English)

構文	摘要
合致(*<relationship>*, *<Exp>*)	条件が1つしか存在しない"一対一"、"多対一"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します。
範囲(*<relationship>*, *<alias>*) 範囲(*<relationship>*)	1つ以上の条件がある"一対一"、"多対一"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します。
すべてで合致(*<relationship>*, *<Exp>*)	"一対多"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します(ターゲット***entity***のグループのすべてのメンバーがルールを満たす必要があるかどうかの判別が必要な場合)。 このフォームは、ルールに条件が1つしか存在しない場合に使用します。
すべての範囲で合致(*<relationship>*) すべての範囲で合致(*<relationship>*, *<alias>*)	"一対多"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します(ターゲット***entity***のグループのすべてのメンバーがルールを満たす必要があるかどうかの判別が必要な場合)。 このフォームは、ルールに条件が1つ以上存在する場合に使用します。

構文	摘要
存在(*<relationship>*, *<Exp>*)	"一対多"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します(ターゲット***entity***のグループのいずれかのメンバーがルールを満たす必要があるかどうかの判別が必要な場合)。 このフォームは、ルールに条件が1つしか存在しない場合に使用します。
範囲で合致(*<relationship>*) 範囲で合致(*<relationship>*, *<alias>*)	"一対多"または"多対多"の***relationship***において、ある***entity***から別の***entity***までの範囲で指定するために使用します(ターゲット***entity***のグループのいずれかのメンバーがルールを満たす必要があるかどうかの判別が必要な場合)。 このフォームは、ルールに条件が1つ以上存在する場合に使用します。
メンバー(*<target>*, *<relationship>*) メンバー(*<target>*, *<alias>*, *<relationship>*)	***entity***インスタンスが***relationship***のメンバーであると推測するための結論として使用します。 ***entity***インスタンスが、2番目の***entity***インスタンスがソースである***relationship***のターゲットであることをテストする条件として使用します。
非メンバー(*<target>*, *<relationship>*)	***entity***インスタンスが、2番目の***entity***インスタンスがソースである***relationship***のターゲットでないことをテストする条件として使用します。
インスタンスの数(*<relationship>*)	存在する***entity***のインスタンスの数をカウントします。
次の条件時のインスタンスの数(*<relationship>*, *<Exp>*)	特定の***entity-level attribute***に特定の値が含まれている***entity***のインスタンスの数をカウントします。
インスタンス最大(*<relationship>*, *<number-attr>*) インスタンス最大(*<relationship>*, *<date-attr>*) インスタンス最大(*<relationship>*, *<datetime-attr>*) インスタンス最大(*<relationship>*, *<time-attr>*)	***entity***のすべてのインスタンスについて、***entity-level***変数の最高値/最新の値を取得します。
次の条件時のインスタンス最大(*<relationship>*, *<number-attr>*, *<condition>*) 次の条件時のインスタンス最大(*<relationship>*, *<date-attr>*, *<condition>*) 次の条件時のインスタンス最大(*<relationship>*, *<datetime-attr>*, *<condition>*) 次の条件時のインスタンス最大(*<relationship>*, *<time-attr>*, *<condition>*)	特定の***entity-level attribute***に特定の値が含まれている***entity***のすべてのインスタンスについて、***entity-level***変数の最高値/最新の値を取得します。
インスタンス最小(*<rela-*	***entity***のすべてのインスタンスについて、***entity-level***変数の最低値/最も古い値を取得しま

構文	摘要
tionship>, *<number-attr>*) インスタンス最小(*<relationship>*, *<date-attr>*) インスタンス最小(*<relationship>*, *<datetime-attr>*) インスタンス最小(*<relationship>*, *<time-attr>*)	す。
次の条件時のインスタンス最小(*<relationship>*, *<number-attr>*, *<condition>*) 次の条件時のインスタンス最小(*<relationship>*, *<date-attr>*, *<condition>*) 次の条件時のインスタンス最小(*<relationship>*, *<datetime-attr>*, *<condition>*) 次の条件時のインスタンス最小(*<relationship>*, *<time-attr>*, *<condition>*)	特定の***entity-level attribute***に特定の値が含まれている***entity***のすべてのインスタンスについて、***entity-level***変数の最低値/最も古い値を取得します。
インスタンス合計(*<relationship>*, *<number-attr>*)	***entity-level***変数のすべてのインスタンスの合計を取得します。
次の条件時のインスタンス合計(*<relationship>*, *<number-attr>*, *<condition>*)	***entity-level***変数(***entity***の特定の***entity-level***ブール***attribute***が真である)のすべてのインスタンスの合計を取得します。
次の条件時のインスタンス値(*<relationship>*, *<number-attr>*, *<condition>*) 次の条件時のインスタンス値(*<relationship>*, *<text-attr>*, *<condition>*) 次の条件時のインスタンス値(*<relationship>*, *<date-attr>*, *<condition>*) 次の条件時のインスタンス値(*<relationship>*, *<datetime-attr>*, *<condition>*) 次の条件時のインスタンス値(*<relationship>*, *<time-attr>*, *<condition>*)	一意の***entity***インスタンスから値を取得します。***relationship***のターゲット***entity***インスタンスの中から、対象を条件で識別します。 • 条件によって単一のターゲット***entity***インスタンスが識別された場合、値はその***entity***インスタンスに対して計算を行った値になります。 • 複数のターゲット・インスタンスが条件を満たす場合は、***uncertain***が返されます。 • 条件を満たすターゲット・インスタンスが存在せず、***relationship***が既知の場合、値は***uncertain***になります。
インスタンスが等しい(*<instance1>*, *<instance2>*)	***entity***の2つのインスタンスが同じインスタンスであるかどうかを判別します。

構文	摘要
インスタンスが等しくない(*<instance1>*, *<instance2>*)	***entity***の2つのインスタンスが同じインスタンスでないかどうかを判別します。
InferInstance(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)存在する**	***entity***インスタンスが存在し、***relationship***のメンバーであると推測するための結論として使用します。

時間推論関数(English)

構文	摘要
間隔実数(*<start-date>*, *<end-date>*, *<variable>*) **間隔実数**(*<start-date>*, *<end-date>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内における、変数の既知の実数値の数をカウントします。
次の条件時の間隔実数(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内で、ブール・フィルタが真である時間のみを対象に、変数の既知の実数値の数をカウントします。
合計日単位間隔(*<start-date>*, *<end-date>*, *<number-attr>*)	開始***date***(包含)から終了***date***(除外)までの間隔内における、通貨または数値変数の合計を計算します。***attribute***は、1日当たりの数量とみなされます。
次の条件時の合計日単位間隔(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内で、条件が真である時間のみを対象に、通貨または数値変数のすべての1日当たりの値の合計を計算します。
最大間隔(*<start-date>*, *<end-date>*, *<number-attr>*) **最大間隔**(*<start-date>*, *<end-date>*, *<date-attr>*) **最大間隔**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **最大間隔**(*<start-date>*, *<end-date>*, *<time-attr>*)	開始***date***(包含)から終了***date***(除外)までの間隔内における、変数の最大値を選択します。

構文	摘要
次の条件時の最大間隔(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **次の条件時の最大間隔(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*)** **次の条件時の最大間隔(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*)** **次の条件時の最大間隔(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)**	開始 ***date*** (包含)から終了 ***date*** (除外)までの間隔内で、条件が真である時間のみを対象に、変数の最大値を選択します。
最小間隔(*<start-date>*, *<end-date>*, *<number-attr>*) **最小間隔(*<start-date>*, *<end-date>*, *<date-attr>*)** **最小間隔(*<start-date>*, *<end-date>*, *<datetime-attr>*)** **最小間隔(*<start-date>*, *<end-date>*, *<time-attr>*)**	開始 ***date*** (包含)から終了 ***date*** (除外)までの間隔内における、変数の最小値を選択します。
次の条件時の最小間隔(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **次の条件時の最小間隔(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*)** **次の条件時の最小間隔(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*)** **次の条件時の最小間隔(*<start-date>*, *<end-date>*, *<time-attr>*,**	開始 ***date*** (包含)から終了 ***date*** (除外)までの間隔内で、条件が真である時間のみを対象に、変数の最小値を選択します。

構文	摘要
<condition>)	
加重平均間隔(*<start-date>*, *<end-date>*, *<number-attribute>*)	開始***date***(包含)から終了***date***(除外)までの間隔内における、通貨または数値変数の平均値を計算します。この平均値は、それぞれの値に適用される期間で加重が付加されます。
次の条件時の加重平均間隔(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内で、ブール条件が真である時間のみを対象に、通貨または数値変数の平均値を計算します(それぞれの値に適用される期間およびフィルタが真である期間で加重が付加されます)。
常時間隔(*<start-date>*, *<end-date>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内にあるすべての時間のブール条件が真の場合にのみ、真を返します。
最低日数間隔(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内で、指定した日数(連続でなくてもかまいません)以上のブール条件が真である場合にのみ、真を返します。
連続日数間隔(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの間隔内で、指定した連続日数以上のブール条件が真である場合にのみ、真を返します。
暫時間隔(*<start-date>*, *<end-date>*, *<condition>*)	開始***date***(包含)から終了***date***(除外)までの期間内で、ブール条件が真のものが1つでもあれば、真を返します。
該当値(*<date>*, *<value>*)	指定した***date***における、指定した***attribute***の値を返します。
最後の場合(*<date>*, *<condition>*)	最後にブール条件が真になった***date***を返します。指定した***date***から(当日を含めて)過去にさかのぼって検索します。
次の場合(*<date>*, *<condition>*)	次にブール条件が真になる***date***を返します。指定した***date***から(当日を含めて)先の日付に向かって検索します。
最後()	***date***値(最遅可能***date***に相当)を返します - つまり、***date***は***date attribute***でかかる時間または式で評価する他のすべての***date***よりも遅いことが保証されます。
先頭()	***date***値(最早可能***date***に相当)を返します - つまり、***date***は***date attribute***でかかる時間または式で評価する他のすべての***date***よりも早いことが保証されます。
次の日時以降の日数(*<date>*, *<end-date>*)	数値変数を返します。この変数は毎日変化し、***date***以降の満で数えた日数になります。
次の日時以降の週数	数値変数を返します。この変数は毎週変化し、***date***以降の満で数えた週数になります。

構文	摘要
(*<date>*, *<end-date>*)	
次の日時以降の月数(*<date>*, *<end-date>*)	数値変数を返します。この変数は毎月変化し、***date***以降の満で数えた月数になります。注意: 指定した***date***がその月の28日より後で、次の月の日数が指定月よりも少ない場合、月ごとの変更ポイントは、その月の最後の日に作成されます。たとえば、指定した***date***が2007年1月28、29、30または31日の場合、最初の変更ポイントは2007年2月28日になります。
次の日時以降の年数(*<date>*, *<end-date>*)	数値変数を返します。この変数は毎年変化し、***date***以降の満で数えた年数になります。
常時日数(*<days>*, *<condition>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、直近の指定日数(現在の日付は含まない)のすべてでブール条件が真の場合にのみ、真になります。
連続日数(*<minDays>*, *<days>*, *<condition>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、直近の指定日数内(現在の日付は含まない)のいずれかの時点で、少なくとも最低連続日数のブール条件が真の場合にのみ、真になります。
暫時日数(*<days>*, *<condition>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、直近の指定日数内(現在の日付は含まない)にブール条件が真のものが1つでもあれば、真になります。
次の日時以降(*<date>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、***date***より後の場合は真、それ以前の場合は偽です。
次の日時以前(*<date>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、***date***より前の場合は真、それ以降の場合は偽です。
次の日時(*<date>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、***date***である場合は真、それより前や後の場合は偽です。
次の日時またはそれ以降(*<date>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、***date***以降の場合は真、それより前の場合は偽です。
次の日時またはそれ以前(*<date>*)	ブール***attribute***を返します。この値は時間の経過とともに変化し、***date***以前の場合は真、それより後の場合は偽です。
開始日から(*<relationship>*, *<date>*, *<value>*)	***relationship***から(ソース***entity***レベルの)単一の時間***attribute***とそのエンティティの値***attribute***を返します。その値は開始***date attribute***から有効です。
終了日から(*<relationship>*, *<date>*, *<value>*)	***relationship***から(ソース***entity***レベルの)単一の時間***attribute***とそのエンティティの値***attribute***を返します。その値は終了***date attribute***まで有効です。
範囲から(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	***relationship***から(ソース***entity***レベルの)単一の時間***attribute***とそのエンティティの値***attribute***を返します。その値は開始***date attribute***(包含)から終了***date attribute***(除外)まで有効です。この値は、次の開始***date***より前に期限が切れる場合、***uncertain***になります。
平日(*<startdate>*, *<enddate>*)	指定した開始***date***(包含)から終了***date***(除外)までの日付が平日の場合は真を、週末の場合は偽を返します。***date***範囲外の場合は、***uncertain***を返します。

構文	摘要
1か月に1回(*<startdate>*, *<end-date>*, *<dayofmonth>*)	日付がday-of-monthパラメータと等しい場合は真を、指定した開始***date***(包含)から終了***date***(除外)の範囲にある他のすべての日付である場合は偽を返します。***date***範囲外の場合は、***uncertain***を返します。day-of-monthが現在の月の日数を超えた場合、その月の最後の日に真になります。そのため、この関数では、ちょうど1か月に1日、真の値が返されます。

検証イベント関数(English)

構文	摘要
エラー (*<text>*)	エラー・イベントは、ユーザーにメッセージを伝達し、エラーを発生させた条件が適用されなくなるまでユーザーが調査を続行できないようにするために使用されます。
警告 (*<text>*)	警告イベントは、ユーザーにメッセージを伝達するために使用されますが、その警告を発生させた条件にかかわらずユーザーが続行することを許可します。

非推奨関数(English)

構文	摘要
カスタム機能の呼び出し(*<A>*, *<B>*)	コード・ライブラリへの外部呼び出しの結果を返します。カスタム機能の呼び出しを成功させるには、Determinations Engineにコード・ライブラリが提供されている必要があります。

논리적 연결자(English)

구문	설명
if 만일 만약	뒤따르는 검증이 있는 결론 라인의 끝에 표시할 수 있는 선택적 조건입니다.
and 그리고	두 ***attributes*** 간의 논리적 결합
or 또는 혹은	두 ***attributes*** 간의 논리적 분리
둘 중 하나 다음 중 하나가 충족됨 다음 중 하나가 충족합니다 다음 중 하나에 해당합니다 다음 중 하나가 충족되었습니다 다음 중 하나 임의	두 개 이상의 ***attributes***을(를) 그룹핑해야 하는 경우, 분리에 사용하는 그룹핑 요소

구문	설명
다음 중 최소 하나가 참 다음 중 무언가가 충족됨	
both 둘 다 모두 다음 모두가 참 다음 모두가 충족됨	두 개 이상의 ***attributes***을(를) 그룹핑해야 하는 경우, 결합에 사용하는 그룹핑 요소
그렇지 않으면	그렇지 않으면 절을 나타내기 위해 테이블 규칙 끝에 표시하는 조건입니다.
=	축약된 구문 및 전체 ***attribute text*** 사이의 범례 항목에서 사용되는 조건입니다.

논리적 함수(English)

구문	설명
사실이 아닙니다 (은 | 는)*<expr>* 사실이 아님**(***<expr>***)** **IsNotTrue(***<expr>***)**	***attribute***의 값이 거짓인 경우 참을 반환하기 위해 사용하는 연산자
(확실함 | **IsCertain**)**(***<var>***)** *<var>*(은 | 이 | 는 | 가 |) 확실함 (확실함 | **IsCertain**)**(***<expr>***)** 확실함 (은 | 이 | 는 | 가 |)*<expr>*	***attribute***의 값이 ***uncertain***이 아닌 경우 참을 반환하기 위해 사용하는 연산자
<var>(은 | 이 | 는 | 가 |) 확실하지 않음 (확실하지 않음 | **IsUncertain**)**(***<var>***)** *<var>*(은 | 이 | 는 | 가 |) 불확실함 (불확실함)**(***<var>***)** 여부가 확실하지 않습니다 (가 | 는)*<expr>* (확실하지 않음 | **IsUncertain**)**(***<expr>***)** 확실하지 않음(은 | 이 | 는 | 가 |)*<expr>* (불확실함)**(***<expr>***)** 불확실함 (은 | 이 | 는 | 가 |)*<expr>*	***attribute***의 값이 ***uncertain***인 경우 참을 반환하는 연산자입니다.
<var>(은 | 을 | 는 | 를 |) 알 수 있음 (알 수 있음 | **IsKnown**)**(***<var>***)** *<var>*(를 | 을) 확인하였음 (알 수 있음 | **IsKnown**)**(***<expr>***)** 알 수 있음 (은 | 을 | 는 | 를 |)*<expr>*	***attribute***에 값이 있는 경우 참을 반환하는 데 사용하는 연산자입니다.

구문	설명
<*var*>(은 | 을 | 는 | 를 |) 알 수 없음 (알 수 없음 | **IsUnKnown**)(<*var*>) (알 수 없음 | **IsUnKnown**)(<*expr*>) 알 수 없음(은 | 을 | 는 | 를 |)<*expr*>	***attribute***에 값이 없는 경우 참을 반환하는 데 사용하는 연산자입니다.

논리적 상수(English)

구문	설명
참	테이블 규칙에 사용되는 상수 참 값입니다.
거짓	테이블 규칙에 사용되는 상수 거짓 값입니다.
확실하지 않음 **불확실함**	테이블 규칙에 사용되는 상수 ***uncertain*** 값입니다.

비교 연산자(English)

구문	설명
<*x*>**<**<*y*>	다음보다 작음 참고: 이 연산자가 숫자 및 통화 값에 대해 사용될 경우 자연 언어 형식이 없습니다.
<*x*> **>** <*y*>	다음보다 큼 참고: 이 연산자가 숫자 및 통화 값에 대해 사용될 경우 자연 언어 형식이 없습니다.
<*x*>**<=**<*y*>	다음보다 작거나 같음
<*x*> **>=** <*y*>	다음보다 크거나 같음
<*x*>**=**<*y*>	같음
<*x*> **<>** <*y*>	같지 않음

숫자 함수(English)

구문	설명
숫자(<*numText*>)	지정된 문자열을 숫자 값으로 변환합니다.
<*x*> **+** <*y*>	수학적 더하기
<*x*> **-** <*y*>	수학적 빼기
<*x*> ***** <*y*>	수학적 곱하기
<*x*> **/** <*y*>	수학적 나누기

구문	설명
<x> **** *<y>*	정수 나누기
<x> **modulo** *<y>*	정수 나누기 후 나머지
최대(*<x>*, *<y>*) 최대(*<date/time/datetime1>*, *<date/time/datetime2>*)	두 값 중 큰 값을 반환합니다.
최소(*<x>*, *<y>*) 최소(*<date/time/datetime1>*, *<date/time/datetime2>*)	두 값 중 작은 값을 반환합니다.
Xy(*<x>*, *<y>*)	y를 지수로 한 x의 거듭제곱
Ex(*<x>*)	x를 지수로 한 상수 e의 거듭제곱
절대값(*<x>*) **|***<val>***|**	x의 절대값
자연 로그(*<x>*)	x의 자연 로그
로그(*<x>*)	밑이 10인 x의 로그
제곱근(*<x>*)	x의 제곱근
반올림(*<x>*, *<n>*)	x를 소수점 n자리까지 반올림합니다.
내림(*<x>*, *<n>*)	x를 소수점 n자리까지 자릅니다.
Sin(*<x>*)	x의 사인
Cos(*<x>*)	x의 코사인
Tan(*<x>*)	x의 탄젠트
Asin(*<x>*)	x의 아크사인
Acos(*<x>*)	x의 아크코사인
Atan(*<x>*)	x의 아크탄젠트

날짜 함수(English)

구문	설명
현재날짜()	세션 시작 시 현재 ***date***을(를)반환합니다.
날짜(*<text>*)	지정된 문자열을 ***date***값으로 변환합니다.
날짜생성(*<year>*, *<month>*, *<day>*)	지정된 연도, 월 및 일로 구성된 ***date***을(를)반환합니다.
일추출(*<date/datetime>*)	***date/datetime attribute***의 일 구성 요소를 반환합니다.

구문	설명
월추출(*<date/datetime>*)	***date/datetime attribute***의 월 구성 요소를 반환합니다.
연도추출(*<date/datetime>*)	***date/datetime attribute***의 연도 구성 요소를 반환합니다.
다음날짜(*<date/datetime>*, *<day>*)	***date*** 또는 그 이전/이후의 다음 평일의 ***date***을(를) 반환합니다(사용된 구문에 따름).
다음지정날짜(*<date>*, *<day>*, *<month>*)	***date***(으)로부터 지정된 일 및 월 이후에 해당하는 다음 인스턴스를 반환합니다.
일추가(*<date/datetime>*, *<num_days>*)	***date***에 일수를 더하거나 뺍니다. 간결한 구문 형식을 사용하는 경우, 입력 ***date***에 일을 더하려면 해당 숫자가 양의 정수여야 하고, 입력 ***date***에서 일을 빼려면 음의 정수여야 합니다.
주추가(*<date/datetime>*, *<num_weeks>*)	***date***에 주 수를 더합니다. 간결한 구문 형식을 사용하는 경우, 입력 ***date***에 주를 더하려면 해당 숫자가 양의 정수여야 합니다.
월추가(*<date/datetime>*, *<num_months>*)	***date***에 개월 수를 더합니다. 간결한 구문 형식을 사용하는 경우, 입력 ***date***에 개월을 더하려면 해당 숫자가 양의 정수여야 합니다.
연도추가(*<date/datetime>*, *<num_years>*)	***date***에 연도 수를 더합니다. 간결한 구문 형식을 사용하는 경우, 입력 ***date***에 연도를 더하려면 해당 숫자가 양의 정수여야 합니다.
평일수(*<date1>*, *<date2>*)	***date*** 및 ***date*** 사이의 평일 수를 셉니다. 즉, 월요일 및 금요일 사이의 일수입니다. 참고: 이른 ***date***은(는) 포함되며 늦은 ***date***은(는) 제외됩니다.
연도시작(*<date/datetime>*)	특정 ***date***이(가) 포함되는 연도의 첫 ***date***을(를) 반환합니다.
연도끝(*<date/datetime>*)	특정 ***date***이(가) 포함되는 연도의 마지막 ***date***을(를) 반환합니다.
일차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 전체 일수를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계포함일차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 전체 일수(포함 방식)를 반환합니다. 이 계산에서는 양끝 날짜를 포함합니다. 두 날짜가 같은 경우 결과는 1입니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계제외일차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 전체 일수(제외 방식)를 반환합니다. 이 계산에서는 양끝 날짜를 제외합니다. 두 날짜가 같은 경우 결과는 0입니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.

구문	설명
주차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 주 수를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계포함주차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 주 수(포함 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계제외주차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 주 수(제외 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
월차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 개월 수를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계포함월차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 개월 수(포함 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계제외월차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 경과된 전체 개월 수(제외 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
연도차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 연도 수를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계포함연도차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 연도 수(포함 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.
경계제외연도차이(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** 및 ***date/datetime2*** 사이의 연도 수(제외 방식)를 반환합니다. 두 날짜의 순서는 결과에 영향을 미치지 않습니다.

시간 함수(English)

구문	설명
시간(*<text>*)	지정된 문자열을 시간으로 변환합니다.
초추출(*<time/datetime>*)	***timeofday/datetime attribute***의 초 구성 요소를 반환합니다.
분추출(*<time/datetime>*)	***timeofday/datetime attribute***의 분 구성 요소를 반환합니다.
시추출(*<time/datetime>*)	***timeofday/datetime attribute***의 시간 구성 요소를 반환합니다.

날짜 및 시간 함수(English)

구문	설명
현재날짜시간()	세션 시작 시 현재 ***date*** 및 시간을 반환합니다.
날짜시간(<*text*>)	지정된 문자열을 ***datetime*** 값으로 변환합니다.
날짜시간연결(<*date*>, <*time*>)	***date***과(와) 시간을 같이 결합하여 ***date*** 시간을 설정합니다.
초차이(<*datetime1*>, <*datetime2*>) 초차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 초 수를 반환합니다.
경계포함초차이(<*datetime1*>, <*datetime2*>) 경계포함초차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 초 수(포함 방식)를 반환합니다.
경계제외초차이(<*datetime1*>, <*datetime2*>) 경계제외초차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 초 수(제외 방식)를 반환합니다.
분차이(<*datetime1*>, <*datetime2*>) 분차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 분 수를 반환합니다.
경계포함분차이(<*datetime1*>, <*datetime2*>) 경계포함분차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 분 수(포함 방식)를 반환합니다.
경계제외분차이(<*datetime1*>, <*datetime2*>) 경계제외분차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 분 수(제외 방식)를 반환합니다.
시차이(<*datetime1*>, <*datetime2*>) 시차이(<*timeOfDay1*>, <*timeOfDay2*>)	***datetime1*** 및 ***datetime2*** 사이의 시간 수를 반환합니다.
경계포함시차이(<*dat-*	***datetime1*** 및 ***datetime2*** 사이의 시간 수(포함 방식)를 반환합니다.

구문	설명
etime1>, *<datetime2>*) **경계포함시차이**(*<timeOfDay1>*, *<timeOfDay2>*)	
경계제외시차이(*<datetime1>*, *<datetime2>*) **경계제외시차이**(*<timeOfDay1>*, *<timeOfDay2>*)	***datetime1*** 및 ***datetime2*** 사이의 시간 수(제외 방식)를 반환합니다.
날짜추출(*<datetime>*)	***datetime attribute***에서 ***date***을(를) 추출합니다.
시간추출(*<datetime>*)	***datetime attribute***에서 시간을 추출합니다. 현재 ***date*** 및 시간에서 시간을 추출하여 ***timeofday attribute***의 값을 규칙을 실행한 시간으로 설정하는 데 사용할 수 있습니다.
시추가(*<datetime>*, *<num_hours>*) **시추가**(*<timeOfDay>*, *<num_hours>*)	***date*** 시간에 시간 수를 더합니다.
분추가(*<datetime>*, *<num_minutes>*) **분추가**(*<timeOfDay>*, *<num_minutes>*)	***date*** 시간에 분 수를 더합니다.
초추가(*<datetime>*, *<num_seconds>*) **초추가**(*<timeOfDay>*, *<num_seconds>*)	***date*** 시간에 초 수를 더합니다.

텍스트 함수(English)

구문	설명
<text1> **&** *<text2>*	***text1***과(와) ***text2***을(를) 결합하여 단일 ***text*** 값을 생성합니다. 참고: 모든 유형의 변수를 사용할 수 있습니다. 값은 규칙 세션에서 설치된 포맷터를 사용하여 형식이 설정됩니다.
	text1과(와) ***text2***을(를) 결합하여 단일 ***text*** 값을 생성합니다. 참고: 모든 유형의 변수를 사용할 수 있습니다. 값은 규칙 세션에서 설치된 포맷터를 사용하여 형식이 설정됩니다.
포함(*<text>*, *<substring>*)	지정된 ***text*** 값이 지정된 ***text*** 하위 문자열을 포함하는지 여부를 나타내는 부울 값을 반환합니다. ***text*** 비교는 대소문자를 구분하지 않습니다.
끝문자(*<text>*, *<sub-*	지정된 ***text*** 값이 지정된 ***text*** 하위 문자열로 끝나는지 여부를 나타내는 부울 값을 반환

구문	설명
string>)	합니다. ***text*** 비교는 대소문자를 구분하지 않습니다.
유효수치(*<text>*)	지정된 ***text*** 값이 적합한 숫자를 나타내는지 여부를 표시하는 부울 값을 반환합니다.
길이(*<text>*)	지정된 ***text*** 값의 문자 길이를 반환합니다.
시작문자(*<text>*, *<sub-string>*)	지정된 ***text*** 값이 지정된 ***text*** 하위 문자열로 시작하는지 여부를 나타내는 부울 값을 반환합니다. ***text*** 비교는 대소문자를 구분하지 않습니다.
부분문자열(*<text>*, *<offset>*, *<length>*)	지정된 오프셋에서 시작하여 지정된 문자 길이만큼 ***text***의 하위문자열을 반환합니다. 문자열 끝에 도달하는 경우 보다 적은 문자가 반환됩니다.
텍스트(*<number>*) 텍스트(*<date>*) 텍스트(*<datetime>*) 텍스트(*<timeOfDay>*)	지정된 숫자 또는 ***date attribute***을(를) ***text*** 값으로 변환합니다.

개체 및 관계 함수(English)

구문	설명
대상(*<relationship>*, *<Exp>*)	조건이 하나뿐인 "일대일", "다대일" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다.
대상범위(*<relationship>*, *<alias>*) 대상범위(*<relationship>*)	조건이 하나 이상인 "일대일", "다대일" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다.
대상전체(*<relationship>*, *<Exp>*)	대상 ***entity*** 그룹의 모든 멤버가 규칙을 충족할 필요가 있는지 여부를 판단할 필요가 있을 때, "일대다" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다. 이 형식은 규칙에 조건이 하나만 있을 때 사용됩니다.
대상전체범위(*<relationship>*) 대상전체범위(*<relationship>*, *<alias>*)	대상 ***entity*** 그룹의 모든 멤버가 규칙을 충족할 필요가 있는지 여부를 판단할 필요가 있을 때, "일대다" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다. 이 형식은 규칙에 조건이 하나 이상 있을 때 사용됩니다.
존재(*<relationship>*, *<Exp>*)	대상 ***entity*** 그룹의 어떠한 멤버가 규칙을 충족할 필요가 있는지 여부를 판단할 필요가 있을 때, "일대다" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다. 이 형식은 규칙에 조건이 하나만 있을 때 사용됩니다.
존재범위(*<relationship>*) 존재범위(*<relationship>*, *<alias>*)	대상 ***entity*** 그룹의 어떠한 멤버가 규칙을 충족할 필요가 있는지 여부를 판단할 필요가 있을 때, "일대다" 또는 "다대다" ***relationship***의 하나의 ***entity***에서 다른 ***entity***(으)로 참조하는 데 사용됩니다. 이 형식은 규칙에 조건이 하나 이상 있을 때 사용됩니다.
멤버임(*<target>*, *<relationship>*)	***entity*** 인스턴스가 ***relationship***의 멤버라는 것을 추론하는 결론으

구문	설명
멤버임(*<target>*, *<alias>*, *<relationship>*)	로 사용됩니다. 특정 ***entity*** 인스턴스에 대해 두 번째 ***entity*** 인스턴스가 소스인 ***relationship***의 대상인지 여부를 테스트하는 조건으로 사용됩니다.
멤버가아님(*<target>*, *<relationship>*)	특정 ***entity*** 인스턴스에 대해 두 번째 ***entity*** 인스턴스가 소스인 ***relationship***의 대상이 아닌지 여부를 테스트하는 조건으로 사용됩니다.
인스턴스수(*<relationship>*)	***entity***에 존재하는 인스턴스의 수를 셉니다.
인스턴스수조건부(*<relationship>*, *<Exp>*)	특정 ***entity-level attribute***이(가) 특정 값을 가지는 ***entity***에 존재하는 인스턴스의 수를 셉니다.
인스턴스최대(*<relationship>*, *<number-attr>*) 인스턴스최대(*<relationship>*, *<date-attr>*) 인스턴스최대(*<relationship>*, *<datetime-attr>*) 인스턴스최대(*<relationship>*, *<time-attr>*)	***entity***의 모든 인스턴스에 대한 ***entity-level*** 변수의 가장 높거나 가장 최근의 값을 구합니다.
인스턴스최대조건부(*<relationship>*, *<number-attr>*, *<condition>*) 인스턴스최대조건부(*<relationship>*, *<date-attr>*, *<condition>*) 인스턴스최대조건부(*<relationship>*, *<datetime-attr>*, *<condition>*) 인스턴스최대조건부(*<relationship>*, *<time-attr>*, *<condition>*)	특정 ***entity-level attribute***이(가) 특정 값을 가지는 ***entity***의 모든 인스턴스에 대한 ***entity-level*** 변수의 가장 높거나 가장 최근의 값을 구합니다.
인스턴스최소(*<relationship>*, *<number-attr>*) 인스턴스최소(*<relationship>*, *<date-attr>*) 인스턴스최소(*<relationship>*, *<datetime-attr>*) 인스턴스최소(*<relationship>*, *<time-attr>*)	***entity***의 모든 인스턴스에 대한 ***entity-level*** 변수의 가장 낮거나 가장 이전의 값을 구합니다.
인스턴스최소조건부(*<relationship>*, *<number-attr>*, *<condition>*) 인스턴스최소조건부(*<relationship>*, *<date-attr>*, *<condition>*) 인스턴스최소조건부(*<relationship>*, *<datetime-attr>*, *<condition>*) 인스턴스최소조건부(*<relationship>*, *<time-attr>*, *<condition>*)	특정 ***entity-level attribute***이(가) 특정 값을 가지는 ***entity***의 모든 인스턴스에 대한 ***entity-level*** 변수의 가장 낮거나 가장 이전의 값을 구합니다.
인스턴스합계(*<relationship>*, *<number-attr>*)	***entity-level*** 변수의 모든 인스턴스 합계를 구합니다.

구문	설명
인스턴스합계조건부(*<relationship>*, *<number-attr>*, *<condition>*)	***entity***에 대해 특정 ***entity-level*** 부울 ***attribute***이(가) 참인 ***entity-level*** 변수의 모든 인스턴스 합계를 구합니다.
인스턴스값조건부(*<relationship>*, *<number-attr>*, *<condition>*) 인스턴스값조건부(*<relationship>*, *<text-attr>*, *<condition>*) 인스턴스값조건부(*<relationship>*, *<date-attr>*, *<condition>*) 인스턴스값조건부(*<relationship>*, *<datetime-attr>*, *<condition>*) 인스턴스값조건부(*<relationship>*, *<time-attr>*, *<condition>*)	하나의 조건으로 ***relationship***의 대상 ***entity*** 인스턴스를 식별하여 고유한 ***entity*** 인스턴스로부터 값을 구합니다. • 해당 조건이 단일 대상 ***entity*** 인스턴스를 식별하는 경우, 해당 ***entity*** 인스턴스에 대해 계산된 값이 결과 값이 됩니다. • 해당 조건을 만족하는 대상 인스턴스가 두 개 이상인 경우, ***uncertain***이(가) 반환됩니다. • 해당 조건을 만족하는 대상 인스턴스가 없고 ***relationship***이(가) 알려진 경우, 값은 ***uncertain***이(가) 됩니다.
인스턴스같음(*<instance1>*, *<instance2>*)	***entity***의 두 인스턴스가 동일한 인스턴스인지를 판단합니다.
인스턴스같지않음(*<instance1>*, *<instance2>*)	***entity***의 두 인스턴스가 동일한 인스턴스가 아닌지를 판단합니다.
추론인스턴스(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** (이 존재합니다 | 가 존재합니다 | 이 존재한다 | 가 존재한다 | 이존재합니다 | 가존재합니다 | 이존재한다 | 가존재한다)	***entity*** 인스턴스가 있으며 ***relationship***의 멤버라는 것을 추론하는 결론으로 사용됩니다.

시간 기준 추론 기능(English)

구문	설명
일정내고유값계산(*<start-date>*, *<end-date>*, *<variable>*) 일정내고유값계산(*<start-date>*, *<end-date>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 변수의 알려진 고유 값의 수를 셉니다.
조건부고유값계산(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 필터가 참인 시간만을 포함하는 변수의 알려진 고유 값의 수를 셉니다.
일정내일일합계(*<start-date>*,	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 통화 또는 숫자 변수의 합계를 계산합니다. ***attribute***은(는) 일별 값으로 간주됩니다.

구문	설명
<end-date>, *<number-attr>*)	
조건부일일합계 (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 조건이 참인 시간만을 포함하는 통화 또는 숫자 변수의 모든 일별 값의 합계를 계산합니다.
일정내최대값 (*<start-date>*, *<end-date>*, *<number-attr>*) **일정내최대값** (*<start-date>*, *<end-date>*, *<date-attr>*) **일정내최대값** (*<start-date>*, *<end-date>*, *<datetime-attr>*) **일정내최대값** (*<start-date>*, *<end-date>*, *<time-attr>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 변수의 최대 값을 선택합니다.
조건부최대값 (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **조건부최대값** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **조건부최대값** (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **조건부최대값** (*<start-date>*, *<end-date>*, *<time-attr>*, *<con-*	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 조건이 참인 시간만을 포함하는 변수의 최대 값을 선택합니다.

구문	설명
dition>)	
일정내최소값(*<start-date>*, *<end-date>*, *<number-attr>*) **일정내최소값**(*<start-date>*, *<end-date>*, *<date-attr>*) **일정내최소값**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **일정내최소값**(*<start-date>*, *<end-date>*, *<time-attr>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 변수의 최소 값을 선택합니다.
조건부최소값(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **조건부최소값**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **조건부최소값**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **조건부최소값**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 조건이 참인 시간만을 포함하는 변수의 최소 값을 선택합니다.
일정가중평균(*<start-date>*, *<end-date>*, *<number-attribute>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 각 값이 적용되는 시간 범위에 가중치를 적용하여 통화 또는 숫자 변수의 평균 값을 계산합니다.

구문	설명
조건부가중평균(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 조건이 참인 시간만을 포함하는 통화 또는 숫자 변수의 평균 값을 계산합니다(각 값이 적용되는 시간 범위에 가중치를 적용하며 필터가 참이어야 함).
일정간격만족(*<start-date>*, *<end-date>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 조건이 항상 참인 경우에만 참을 반환합니다.
일정내포함(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 조건이 지정된 일수 이상(연속될 필요는 없음) 참인 경우에만 참을 반환합니다.
연속일정포함(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 조건이 지정된 연속 일수 이상 참인 경우에만 참을 반환합니다.
일정내값존재(*<start-date>*, *<end-date>*, *<condition>*)	시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지의 간격에 대해 부울 조건이 한 번이라도 참인 경우에만 참을 반환합니다.
값시점(*<date>*, *<value>*)	지정된 ***date***에 지정된 ***attribute*** 값을 반환합니다.
최근참값(*<date>*, *<condition>*)	지정된 ***date***을(를) 포함하여 그 이전으로 거슬러 올라가며 확인하여, 부울 조건이 마지막으로 참인 ***date***을(를) 반환합니다.
다음참값(*<date>*, *<condition>*)	지정된 ***date***을(를) 포함하여 그 이후를 시간 흐름에 따라 확인하여, 부울 조건이 다음으로 참인 ***date***을(를) 반환합니다.
최근값()	가능한 가장 늦은 ***date***과(와) 동등한 ***date*** 값을 반환합니다. 이는 ***date attribute***이(가) 가질 수 있거나 표현식이 계산 가능한 다른 모든 ***date***보다 늦다는 것이 보장된 ***date***입니다.
최초값()	가능한 가장 이른 ***date***과(와) 동등한 ***date*** 값을 반환합니다. 이는 ***date attribute***이(가) 가질 수 있거나 표현식이 계산 가능한 다른 모든 ***date***보다 이르다는 것이 보장된 ***date***입니다.
기준이후충족일수(*<date>*, *<end-date>*)	매일 변화하며 ***date*** 이후 전체 일수인 숫자 변수를 반환합니다.
기준시작주	매주 변화하며 ***date*** 이후 전체 주 수인 숫자 변수를 반환합니다.

구문	설명
(*<date>*, *<end-date>*)	
기준시작월 (*<date>*, *<end-date>*)	매월 변화하며 ***date*** 이후 전체 개월 수인 숫자 변수를 반환합니다. 참고: 입력된 ***date***이(가) 해당 월의 28일 이후이며, 그 다음 월의 일수가 입력된 월보다 작은 경우, 기념 월의 변경 시점은 그 월의 마지막 날에 생성됩니다. 예를 들어, 입력된 ***date***이(가) 2007년 1월 28일, 29일, 30일 또는 31일이면 첫 변경 시점은 2007년 2월 28일이 됩니다.
기준시작연도 (*<date>*, *<end-date>*)	매년 변화하며 ***date*** 이후 전체 연도 수인 숫자 변수를 반환합니다.
지속충족값 (*<days>*, *<condition>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 지정된 선행 일수(현재 일 제외) 동안 부울 조건이 항상 참일 때만 참이 됩니다.
연속충족값 (*<minDays>*, *<days>*, *<condition>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 선행 설정 일수(현재 일 제외) 내에서 언제든지 최소 연속 일수 이상 부울 조건이 참일 경우에만 참이 됩니다.
간헐기준적용 (*<days>*, *<condition>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 지정된 선행 일수(현재 일 제외) 동안 부울 조건이 한 번이라도 참일 때만 참이 됩니다.
기준이후충족값 (*<date>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 특정 ***date*** 이후는 참이고, 해당 일 및 이전은 거짓이 됩니다.
기준이전충족값 (*<date>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 특정 ***date*** 이전은 참이고, 해당 일 및 이후는 거짓이 됩니다.
기준적용시 (*<date>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 특정 ***date***이면 참이고, 그 이전 및 이후는 거짓이 됩니다.
기준포함이후충족값(*<date>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 특정 ***date*** 및 이후는 참이고, 이전은 거짓이 됩니다.
기준포함이전충족값(*<date>*)	시간에 따라 변화하는 부울 ***attribute***을(를) 반환합니다. 특정 ***date*** 및 이전은 참이고, 이후는 거짓이 됩니다.
기준시작일(*<relationship>*, *<date>*, *<value>*)	***relationship***의 단일 시간 기준 ***attribute***(소스 ***entity*** 레벨) 및 개체의 값 ***attribute***을(를) 시작 ***date attribute***(으)로부터 유효한 값과 함께 반환합니다.
기준종료일(*<relationship>*, *<date>*, *<value>*)	***relationship***의 단일 시간 기준 ***attribute***(소스 ***entity*** 레벨) 및 개체의 값 ***attribute***을(를) 종료 ***date attribute***까지 유효한 값과 함께 반환합니다.
기준범위(*<rela-*	***relationship***의 단일 시간 기준 ***attribute***(소스 ***entity*** 레벨) 및 개체의 값 ***attribute***을(를) 시작

구문	설명
tionship>, <startdate>, <enddate>, <Value>)	***date attribute***(포함 방식)에서 종료 ***date attribute***(제외 방식)까지 유효한 값과 함께 반환합니다. 다음 시작 ***date*** 이전에 만료되는 경우 값은 ***uncertain***이(가) 됩니다.
기준평일 (*<startdate>, <enddate>*)	지정된 시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지 날짜가 평일이면 참을, 날짜가 주말이면 거짓을 반환합니다. ***date*** 범위를 벗어나는 경우 ***uncertain***을(를) 반환합니다.
기준(*<startdate>, <enddate>, <dayofmonth>*)	지정된 시작 ***date***(포함 방식)에서 종료 ***date***(제외 방식)까지 일이 월중 특정일 매개변수와 같으면 참을, 그 외 모든 월중 특정일이면 거짓을 반환합니다. ***date*** 범위를 벗어나는 경우 ***uncertain***을(를) 반환합니다. 월중 특정일이 현재 월의 일수를 초과하는 경우 그 월의 마지막 날에 값이 참이 됩니다. 따라서 함수가 월별로 정확히 하루에 대해 참 값을 반환합니다.

검증 이벤트 함수(English)

구문	설명
오류 (*<text>*)	사용자에게 메시지를 전달하기 위해 오류 이벤트가 사용되며 사용자는 오류를 트리거한 조건이 더 이상 적용되지 않을 때까지 조사를 계속할 수 없습니다.
경고 (*<text>*)	사용자에게 메시지를 전달하기 위해 경고 이벤트가 사용되지만 사용자는 경고를 트리거한 조건이 있어도 계속할 수 있습니다.

사용되지 않는 함수(English)

구문	설명
사용자지정기능 호출(*<A>, <B>*)	외부 호출의 결과를 코드 라이브러리에 반환합니다. 사용자 지정 함수 호출에 성공하기 위해서는 코드 라이브러리가 Determinations Engine에 제공되어야 합니다.

Logiske koblinger(English)

Syntaks	Beskrivelse
hvis	Valgfri betingelse som kan forekomme på slutten av en konklusjonslinje som har et påfølgende bevis
og	Logisk konjunksjon mellom to ***attributes***
eller	Logisk disjunksjon mellom to ***attributes***
enten **en av**	Grupperingselement som brukes for disjunksjoner der to eller flere ***attributes*** må grupperes

Syntaks	Beskrivelse
any **at least one of the following is true** **any of the following are satisfied**	
begge **alle** **all of the following are true** **all of the following are satisfied**	Grupperingselement som brukes for konjunksjoner der to eller flere ***attributes*** må grupperes
ellers	Betingelse som vises på slutten av en tabellregel for å angi ellers-leddet
er	Betingelse som brukes i en forklaringsoppføring mellom forkortet uttrykk og fullstendig ***attribute text***

Logiske funksjoner(English)

Syntaks	Beskrivelse
det er ikke sant at *<attr>*	Operator som brukes til å returnere sann hvis ***attribute*** har en verdi som er usann
<var> **er sikkert** **det er usikkert om** *<attr>* **det er sikkert hvorvidt** *<attr>*	Operator som brukes til å returnere sann hvis ***attribute*** har en verdi som ikke er ***uncertain***
<var> **er usikkert** *<var>* **er ikke sikkert** **det er usikkert at** *<attr>* **det er usikkert om** *<attr>* **det er usikkert hvorvidt** *<attr>* **det er ikke sikkert at** *<attr>* **usikker**	Operator som brukes til å returnere sann hvis ***attribute***-verdien er ***uncertain***
<var> **er kjent** **det er kjent hvorvidt** *<attr>*	Operator som brukes til å returnere sann hvis ***attribute*** har enhver verdi
<var> **er ukjent** **det er ukjent om** *<attr>* **det er ukjent hvorvidt** *<attr>* **ukjent**	Operator som brukes til å returnere sann hvis ***attribute*** ikke har en verdi

Logiske konstanter(English)

Syntaks	Beskrivelse
sann	Konstant sann verdi som brukes for tabellregler.
usann	Konstant usann verdi som brukes for tabellregler.
usikker	Konstant ***uncertain***-verdi som brukes for tabellregler.

Sammenligningsoperatorer(English)

Syntaks	Beskrivelse
*<lhs>***<***<rhs>* *<lhs>* **er tidligere enn** *<rhs>*	Mindre enn Merknad: Det finnes ingen naturlig språkform der denne operatoren brukes med tall- og valutaverdier.
<lhs> **>** *<rhs>* *<lhs>* **er senere enn** *<rhs>*	Større enn Merknad: Det finnes ingen naturlig språkform der denne operatoren brukes med tall- og valutaverdier.
*<lhs>***<=***<rhs>* *<lhs>* **er mindre enn eller lik** *<rhs>*	Mindre enn eller lik
<lhs> **>=** *<rhs>* *<lhs>* **er større enn eller lik** *<rhs>*	Større enn eller lik
*<lhs>***=***<rhs>* *<lhs>* **er lik** *<rhs>* *<lhs>* **equals** *<rhs>*	Er lik
<lhs> **er ikke lik** *<rhs>* *<lhs>* **<>** *<rhs>*	Ikke lik

Numeriske funksjoner(English)

Syntaks	Beskrivelse
Number(*<numText>*)	Konverter den angitte strengen til en tallverdi
<lhs> **+** *<rhs>*	Matematisk addisjon
<lhs> **-** *<rhs>*	Matematisk subtraksjon
<lhs> * *<rhs>*	Matematisk multiplikasjon
<lhs> **/** *<rhs>*	Matematisk divisjon
<lhs> \ *<rhs>*	Heltallsdivisjon

Syntaks	Beskrivelse
<lhs> **modulo** *<rhs>*	Gjenstående etter heltallsdivisjon
Maximum(*<x>*, *<y>*) **Maximum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **det høyeste av** *<val1>* **og** *<val2>* **den siste av** *<val1>* **og** *<val2>*	Returnerer den største av to verdier
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) **utleier av** *<val1>* **og** *<val2>* **det tidligste av** *<val1>* **og** *<val2>*	Returnerer den minste av to verdier
Xy(*<x>*, *<y>*)	x i y. potens
Ex(*<x>*)	Konstant e i x. potens
Abs(*<x>*)	Absolutt verdi av x
Ln(*<x>*)	Naturlig logaritme for x
Log(*<x>*)	Logaritmebase 10 for x
Sqrt(*<x>*) **kvadratroten av** *<val>*	Kvadratrot av x
Round(*<x>*, *<n>*) *<val>* **avrundet til** *<num_places>* **desimal** *<val>* **avrundet til** *<num_places>* **desimaler**	Avrunder x til n desimaler
Trunc(*<x>*, *<n>*)	x avkortet til n desimaler
Sin(*<x>*)	Sinus av x
Cos(*<x>*)	Cosinus av x
Tan(*<x>*)	Tangens av x
Asin(*<x>*)	Arcsinus av x
Acos(*<x>*)	Arccosinus av x
Atan(*<x>*)	Arctangens av x

Datofunksjoner(English)

Syntaks	Beskrivelse
CurrentDate() **gjeldende dato**	Returnerer gjeldende ***date*** ved oppstart av økten.

Syntaks	Beskrivelse
Date(*<text>*)	Konverterer den angitte strengen til en ***date***-verdi
MakeDate(*<year>*, *<month>*, *<day>*)	Returnerer ***date*** dannet på grunnlag av angitt år, måned og dag.
ExtractDay(*<date/datetime>*)	Returnerer dagskomponenten i ***date/datetime attribute***.
ExtractMonth(*<date/datetime>*)	Returnerer månedskomponenten i ***date/datetime attribute***.
ExtractYear(*<date/datetime>*)	Returnerer årskomponenten i ***date/datetime attribute***.
NextDayOfTheWeek (*<date/datetime>*, *<day>*) **neste mandag på eller etter** *<from-date>* **neste tirsdag på eller etter** *<from-date>* **neste onsdag på eller etter** *<from-date>* **neste torsdag på eller etter** *<from-date>* **neste fredag på eller etter** *<from-date>* **neste lørdag på eller etter** *<from-date>* **neste søndag på eller etter** *<from-date>*	Returnerer ***date*** for neste ukedag på eller før/etter ***date*** (avhengig av syntaksen som brukes).
NextDate(*<date>*, *<day>*, *<month>*)	Returnerer den neste forekomsten av angitt dag og måned etter ***date***.
AddDays(*<date/datetime>*, *<num_days>*) **datoen** *<num_days>* **dager etter** *<datetime>* **tiden** *<num_days>* **dager etter** *<datetime>*	Legger til / trekker fra et antall dager for ***date***. Når den korte syntaktiske formen brukes, må antallet være et positivt heltall hvis du vil legge dager til inndataene ***date***, eller et negativt tall hvis du vil trekke dager fra inndataene ***date***.
AddWeeks (*<date/datetime>*, *<num_weeks>*) **datoen** *<num_weeks>* **uker etter** *<datetime>* **tiden** *<num_weeks>* **uker**	Legger til et antall uker for ***date***. Når den korte syntaktiske formen brukes, må antallet være et positivt heltall hvis du vil legge uker til inndataene ***date***.

Syntaks	Beskrivelse
etter *<datetime>*	
AddMonths(*<date/datetime>*, *<num_months>*) **datoen** *<num_months>* **måneder etter at** *<datetime>* **tiden** *<num_months>* **måneder etter** *<datetime>*	Legger til et antall måneder for ***date***. Når den korte syntaktiske formen brukes, må antallet være et positivt heltall hvis du vil legge måneder til inndataene ***date***.
AddYears(*<date/datetime>*, *<num_years>*) **datoen** *<num_years>* **år etter at** *<datetime>* **tiden** *<num_years>* **år etter** *<datetime>*	Legger til et antall år for ***date***. Når den korte syntaktiske formen brukes, må antallet være et positivt heltall hvis du vil legge år til inndataene ***date***.
WeekdayCount(*<date1>*, *<date2>*) **antall ukedager (inkludert) mellom** *<date1>* **og** *<date2>*	Teller antall ukedager mellom ***date***1 og ***date***2, dvs. antall dager mellom mandag og fredag. Merknad: Første ***date*** er inkludert og siste ***date*** er utelatt.
YearStart(*<date/datetime>*) **den første dagen i året hvor** *<from-date>* **faller**	Returnerer første ***date*** i året der ***date*** ligger.
YearEnd(*<date/datetime>*) **den siste dagen i året hvor** *<from-date>* **faller**	Returnerer siste ***date*** i året der ***date*** ligger.
DayDifference(*<date/datetime1>*, *<date/datetime2>*) **antall dager fra** *<date1>* **til** *<date2>*	Returnerer antall hele dager mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
DayDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **antall dager (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer antall hele dager (inkluderende) mellom ***date/datetime1*** og ***date/datetime2***. Begge sluttpunktene inngår i denne beregningen. Når datoene er de samme, blir resultatet 1. Rekkefølgen på de to datoene påvirker ikke resultatet.
DayDifferenceExclusive (*<date/datetime1>*, *<date/d-*	Returnerer antall hele dager (ekskluderende) mellom ***date/datetime1*** og ***date/datetime2***. Begge sluttpunktene utelates fra denne beregningen. Når datoene er de

Syntaks	Beskrivelse
atetime2>) **antall dager (eksklusive) fra** *<date1>* **til** *<date2>*	samme, blir resultatet 0. Rekkefølgen på de to datoene påvirker ikke resultatet.
WeekDifference(*<date/datetime1>*, *<date/datetime2>*) **antall uker fra** *<date1>* **til** *<date2>*	Returnerer antall hele medgåtte uker mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
WeekDifferenceInclusive(*<date/datetime1>*, *<date/datetime2>*) **antall uker (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer inkluderende antall hele medgåtte uker mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
WeekDifferenceExclusive(*<date/datetime1>*, *<date/datetime2>*) **antall uker (eksklusive) fra** *<date1>* **til** *<date2>*	Returnerer ekskluderende antall hele medgåtte uker mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
MonthDifference(*<date/datetime1>*, *<date/datetime2>*) **antall måneder fra** *<date1>* **til** *<date2>*	Returnerer antall hele medgåtte måneder mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
MonthDifferenceInclusive(*<date/datetime1>*, *<date/datetime2>*) **antall måneder (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer antall hele inkluderende medgåtte måneder mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
MonthDifferenceExclusive(*<date/datetime1>*, *<date/datetime2>*) **antall måneder (eksklusive) fra** *<date1>* **til** *<date2>*	Returnerer antall hele ekskluderende medgåtte måneder mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
YearDifference(*<date/datetime1>*, *<date/datetime2>*) **antall hele år som**	Returnerer antall år mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.

Syntaks	Beskrivelse
<date2> **er etter** *<date1>* **antall år fra** *<date1>* **og** *<date2>*	
YearDifferenceInclusive (*<date/datetime1>*, *<date/datetime2>*) **antall år (inkluderende) fra** *<date1>* **og** *<date2>*	Returnerer inkluderende antall år mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.
YearDifferenceExclusive (*<date/datetime1>*, *<date/datetime2>*) **antall år (eksklusive) fra** *<date1>* **og** *<date2>*	Returnerer ekskluderende antall år mellom ***date/datetime1*** og ***date/datetime2***. Rekkefølgen på de to datoene påvirker ikke resultatet.

Klokkeslettfunksjoner(English)

Syntaks	Beskrivelse
TimeOfDay(*<text>*)	Konverterer den angitte strengen til et tidspunkt på dagen
ExtractSecond(*<time/datetime>*)	Returnerer sekundkomponenten i ***timeofday/datetime attribute***.
ExtractMinute(*<time/datetime>*)	Returnerer minuttkomponenten i ***timeofday/datetime attribute***.
ExtractHour(*<time/datetime>*)	Returnerer timekomponenten i ***timeofday/datetime attribute***.

Dato- og tidsfunksjoner(English)

Syntaks	Beskrivelse
CurrentDateTime()	Returnerer gjeldende ***date*** og tidspunkt ved oppstart av økten.
DateTime(*<text>*)	Konverterer den angitte strengen til en ***datetime***-verdi
ConcatenateDateTime (*<date>*, *<time>*) *<date>* **kl** *<time-of-day>* *<time-of-day>* **på** *<date>*	Angir ***date***-tidspunktet ved å slå sammen ***date*** og tidspunkt på dagen.
SecondDifference(*<datetime1>*, *<datetime2>*) **SecondDifference** (*<timeOfDay1>*, *<timeOfDay2>*) **antall sekunder fra**	Returnerer antall sekunder mellom ***datetime1*** og ***datetime2***.

Syntaks	Beskrivelse
<date1> **til** *<date2>*	
SecondDifferenceInclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall sekunder (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer inkluderende antall sekunder mellom ***datetime1*** og ***datetime2***.
SecondDifferenceExclusive (*<datetime1>*, *<datetime2>*) **SecondDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall sekunder (eksklusive) fra** *<date1>* **til** *<date2>*	Returnerer ekskluderende antall sekunder mellom ***datetime1*** og ***datetime2***.
MinuteDifference(*<datetime1>*, *<datetime2>*) **MinuteDifference** (*<timeOfDay1>*, *<timeOfDay2>*) **antall minutter fra** *<date1>* **til** *<date2>*	Returnerer antall minutter mellom ***datetime1*** og ***datetime2***.
MinuteDifferenceInclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall minutter (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer inkluderende antall minutter mellom ***datetime1*** og ***datetime2***.
MinuteDifferenceExclusive (*<datetime1>*, *<datetime2>*) **MinuteDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall minutter (eksklusive) fra** *<date1>* **til** *<date2>*	Returnerer ekskluderende antall minutter mellom ***datetime1*** og ***datetime2***.
HourDifference(*<datetime1>*, *<datetime2>*)	Returnerer antall timer mellom ***datetime1*** og ***datetime2***.

Syntaks	Beskrivelse
HourDifference (*<timeOfDay1>*, *<timeOfDay2>*) **antall timer fra** *<date1>* **til** *<date2>*	
HourDifferenceInclusive (*<datetime1>*, *<datetime2>*) **HourDifferenceInclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall timer (inkluderende) fra** *<date1>* **til** *<date2>*	Returnerer inkluderende antall timer mellom ***datetime1*** og ***datetime2***.
HourDifferenceExclusive (*<datetime1>*, *<datetime2>*) **HourDifferenceExclusive** (*<timeOfDay1>*, *<timeOfDay2>*) **antall timer (eksklusive) fra** *<date1>* **til** *<date2>*	Returnerer ekskluderende antall timer mellom ***datetime1*** og ***datetime2***.
ExtractDate(*<datetime>*)	Trekker ut ***date*** fra ***datetime attribute***.
ExtractTimeOfDay(*<datetime>*)	Trekker ut tidspunkt på dagen fra ***datetime attribute***. Kan brukes til å angi utførelsestidspunktet for regelen som verdi for ***timeofday attribute*** ved å trekke ut tidspunktet fra gjeldende ***date*** og tidspunkt.
AddHours(*<datetime>*, *<num_hours>*) **AddHours**(*<timeOfDay>*, *<num_hours>*) **tiden** *<num_hours>* **timer etter** *<datetime>*	Legger et antall timer til et ***date***-tidspunkt.
AddMinutes(*<datetime>*, *<num_minutes>*) **AddMinutes**(*<timeOfDay>*, *<num_minutes>*) **tiden** *<num_minutes>* **minutter etter** *<datetime>*	Legger et antall minutter til et ***date***-tidspunkt.
AddSeconds(*<datetime>*, *<num_seconds>*) **AddSeconds**(*<timeOfDay>*, *<num_seconds>*)	Legger et antall sekunder til et ***date***-tidspunkt.

Syntaks	Beskrivelse
tiden *<num_seconds>* **sekunder etter** *<datetime>*	

Tekstfunksjoner(English)

Syntaks	Beskrivelse
<lhs> **&** *<rhs>*	Kombinerer ***text1*** med ***text2*** osv. for å danne én ***text***-verdi. Du kan bruke variabler av alle typer. Verdiene formateres ved hjelp av formatereren som er installert i regeløkten.
på sammensetning av *<text1>* **&** *<text2>*	Kombinerer ***text1*** med ***text2*** osv. for å danne én ***text***-verdi. Du kan bruke variabler av alle typer. Verdiene formateres ved hjelp av formatereren som er installert i regeløkten.
Contains(*<text>*, *<substring>*)	Returnerer en boolsk verdi som angir om den angitte ***text***-verdien inneholder den angitte ***text***-delstrengen. ***text***-sammenligningen skiller ikke mellom store og små bokstaver.
EndsWith(*<text>*, *<substring>*)	Returnerer en boolsk verdi som angir om den angitte ***text***-verdien slutter med den angitte ***text***-delstrengen. ***text***-sammenligningen skiller ikke mellom store og små bokstaver.
IsNumber(*<text>*)	Returnerer en boolsk verdi som angir om den angitte ***text***-verdien representerer et gyldig tall.
Length(*<text>*)	Returnerer tegnlengden for den angitte ***text***-verdien.
StartsWith(*<text>*, *<substring>*)	Returnerer en boolsk verdi som angir om den angitte ***text***-verdien starter med den angitte ***text***-delstrengen. ***text***-sammenligningen skiller ikke mellom store og små bokstaver.
Substring(*<text>*, *<offset>*, *<length>*)	Returnerer delstrengen for ***text*** som begynner på angitt forskyvning, dvs. den angitte tegnlengden. Færre tegn returneres hvis slutten på strengen nås.
Text(*<number>*) **Text**(*<date>*) **Text**(*<datetime>*) **Text**(*<timeOfDay>*)	Konverter angitt tall eller ***date attribute*** til en ***text***-verdi.

Entitets- og relasjonsfunksjoner(English)

Syntaks	Beskrivelse
For(*<relationship>*, *<Exp>*) **i tilfelle av** *<ent>*, *<attr>* *<val>*, **i tilfelle** *<ent>*	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til én, Mange til én eller Mange til mange der det bare er én betingelse.
ForScope(*<relationship>*, *<alias>*) **ForScope**(*<relationship>*)	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til én, Mange til én eller Mange til mange der det er en eller flere betingelser.

Syntaks	Beskrivelse
i tilfelle av *<ent>* **i tilfelle av** *<ent>* **(**<alias>**)**	
ForAll(*<relationship>*, *<Exp>*) **hvert av** *<ent-attr>* **for hvert av** *<ent>***,** *<attr>* **for hele** *<ent>***,** *<attr>*	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til mange eller Mange til mange når du har behov for å fastslå om alle medlemmer i ***entity***-målgruppen må oppfylle regelen. Dette skjermbildet brukes når det bare er én betingelse i regelen.
ForAllScope(*<relationship>*) **ForAllScope**(*<relationship>*, *<alias>*) **for alle** *<ent>* **hvert av** *<ent>* **for alle** *<ent>* **(**<alias>**)** **hvert av** *<ent>* **(**<alias>**)** **for hver av** *<ent>* **(**<alias>**)**	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til mange eller Mange til mange når du har behov for å fastslå om alle medlemmer i ***entity***-målgruppen må oppfylle regelen. Dette skjermbildet brukes når det er en eller flere betingelser i regelen.
Exists(*<relationship>*, *<Exp>*) **minst ett av** *<ent-attr>* **for minst ett av** *<ent>***,** *<attr>*	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til mange eller Mange til mange når du har behov for å fastslå om noen medlemmer i ***entity***-målgruppen må oppfylle regelen. Dette skjermbildet brukes når det bare er én betingelse i regelen.
ExistsScope(*<relationship>*) **ExistsScope**(*<relationship>*, *<alias>*) **minst ett av** *<ent>* **minst ett av** *<ent>* **(**<alias>**)** **for minst ett av** *<ent>* **(**<alias>**)**	Brukes til referanser fra én ***entity*** til en annen ***entity*** i en ***relationship*** av typen Én til mange eller Mange til mange når du har behov for å fastslå om noen medlemmer i ***entity***-målgruppen må oppfylle regelen. Dette skjermbildet brukes når det er en eller flere betingelser i regelen.
IsMemberOf(*<target>*, *<relationship>*) **IsMemberOf**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **er medlem av** *<ent>* <ent-target> **(**<alias>**) er medlem av** *<ent>*	Brukes som en konklusjon for å utlede at en ***entity***-forekomst er et medlem av en ***relationship***. Brukes som en betingelse for å teste at en ***entity***-forekomst er et mål for en ***relationship*** som har en annen ***entity***-forekomst som kilde.
IsNotMemberOf(*<target>*, *<relationship>*)	Brukes som en betingelse for å teste at en ***entity***-forekomst ikke er et mål for en ***relationship*** som har en annen ***entity***-forekomst som kilde.
InstanceCount (*<relationship>*) **antall** *<ent>*	Teller antall forekomster som finnes for ***entity***.
InstanceCountIf(*<rela-*	Teller antall forekomster som finnes av ***entity*** der bestemt ***entity-level attribute*** har en

Syntaks	Beskrivelse
tionship>, *<Exp>*) **antall** *<ent>* **for der er det slik at** *<condition>*	bestemt verdi.
InstanceMaximum(*<relationship>*, *<number-attr>*) **InstanceMaximum**(*<relationship>*, *<date-attr>*) **InstanceMaximum**(*<relationship>*, *<datetime-attr>*) **InstanceMaximum**(*<relationship>*, *<time-attr>*) *<date-attr>* **som er den siste for alle** *<ent>* *<max-attr>* **som er størst for alle** *<ent>* **den siste av alle** *<ent-attr>* **den siste av alle** *<attr>* **for** *<ent>* **den største av** [**alle**]*<ent-attr>* **den største av** [**all**]*<attr>* **for** [**alle**]*<ent>*	Henter den høyeste/seneste verdien for en ***entity-level***-variabel for alle forekomster av ***entity***.
InstanceMaximumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMaximumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **som er den siste for alle** *<ent>* **for der er det slik at** *<ent-test>* *<max-attr>* **som er det største for alle** *<ent>* **for der er det slik at** *<ent-test>* **den siste av alle** *<ent-attr>* **der det er slik at** *<ent-test>* **den største av alle** *<ent-attr>*	Henter den høyeste/seneste verdien for en ***entity-level***-variabel for alle forekomster av ***entity*** der bestemt ***entity-level attribute*** har en bestemt verdi.

Syntaks	Beskrivelse
der det er slik at *<ent-test>* **den største av** *<attr>* **for alle** *<ent>* **der det er slik at** *<ent-test>*	
InstanceMinimum(*<relationship>*, *<number-attr>*) **InstanceMinimum**(*<relationship>*, *<date-attr>*) **InstanceMinimum**(*<relationship>*, *<datetime-attr>*) **InstanceMinimum**(*<relationship>*, *<time-attr>*) *<date-attr>* **som er den tidligste for alle** *<ent>* *<attr>* **som er det minste for alle** *<ent>* **det tidligste av alle** *<ent-attr>* **det tidligste av alle** *<attr>* **for** *<ent>* **den minste av** [**alle**]*<ent-attr>* **den minste av** [**alle**]*<attr>* **for** [**alle**]*<ent>*	Henter den laveste/tidligste verdien for en ***entity-level***-variabel for alle forekomster av ***entity***.
InstanceMinimumIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstanceMinimumIf**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **som er den tidligste for alle** *<ent>* **for der er det slik at** *<ent-test>* *<num-attr>* **som er det minste for alle** *<ent>* **for der er det slik at** *<ent-test>*	Henter den laveste/tidligste verdien for en ***entity-level***-variabel for alle forekomster av ***entity*** der bestemt ***entity-level attribute*** har en bestemt verdi.

Syntaks	Beskrivelse
den minste av alle *<ent-attr>* **der det er slik at** *<ent-test>* **den minste av alle** *<attr>* **for** *<ent>* **som det er slik at** *<ent-test>* **det tidligste av alle** *<attr>* **for** *<ent>* **der det er slik at** *<ent-test>*	
InstanceSum(*<relationship>*, *<number-attr>*) *<num-attr>* **utgjorde for alle** *<ent>* **den totale mengden** [**av alle**]*<ent-attr>* **totalen for alle** *<ent-attr>* **totalt for alle** *<ent>*, *<attr>*	Henter summen av alle forekomster av en ***entity-level***-variabel.
InstanceSumIf (*<relationship>*, *<number-attr>*, *<condition>*) *<num-attr>* **utgjorde for alle** *<ent>* **for der er det slik at** *<ent-test>* **den totale mengden av alle** *<ent-attr>* **bare der** *<condition>* **den totale mengden av** [**alle**]*<ent-attr>* **som det er tilfelle at** *<condition>* **totalt for alle** *<ent>*, *<attr>* **bare der** *<condition>*	Henter summen av alle forekomster av en ***entity-level***-variabel der bestemt ***entity-level*** boolsk ***attribute*** er sann for ***entity***.
InstanceValueIf(*<relationship>*, *<number-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<text-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<date-attr>*, *<condition>*) **InstanceValueIf**(*<relationship>*, *<datetime-attr>*, *<condition>*)	Henter en verdi fra en unik ***entity***-forekomst, identifisert fra ***entity***-målforekomstene i ***relationship*** med en betingelse. • Hvis betingelsen identifiserer én ***entity***-målforekomst, er dette verdien som beregnes mot denne ***entity***-forekomsten. • Hvis mer enn én målforekomst oppfyller betingelsen, returneres ***uncertain***. • Hvis ingen målforekomster oppfyller betingelsen og ***relationship*** er kjent, er verdien ***uncertain***.

Syntaks	Beskrivelse
InstanceValueIf(*<relationship>*, *<time-attr>*, *<condition>*)	
InstanceEquals(*<instance1>*, *<instance2>*)	Fastslår om to forekomster av ***entity*** er den samme forekomsten.
InstanceNotEquals (*<instance1>*, *<instance2>*)	Fastslår om to forekomster av ***entity*** ikke er den samme forekomsten.
InferInstance(*<relationship>*, *<identity>*)	Brukes som en konklusjon for å utlede at det finnes en ***entity***-forekomst som er medlem av ***relationship***.

Funksjoner for tidsbestemt resonnering(English)

Syntaks	Beskrivelse
IntervalCountDistinct (*<start-date>*, *<end-date>*, *<variable>*) **IntervalCountDistinct** (*<start-date>*, *<end-date>*, *<condition>*)	Teller antall kjente unike verdier for variabelen, i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
IntervalCountDistinctIf (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Teller antall kjente unike verdier for variabelen, i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), bare inkludert tilfeller der et boolsk filter er sant.
IntervalDailySum(*<start-date>*, *<end-date>*, *<number-attr>*)	Beregner summen for en valuta- eller tallvariabel, i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt). ***attribute*** forutsettes å være en daglig mengde.
IntervalDailySumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Beregner summen av alle de daglige verdiene for en valuta- eller tallvariabel, i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), bare inkludert tilfeller der en betingelse er sann.
IntervalMaximum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMaximum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMaximum**(*<start-*	Velger maksimumsverdien for en variabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).

Syntaks	Beskrivelse
date>, *<end-date>*, *<time-attr>*)	
IntervalMaximumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMaximumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Velger maksimumsverdien for en variabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), bare inkludert tilfeller der en betingelse er sann.
IntervalMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervalMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Velger minimumsverdien for en variabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
IntervalMinimumIf(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervalMinimumIf**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Velger minimumsverdien for en variabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), bare inkludert tilfeller der en betingelse er sann.
IntervalWeightedAverage (*<start-date>*, *<end-date>*, *<number-attribute>*)	Beregner gjennomsnittsverdien for en valuta- eller tallvariabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), vektet med tidsrommet som hver verdi gjelder.

Syntaks	Beskrivelse
IntervalWeightedAverageIf (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Beregner gjennomsnittsverdien for en valuta- eller tallvariabel i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt), bare inkludert tilfeller der en boolsk betingelse er sann (vektet med tidsrommet som hver verdi gjelder og der filteret er sant).
IntervalAlways(*<start-date>*, *<end-date>*, *<condition>*)	Returnerer sann bare hvis en boolsk betingelse alltid er sann i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
IntervalAtLeastDays(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerer sann bare hvis en boolsk betingelse er sann for minst det angitte antall dager (ikke nødvendigvis sammenhengende) i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
IntervalConsecutiveDays (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerer sann bare hvis en boolsk betingelse er sann for minst et angitt antall sammenhengende dager i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
IntervalSometimes(*<start-date>*, *<end-date>*, *<condition>*)	Returnerer sann bare hvis en boolsk betingelse tidvis er sann i intervallet fra start ***date*** (inkludert) til slutt ***date*** (utelatt).
ValueAt(*<date>*, *<value>*)	Returnerer verdien for angitt ***attribute*** på angitt ***date***.
WhenLast(*<date>*, *<condition>*)	Returnerer ***date*** da en boolsk betingelse sist var sann, bakover fra (og inkludert) angitt ***date***.
WhenNext(*<date>*, *<condition>*)	Returnerer neste ***date*** da en boolsk betingelse vil være sann, fremover fra (og inkludert) angitt ***date***.
Latest()	Returnerer en ***date***-verdi som tilsvarer senest mulig ***date*** - nemlig en ***date*** som garantert vil være senere enn noen annen ***date*** som ***date attribute*** kan ta eller et uttrykk kan evalueres til.
Earliest()	Returnerer en ***date***-verdi som tilsvarer tidligst mulig ***date*** - nemlig en ***date*** som garantert vil være tidligere enn noen annen ***date*** som ***date attribute*** kan ta eller et uttrykk kan evalueres til.
TemporalDaysSince(*<date>*, *<end-date>*)	Returnerer en tallvariabel som varierer hver dag, og er lik antall hele dager siden ***date***.
TemporalWeeksSince (*<date>*, *<end-date>*)	Returnerer en tallvariabel som varierer hver uke, og er lik antall hele uker siden ***date***.
TemporalMonthsSince (*<date>*, *<end-date>*)	Returnerer en tallvariabel som varierer hver måned, og er lik antall hele måneder siden ***date***. Merknad: Når angitt ***date*** er etter den 28. dagen i måneden, og en påfølgende måned har færre dager enn den angitte måneden, opprettes endringspunktet for merkemåneden på den siste dagen i denne måneden. Hvis for eksempel angitt ***date*** er 28., 29., 30. eller 31. januar 2007, vil det første endringspunktet være 28. februar 2007.

Syntaks	Beskrivelse
TemporalYearsSince (*<date>*, *<end-date>*)	Returnerer en tallvariabel som varierer hvert år, og er lik antall hele år siden ***date***.
TemporalAlwaysDays (*<days>*, *<condition>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann bare hvis en boolsk betingelse er sann for alle av et angitt antall forutgående dager, ikke inkludert dagens dato.
TemporalConsecutiveDays (*<minDays>*, *<days>*, *<condition>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann bare hvis en boolsk betingelse er sann for minst et minste antall sammenhengende dager når som helst innenfor angitt antall forutgående dager, ikke inkludert dagens dato.
TemporalSometimesDays (*<days>*, *<condition>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann bare hvis en boolsk betingelse tidvis er sann innenfor et angitt antall forutgående dager, ikke inkludert dagens dato.
TemporalAfter(*<date>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann etter ***date*** og usann på og før.
TemporalBefore(*<date>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann før ***date*** og usann på og etter.
TemporalOn(*<date>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann på ***date*** og usann før og etter.
TemporalOnOrAfter (*<date>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann på eller etter ***date*** og usann før.
TemporalOnOrBefore (*<date>*)	Returnerer boolsk ***attribute*** som varierer over tid, og er sann på og før ***date*** og usann etter.
TemporalFromStartDate (*<relationship>*, *<date>*, *<value>*)	Returnerer én tidsbestemt ***attribute*** (på ***entity***-kildenivå) fra ***relationship*** og en verdi ***attribute*** for entitetene, med verdier som er gjeldende fra en start ***date attribute***.
TemporalFromEndDate (*<relationship>*, *<date>*, *<value>*)	Returnerer én tidsbestemt ***attribute*** (på ***entity***-kildenivå) fra ***relationship*** og en verdi ***attribute*** for entitetene, med verdier som er gjeldende inntil en slutt ***date attribute***.
TemporalFromRange(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Returnerer én tidsbestemt ***attribute*** (på ***entity***-kildenivå) fra ***relationship*** og en verdi ***attribute*** for entitetene, med verdier som er gjeldende fra start ***date attribute*** (inkludert) til slutt ***date attribute*** (utelatt). Verdien er ***uncertain*** hvis den utløper før neste start ***date***.
TemporalIsWeekday (*<startdate>*, *<enddate>*)	Returnerer sann for datoer som er ukedager og usann for datoer som er helgedager fra angitt start ***date*** (inkludert) til slutt ***date*** (utelatt). Returnerer ***uncertain*** utenfor ***date***-området.
TemporalOncePerMonth (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Returnerer sann hvis dagen er lik parameteren for dag i måneden og usann for alle andre dager i måneden fra angitt start ***date*** (inkludert) til slutt ***date*** (utelatt). Returnerer ***uncertain*** utenfor ***date***-området. Når dagen i måneden overskrider antall dager i inneværende måned, er verdien sann på den siste dagen i denne måneden, slik at funksjonen returnerer en verdi som er sann nøyaktig én dag per måned.

Funksjoner for valideringshendelser(English)

Syntaks	Beskrivelse
Feil(<*text*>)	En feilhendelse brukes til å gi brukeren en melding og hindre vedkommende i å fortsette en undersøkelse til betingelsen som utløste feilen, ikke lenger gjelder.
Advarsel (<*text*>)	En advarselshendelse brukes til å gi brukeren en melding, men tillater at vedkommende fortsetter til tross for betingelsen som utløste advarselen.

Frarådede funksjoner(English)

Syntaks	Beskrivelse
CallCustomFunction (<*A*>, <*B*>)	Returnerer resultatet av et eksternt kall til et kodebibliotek. Kodebiblioteket må angis for fastsettelsesmotoren for at det egendefinerte funksjonskallet skal kunne utføres.

Łączniki logiczne(English)

Składnia	Opis
jeśli	Opcjonalny termin, mogący występować na końcu linii wniosku, po której następuje dowód
i	Koniunkcja logiczna dwóch atrybutów ***attributes***
lub	Alternatywa logiczna dwóch atrybutów ***attributes***
którekolwiek **jedno z** **dowolne** **co najmniej jedno z poniższych stwierdzeń jest prawdziwe** **dowolny z poniższych warunków jest spełniony**	Element grupujący używany w przypadku alternatyw, w których należy zgrupować przynajmniej dwie wartości ***attributes***
oba **wszystkie** **wszystkie z poniższych stwierdzeń są prawdziwe** **wszystkie z poniższych warunków są spełnione**	Element grupujący używany w przypadku koniunkcji, w których należy zgrupować przynajmniej dwie wartości ***attributes***
w przeciwnym razie	Termin występujący na końcu reguły tabeli, oznaczający klauzulę "w przeciwnym razie"
jest	Termin używany w zapisie opisu między skróconym wyrażeniem a pełnym tekstem ***text*** atrybutu ***attribute***

Funkcje logiczne(English)

Składnia	Opis
nie jest prawdą, że *<attr>*	Operator używany do zwracania wartości "prawda", jeśli ***attribute*** ma wartość "fałsz"
<var> **jest pewny** **jest pewne, że** *<attr>*	Operator używany do zwracania wartości "prawda", jeśli ***attribute*** ma wartość inną niż ***uncertain***
<var> **jest niepewny** **nie jest pewne, że** *<attr>* **nie jest pewne, czy** *<attr>* **nie jest pewne, że** *<attr>* **niepewne**	Operator używany do zwracania wartości "prawda", jeśli wartość ***attribute*** to ***uncertain***
<var> **jest znany** **wiadomo, czy** *<attr>*	Operator używany do zwracania wartości "prawda", jeśli ***attribute*** ma dowolną wartość
<var> **jest nieznany** **nie wiadomo, czy** *<attr>* **nieznane**	Operator używany do zwracania wartości "prawda", jeśli ***attribute*** nie ma wartości

Stałe logiczne(English)

Składnia	Opis
prawda	Stała wartość "prawda" używana dla reguł tabeli.
fałsz	Stała wartość "fałsz" używana dla reguł tabeli.
niepewne	Stała wartość "***uncertain***" używana dla reguł tabeli.

Operatory porównania(English)

Składnia	Opis
*<x>***<***<y>*	Mniejsze niż Uwaga: Nie istnieje naturalna forma językowa, jeśli ten operator używany jest z wartościami liczbowymi i walutowymi.
<x> **>** *<y>*	Większe niż Uwaga: Nie istnieje naturalna forma językowa, jeśli ten operator używany jest z wartościami liczbowymi i walutowymi.
*<x>***<=***<y>*	Mniejsze lub równe
<x> **>=** *<y>*	Większe lub równe

Składnia	Opis
<x>=*<y>*	Równa się
<x> **<>** *<y>*	Nie równa się

Funkcje liczbowe(English)

Składnia	Opis
Liczba(*<numText>*)	Konwertuje określony napis na wartość liczbową
<x> **+** *<y>*	Dodawanie matematyczne
<x> **-** *<y>*	Odejmowanie matematyczne
<x> * *<y>*	Mnożenie matematyczne
<x> **/** *<y>*	Dzielenie matematyczne
<x> **** *<y>*	Dzielenie całkowite
<x> **modulo** *<y>*	Reszta z dzielenia całkowitego
Maksimum(*<x>*, *<y>*) **Maksimum**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Zwraca większą z dwóch wartości
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Zwraca mniejszą z dwóch wartości
Xy(*<x>*, *<y>*)	x do potęgi y
Wartość wykładnicza(*<x>*)	Stała x do potęgi y
Wartość bezwzględna(*<x>*) **|***<val>***|**	Wartość bezwzględna x
Logarytm naturalny(*<x>*)	Logarytm naturalny z x
Logarytm(*<x>*)	Logarytm dziesiętny z x
Pierwiastek kwadratowy(*<x>*)	Pierwiastek kwadratowy z x
Zaokrąglenie(*<x>*, *<n>*)	Zaokrągla x do n miejsc po przecinku
Ograniczenie cyfr po przecinku(*<x>*, *<n>*)	Wartość x skrócona do n miejsc po przecinku
Sin(*<x>*)	Sinus x
Cos(*<x>*)	Cosinus x
Tan(*<x>*)	Tangens x

Składnia	Opis
Asin(*<x>*)	Arcus sinus x
Acos(*<x>*)	Arcus cosinus x
Atan(*<x>*)	Arcus tangens x

Funkcje dotyczące daty(English)

Składnia	Opis
BieżącaData()	Zwraca bieżącą wartość ***date*** na początku sesji.
Data(*<text>*)	Konwertuje określony napis na wartość ***date***
UtwórzDatę(*<year>*, *<month>*, *<day>*)	Zwraca atrybut typu ***date*** utworzony z określonego roku, miesiąca i dnia.
WyodrębnijDzień(*<date/datetime>*)	Zwraca składnik "dzień" wartości w polu ***date/datetime attribute***.
WyodrębnijMiesiąc(*<date/datetime>*)	Zwraca składnik "miesiąc" wartości w polu ***date/datetime attribute***.
WyodrębnijRok (*<date/datetime>*)	Zwraca składnik "rok" wartości w polu ***date/datetime attribute***.
NastępnyDzieńTygodnia (*<date/datetime>*, *<day>*)	Zwraca datę ***date*** następnego dnia roboczego, która jest równa, wcześniejsza lub późniejsza względem ***date*** (zależnie od zastosowanej składni).
NastęnaData(*<date>*, *<day>*, *<month>*)	Zwraca następne wystąpienie danego dnia i miesiąca po dacie ***date***.
DodajDni(*<date/datetime>*, *<num_days>*)	Dodaje/odejmuje liczbę dni do/od ***date***. W przypadku użycia zwięzłej formy syntaktycznej musi to być dodatnia liczba całkowita, jeśli trzeba dodać dni do wejściowej wartości ***date***, albo liczba ujemna, jeśli trzeba odjąć dni od wejściowej wartości ***date***.
DodajTygodnie(*<date/datetime>*, *<num_weeks>*)	Dodaje liczbę tygodni w polu ***date***. Przy użyciu zwięzłej formy syntaktycznej liczba musi być dodatnią liczbą całkowitą, aby dodać tygodnie do wejściowej wartości w polu ***date***.
DodajMiesiące(*<date/datetime>*, *<num_months>*)	Dodaje liczbę miesięcy w polu ***date***. Przy użyciu zwięzłej formy syntaktycznej liczba musi być dodatnią liczbą całkowitą, aby dodać miesiące do wejściowej wartości w polu ***date***.
DodajLata(*<date/datetime>*, *<num_years>*)	Dodaje liczbę lat w polu ***date***. Przy użyciu zwięzłej formy syntaktycznej liczba musi być dodatnią liczbą całkowitą, aby dodać lata do wejściowej wartości w polu ***date***.
LiczbaDniRoboczych(*<date1>*, *<date2>*)	Oblicza liczbę dni roboczych między ***date***1 a ***date***2, czyli liczbę dni przypadających między poniedziałkiem a piątkiem. Uwaga: Pierwsza wartość ***date*** jest włączana do zakresu, a druga wartość ***date*** - nie.

Składnia	Opis
PoczątekRoku(*<date/datetime>*)	Zwraca pierwszą wartość ***date*** w roku, w którym przypada ***date***.
KoniecRoku(*<date/datetime>*)	Zwraca ostatnią wartość ***date*** w roku, w którym przypada ***date***.
RóżnicaDni(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych dni między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaDniWłącznie(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych dni (włącznie) między ***date/datetime1*** a ***date/datetime2***. Obliczenie uwzględnia obydwa punkty końcowe. Jeśli obie daty mają tę samą wartość, wynik wynosi 1. Kolejność dat nie ma wpływu na wynik.
RóżnicaDniZWyłączeniem(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych dni (z wyłączeniem) między ***date/datetime1*** a ***date/datetime2***. Obliczenie nie uwzględnia żadnego z dwóch punktów końcowych. Jeśli obie daty mają tę samą wartość, wynik wynosi 0. Kolejność dat nie ma wpływu na wynik.
RóżnicaTygodni(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych tygodni, jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaTygodniWłącznie(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych tygodni (włącznie), jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaTygodniZWyłączeniem(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych tygodni (z wyłączeniem), jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaMiesięcy(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych miesięcy, jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaMiesięcyWłącznie(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych miesięcy (włącznie), jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaMiesięcyZWyłączeniem(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę pełnych miesięcy (z wyłączeniem), jakie upłynęły między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaLat(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę lat między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaLatWłącznie(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę lat (włącznie) między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.
RóżnicaLatZWyłączeniem(*<date/datetime1>*, *<date/datetime2>*)	Zwraca liczbę lat (z wyłączeniem) między ***date/datetime1*** a ***date/datetime2***. Kolejność dat nie ma wpływu na wynik.

Funkcje dotyczące pory dnia(English)

Składnia	Opis
PoraDnia(*<text>*)	Konwertuje dany napis na porę dnia
WyodrębnijSekundy(*<time/datetime>*)	Zwraca składnik "sekunda" wartości w polu ***timeofday/datetime attribute***.
WyodrębnijMinuty(*<time/datetime>*)	Zwraca składnik "minuta" wartości w polu ***timeofday/datetime attribute***.
WyodrębnijGodzinę(*<time/datetime>*)	Zwraca składnik "godzina" wartości w polu ***timeofday/datetime attribute***.

Funkcje dotyczące daty i godziny(English)

Składnia	Opis
BieżącaDataIGodzina()	Zwraca bieżącą wartość ***date*** i godzinę na początku sesji.
DataGodzina(*<text>*)	Konwertuje określony napis na wartość ***datetime***
PołączDatęGodzinę(*<date>*, *<time>*)	Ustawia godzinę w polu ***date***, łącząc ze sobą atrybut ***date*** i porę dnia.
RóżnicaSekund(*<datetime1>*, *<datetime2>*) **RóżnicaSekund**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę sekund między ***datetime1*** a ***datetime2***.
RóżnicaSekundWłącznie(*<datetime1>*, *<datetime2>*) **RóżnicaSekundWłącznie**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę sekund między ***datetime1*** a ***datetime2*** włącznie z wartościami krańcowymi.
RóżnicaSekundZWyłączeniem(*<datetime1>*, *<datetime2>*) **RóżnicaSekundZWyłączeniem**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę sekund między ***datetime1*** a ***datetime2*** z wyłączeniem wartości krańcowych.
RóżnicaMinut(*<datetime1>*, *<datetime2>*) **RóżnicaMinut**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę minut między ***datetime1*** a ***datetime2***.
RóżnicaMinutWłącznie(*<datetime1>*, *<datetime2>*) **RóżnicaMinutWłącznie**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę minut między ***datetime1*** a ***datetime2*** włącznie z wartościami krańcowymi.
RóżnicaMinutZWyłączeniem(*<datetime1>*, *<datetime2>*) **RóżnicaMinutZWyłączeniem**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę minut między ***datetime1*** a ***datetime2*** z wyłączeniem wartości krańcowych.

Składnia	Opis
RóżnicaGodzin(*<datetime1>*, *<datetime2>*) **RóżnicaGodzin**(*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę godzin między ***datetime1*** a ***datetime2***.
RóżnicaGodzinWłącznie(*<datetime1>*, *<datetime2>*) **RóżnicaGodzinWłącznie** (*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę godzin między ***datetime1*** a ***datetime2*** włącznie z wartościami krańcowymi.
RóżnicaGodzinZWyłączeniem (*<datetime1>*, *<datetime2>*) **RóżnicaGodzinZWyłączeniem** (*<timeOfDay1>*, *<timeOfDay2>*)	Zwraca liczbę godzin między ***datetime1*** a ***datetime2*** z wyłączeniem wartości krańcowych.
WyodrębnijDatę(*<datetime>*)	Wyodrębnia wartość ***date*** z wartości w polu ***datetime attribute***.
WyodrębnijPoręDnia(*<datetime>*)	Wyodrębnia porę dnia z wartości w polu ***datetime attribute***. Można jej użyć do ustawienia wartości w polu ***timeofday attribute*** na godzinę wykonania reguły poprzez wyodrębnienie godziny z bieżącej wartości w polu ***date*** i godzina.
DodajGodziny(*<datetime>*, *<num_hours>*) **DodajGodziny**(*<timeOfDay>*, *<num_hours>*)	Dodaje liczbę godzin w polu ***date***.
DodajMinuty(*<datetime>*, *<num_minutes>*) **DodajMinuty**(*<timeOfDay>*, *<num_minutes>*)	Dodaje liczbę minut w polu ***date***.
DodajSekundy(*<datetime>*, *<num_seconds>*) **DodajSekundy**(*<timeOfDay>*, *<num_seconds>*)	Dodaje liczbę sekund w polu ***date***.

Funkcje tekstowe(English)

Składnia	Opis
<text1> **&** *<text2>*	Łączy ***text1*** z ***text2*** itd. w celu utworzenia pojedynczej wartości ***text***. Uwaga: można używać zmiennych dowolnego typu. Wartości formatowane są za pomocą programu formatującego zainstalowanego w sesji reguły.
	Łączy ***text1*** z ***text2*** itd. w celu utworzenia pojedynczej wartości ***text***. Uwaga: można używać zmiennych dowolnego typu. Wartości formatowane są za pomocą programu formatującego zainstalowanego w sesji reguły.

Składnia	Opis
Zawiera(*<text>*, *<substring>*)	Zwraca wartość logiczną wskazującą, czy dana wartość w polu ***text*** zawiera dany fragment napisu ***text***. W porównaniu ***text*** jest uwzględniana wielkość liter.
KończySię(*<text>*, *<substring>*)	Zwraca wartość logiczną wskazującą, czy dana wartość w polu ***text*** kończy się danym fragmentem napisu ***text***. W porównaniu ***text*** jest uwzględniana wielkość liter.
JestLiczbą(*<text>*)	Zwraca wartość logiczną wskazującą, czy dana wartość ***text*** oznacza prawidłową liczbę.
Długość(*<text>*)	Zwraca ilość znaków danej wartości atrybutu typu ***text***.
ZaczynaSię(*<text>*, *<substring>*)	Zwraca wartość logiczną wskazującą, czy dana wartość w polu ***text*** zaczyna się danym fragmentem napisu ***text***. W porównaniu ***text*** jest uwzględniana wielkość liter.
CiągPodrzędny (*<text>*, *<offset>*, *<length>*)	Zwraca fragment napisu atrybutu typu ***text*** zaczynający się z danym przesunięciem, o określonej ilości znaków. Jeśli zostanie osiągnięty koniec napisu, zwracanych jest mniej znaków.
Tekst(*<number>*) **Tekst**(*<date>*) **Tekst**(*<datetime>*) **Tekst**(*<timeOfDay>*)	Konwertuje określoną wartość liczbową lub wartość w polu ***date attribute*** na wartość atrybutu typu ***text***.

Funkcje dotyczące encji i relacji(English)

Składnia	Opis
Dla(*<relationship>*, *<Exp>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do jednego", "wiele do jednego" lub "wiele do wielu" ***relationship*** w przypadku, gdy istnieje tylko jeden warunek.
DlaZakresu(*<relationship>*, *<alias>*) **DlaZakresu**(*<relationship>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do jednego", "wiele do jednego" lub "wiele do wielu" ***relationship*** w przypadku, gdy istnieje przynajmniej jeden warunek.
DlaWszystkich(*<relationship>*, *<Exp>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do wielu" lub "wiele do wielu" ***relationship***, aby określić, czy wszystkie elementy docelowej grupy encji ***entity*** powinny być zgodne z regułą. Ta forma wykorzystywana jest w przypadku, gdy reguła zawiera tylko jeden warunek.
DlaCałegoZakresu(*<relationship>*) **DlaCałegoZakresu**(*<relationship>*, *<alias>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do wielu" lub "wiele do wielu" ***relationship***, aby określić, czy wszystkie elementy docelowej grupy encji ***entity*** powinny być zgodne z regułą. Ta forma wykorzystywana jest w przypadku, gdy reguła zawiera przynajmniej jeden warunek.
Istnieje(*<relationship>*, *<Exp>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do wielu" lub "wiele do wielu" ***relationship***, aby określić, czy jakiekolwiek elementy docelowej grupy encji ***entity*** powinny być zgodne z regułą.

Składnia	Opis
	Ta forma wykorzystywana jest w przypadku, gdy reguła zawiera tylko jeden warunek.
ZakresIstnieje(*<relationship>*) **ZakresIstnieje**(*<relationship>*, *<alias>*)	Używana w celu odniesienia encji ***entity*** względem innej encji ***entity*** w relacji "jeden do wielu" lub "wiele do wielu" ***relationship***, aby określić, czy jakiekolwiek elementy docelowej grupy encji ***entity*** powinny być zgodne z regułą. Ta forma wykorzystywana jest w przypadku, gdy reguła zawiera przynajmniej jeden warunek.
JestSkładową(*<target>*, *<relationship>*) **JestSkładową**(*<target>*, *<alias>*, *<relationship>*)	Używana jako wniosek sugerujący, że wystąpienie encji ***entity*** należy do relacji ***relationship***. Służy jako warunek pozwalający sprawdzić, czy wystąpienie encji ***entity*** jest celem relacji ***relationship***, której źródłem jest wystąpienie drugiej encji ***entity***.
NieJestSkładową(*<target>*, *<relationship>*)	Używana jako warunek pozwalający sprawdzić, czy wystąpienie encji ***entity*** nie jest celem relacji ***relationship***, której źródłem jest drugie wystąpienie encji ***entity***.
LiczbaWystąpień(*<relationship>*)	Liczy liczbę wystąpień istniejących dla encji ***entity***.
LiczbaWystąpieńJeśli(*<relationship>*, *<Exp>*)	Liczy liczbę wystąpień istniejących dla encji ***entity***, dla których atrybut ***entity-level attribute*** ma konkretną wartość.
MaksimumWystąpień(*<relationship>*, *<number-attr>*) **MaksimumWystąpień**(*<relationship>*, *<date-attr>*) **MaksimumWystąpień**(*<relationship>*, *<datetime-attr>*) **MaksimumWystąpień**(*<relationship>*, *<time-attr>*)	Pobiera najwyższą/najnowszą wartość zmiennej ***entity-level*** dla wszystkich wystąpień encji ***entity***.
MaksimumWystąpieńJeśli (*<relationship>*, *<number-attr>*, *<condition>*) **MaksimumWystąpieńJeśli** (*<relationship>*, *<date-attr>*, *<condition>*) **MaksimumWystąpieńJeśli** (*<relationship>*, *<datetime-attr>*, *<condition>*) **MaksimumWystąpieńJeśli** (*<relationship>*, *<time-attr>*, *<condition>*)	Pobiera najwyższą/najnowszą wartość zmiennej ***entity-level*** dla wszystkich wystąpień encji ***entity***, dla których konkretny atrybut ***entity-level attribute*** ma konkretną wartość.
MinimumWystąpień(*<relationship>*, *<number-attr>*) **MinimumWystąpień**(*<rela-*	Pobiera najniższą/najstarszą wartość zmiennej ***entity-level*** dla wszystkich wystąpień encji ***entity***.

Składnia	Opis
tionship>, <*date-attr*>) **MinimumWystąpień**(<*relationship*>, <*datetime-attr*>) **MinimumWystąpień**(<*relationship*>, <*time-attr*>)	
MinimumWystąpieńJeśli (<*relationship*>, <*number-attr*>, <*condition*>) **MinimumWystąpieńJeśli** (<*relationship*>, <*date-attr*>, <*condition*>) **MinimumWystąpieńJeśli** (<*relationship*>, <*datetime-attr*>, <*condition*>) **MinimumWystąpieńJeśli** (<*relationship*>, <*time-attr*>, <*condition*>)	Pobiera najniższą/najstarszą wartość zmiennej ***entity-level*** dla wszystkich wystąpień encji ***entity***, dla których konkretny atrybut ***entity-level attribute*** ma konkretną wartość.
SumaWystąpień(<*relationship*>, <*number-attr*>)	Pobiera sumę wszystkich wystąpień zmiennej ***entity-level***.
SumaWystąpieńJeśli(<*relationship*>, <*number-attr*>, <*condition*>)	Pobiera sumę wszystkich wystąpień zmiennej ***entity-level***, w przypadku których prawdą jest dla ***entity***, że określona wartość logiczna ***attribute*** zmiennej ***entity-level*** to "prawda".
WartośćWystąpieńJeśli (<*relationship*>, <*number-attr*>, <*condition*>) **WartośćWystąpieńJeśli** (<*relationship*>, <*text-attr*>, <*condition*>) **WartośćWystąpieńJeśli** (<*relationship*>, <*date-attr*>, <*condition*>) **WartośćWystąpieńJeśli** (<*relationship*>, <*datetime-attr*>, <*condition*>) **WartośćWystąpieńJeśli** (<*relationship*>, <*time-attr*>, <*condition*>)	Pobiera wartość z unikatowego wystąpienia ***entity***, zidentyfikowanego spośród docelowych wystąpień ***entity*** encji ***relationship*** za pomocą warunku. • Jeśli warunek zidentyfikuje jedno docelowe wystąpienie ***entity***, wówczas wartość jest wartością obliczoną na podstawie tego wystąpienia ***entity***. • Jeśli warunek jest spełniony przez więcej niż jedno docelowe wystąpienie, wówczas zwracana jest wartość ***uncertain***. • Jeśli żadne docelowe wystąpienie nie spełnia warunku i wartość ***relationship*** jest znana, wówczas wartość wynosi ***uncertain***.
LiczbaWystąpieńRówne (<*instance1*>, <*instance2*>)	Określa, czy dwa wystąpienia encji ***entity*** są tym samym wystąpieniem.
LiczbaWystąpieńNierówne	Określa, czy dwa wystąpienia encji ***entity*** nie są tym samym wystąpieniem.

Składnia	Opis
(*<instance1>*, *<instance2>*)	
WnioskowanieWystąpienia (*<relationship>*, *<identity>*)	Używana jako wniosek sugerujący, że wystąpienie encji ***entity*** istnieje i należy do relacji ***relationship***.

Funkcje dotyczące relacji okresowych(English)

Składnia	Opis
LiczbaPrzedziałówOdmienne(*<start-date>*, *<end-date>*, *<variable>*) **LiczbaPrzedziałówOdmienne**(*<start-date>*, *<end-date>*, *<condition>*)	Liczy liczbę znanych odmiennych wartości dla zmiennej, w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
LiczbaPrzedziałówOdmienneJeśli (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Liczy liczbę znanych odmiennych wartości dla zmiennej, w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), uwzględniając tylko sytuacje, gdy wartość dla filtra logicznego to "prawda".
SumaDziennaPrzedziału(*<start-date>*, *<end-date>*, *<number-attr>*)	Oblicza sumę wartości zmiennej walutowej lub liczbowej, w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem). Przyjmuje się, że atrybut ***attribute*** jest ilością dzienną.
SumaDziennaPrzedziałuJeśli(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Oblicza sumę wszystkich dziennych wartości zmiennej walutowej lub liczbowej, w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), uwzględniając tylko sytuacje, gdy warunek jest spełniony.
MaksimumPrzedziału(*<start-date>*, *<end-date>*, *<number-attr>*) **MaksimumPrzedziału**(*<start-date>*, *<end-date>*, *<date-attr>*) **MaksimumPrzedziału**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MaksimumPrzedziału**(*<start-date>*, *<end-date>*, *<time-attr>*)	Wybiera wartość maksymalną zmiennej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
MaksimumPrzedziałuJeśli(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MaksimumPrzedziałuJeśli**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MaksimumPrzedziałuJeśli**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MaksimumPrzedziałuJeśli**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Wybiera wartość maksymalną zmiennej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), uwzględniając tylko sytuacje, gdy warunek jest spełniony.
MinimumPrzedziału(*<start-date>*, *<end-date>*, *<number-attr>*)	Wybiera wartość minimalną zmiennej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).

Składnia	Opis
MinimumPrzedziału(*<start-date>*, *<end-date>*, *<date-attr>*) **MinimumPrzedziału**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MinimumPrzedziału**(*<start-date>*, *<end-date>*, *<time-attr>*)	
MinimumPrzedziałyJeśli(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MinimumPrzedziałyJeśli**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MinimumPrzedziałyJeśli**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MinimumPrzedziałyJeśli**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Wybiera wartość minimalną zmiennej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), uwzględniając tylko sytuacje, gdy warunek jest spełniony.
ŚredniaWażonaPrzedziału(*<start-date>*, *<end-date>*, *<number-attribute>*)	Oblicza średnią wartość zmiennej walutowej lub liczbowej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), ważoną w zależności od przedziałów czasu, do których odnoszą się poszczególne wartości.
ŚredniaWażonaPrzedziałuJeśli(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Oblicza średnią wartość zmiennej walutowej lub liczbowej w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem), uwzględniając tylko sytuacje, gdy warunek logiczny jest spełniony (ważoną w zależności od przedziałów czasu, do których odnoszą się poszczególne wartości).
PrzedziałZawsze(*<start-date>*, *<end-date>*, *<condition>*)	Zwraca wartość "prawda" wtedy i tylko wtedy, gdy warunek logiczny jest zawsze spełniony w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
PrzedziałCoNajmniejDni(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Zwraca wartość "prawda" wtedy i tylko wtedy, gdy warunek logiczny jest spełniony co najmniej dla określonej liczby dni (niekoniecznie kolejnych) w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
PrzedziałKolejnychDni(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Zwraca wartość "prawda" wtedy i tylko wtedy, gdy warunek logiczny jest spełniony co najmniej dla danej liczby kolejnych dni w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
PrzedziałCzasami(*<start-date>*, *<end-date>*, *<condition>*)	Zwraca wartość "prawda" wtedy i tylko wtedy, gdy warunek logiczny jest kiedykolwiek spełniony w przedziale od daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem).
WartośćW(*<date>*, *<value>*)	Zwraca wartość danego atrybutu ***attribute*** dla określonej wartości atrybutu typu ***date***.
GdyOstatni(*<date>*, *<condition>*)	Zwraca wartość atrybutu ***date***, dla której warunek logiczny był ostatnim razem

Składnia	Opis
	spełniony, patrząc wstecz od określonej wartości ***date*** (włącznie z tą wartością).
GdyNastępny(*<date>*, *<condition>*)	Zwraca wartość ***date***, dla której warunek logiczny będzie następnym razem spełniony, zaczynając od określonej wartości ***date*** (włącznie z tą wartością).
Najpóźniejsze()	Zwraca wartość w polu ***date*** równoważną najpóźniejszej możliwej wartości w polu ***date*** - mianowicie wartość ***date***, która na pewno jest późniejsza niż każda inna wartość w polu ***date***, jaką może przyjąć ***date attribute*** lub wyrażenie.
Najwcześniejsze()	Zwraca wartość w polu ***date*** równoważną najwcześniejszej możliwej wartości w polu ***date*** - mianowicie wartość ***date***, która na pewno jest wcześniejsza niż każda inna wartość w polu ***date***, jaką może przyjąć ***date attribute*** lub wyrażenie.
OkresowoZmienneDniOd(*<date>*, *<end-date>*)	Zwraca zmienną liczbową różną każdego dnia i wskazującą liczbę pełnych dni od daty ***date***.
OkresowoZmienneTygodnieOd(*<date>*, *<end-date>*)	Zwraca zmienną liczbową różną każdego tygodnia i wskazującą liczbę pełnych tygodni od daty ***date***.
OkresowoZmienneMiesiąceOd(*<date>*, *<end-date>*)	Zwraca zmienną liczbową różną każdego miesiąca i wskazującą liczbę pełnych miesięcy od daty ***date***. Uwaga: W przypadku, gdy podana data ***date*** występuje po 28 dniu miesiąca, a kolejny miesiąc jest krótszy od podanego, punkt zmiany miesiąca rocznicowego zostanie utworzony ostatniego dnia kolejnego miesiąca. Przykładowo, jeśli podana data ***date*** to 28, 29, 30 lub 31 stycznia 2007 roku, to pierwszym punktem zmiany będzie dzień 28 lutego 2007 roku.
OkresowoZmienneLataOd(*<date>*, *<end-date>*)	Zwraca zmienną liczbową różną każdego roku i wskazującą liczbę pełnych lat od daty ***date***.
OkresowoZmienneZawszeDni(*<days>*, *<condition>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą wtedy i tylko wtedy, gdy warunek logiczny jest spełniony dla wszystkich z danej liczby poprzedzających dni, bez uwzględnienia dnia bieżącego.
OkresowoZmienneKolejneDni (*<minDays>*, *<days>*, *<condition>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą wtedy i tylko wtedy, gdy warunek logiczny jest spełniony co najmniej dla minimalnej liczby kolejnych dni w dowolnym czasie w ciągu ustalonej liczby poprzedzających dni, bez uwzględnienia dnia bieżącego.
OkresowoZmienneCzasamiDni (*<days>*, *<condition>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą wtedy i tylko wtedy, gdy warunek logiczny jest kiedykolwiek spełniony w ciągu określonej liczby poprzedzających dni, bez uwzględnienia dnia bieżącego.
OkresowoZmiennePo(*<date>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą po dacie ***date*** i fałszem w i przed tą datą.
OkresowoZmiennePrzed(*<date>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą

Składnia	Opis
	prawdą przed datą ***date*** i fałszem w i po tej dacie.
OkresowoZmienneWDniu(*<date>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą w datę ***date*** i fałszem przed i po tej dacie.
OkresowoZmienneWDniuLubPo (*<date>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą po dacie ***date*** i fałszem przed tą datą.
OkresowoZmienneWDniuLubPrzed (*<date>*)	Zwraca wartość atrybutu logicznego ***attribute*** zmienną w czasie i będącą prawdą przed datą ***date*** i fałszem po tej dacie.
OkresowoZmienneOdDatyRozpoczęcia (*<relationship>*, *<date>*, *<value>*)	Zwraca pojedynczą wartość atrybutu czasowego ***attribute*** (na poziomie encji źródłowej ***entity***) z relacji ***relationship*** i wartości atrybutu ***attribute*** dla encji, o wartościach obowiązujących od daty początkowej ***date attribute***.
OkresowoZmienneOdDatyZakończenia (*<relationship>*, *<date>*, *<value>*)	Zwraca pojedynczą wartość atrybutu czasowego ***attribute*** (na poziomie encji źródłowej ***entity***) z relacji ***relationship*** i wartości atrybutu ***attribute*** dla encji, o wartościach obowiązujących do daty końcowej ***date attribute***.
OkresowoZmienneOdZakresu(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Zwraca pojedynczą wartość atrybutu czasowego ***attribute*** (na poziomie encji źródłowej ***entity***) z relacji ***relationship*** i wartości atrybutu ***attribute*** dla encji, o wartościach obowiązujących od daty początkowej ***date attribute*** (włącznie) do daty końcowej ***date attribute*** (z wyłączeniem). Zwraca wartość ***uncertain***, jeśli przestanie obowiązywać przed następną datą początkową ***date***.
OkresowoZmienneJestDzieńRoboczy (*<startdate>*, *<enddate>*)	Zwraca wartość "prawda" w dni będące dniami roboczymi i "fałsz" w dni weekendu od określonej daty początkowej ***date*** (włącznie) i daty końcowej ***date*** (z wyłączeniem). Zwraca wartość ***uncertain*** poza zakresem ***date***.
OkresowoZmienneRazWMiesiącu (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Zwraca wartość "prawda", jeśli dzień jest równy wartości parametru dzień miesiąca, i "fałsz" we wszystkie inne dni od określonej daty początkowej ***date*** (włącznie) do daty końcowej ***date*** (z wyłączeniem). Zwraca wartość ***uncertain*** poza zakresem ***date***. Gdy wartość parametru dzień miesiąca przekracza liczbę dni w bieżącym miesiącu, zwracana jest wartość "prawda" - w ten sposób funkcja zwraca wartość "prawda" dokładnie przez jeden dzień w miesiącu.

Funkcje zdarzeń sprawdzania(English)

Składnia	Opis
Błąd (*<text>*)	Zdarzenie błędu służy do przekazywania użytkownikowi komunikatu oraz do przerywania toku badania aż do momentu, gdy warunek wywołujący błąd przestanie obowiązywać.
Ostrzeżenie (*<text>*)	Zdarzenie ostrzeżenia jest używane do przekazywania użytkownikowi komunikatu, ale umożliwia kontynuowanie pracy pomimo zaistnienia warunku, który zainicjował ostrzeżenie.

Funkcje odrzucone(English)

Składnia	Opis
WywołajFunkcjęNiestandardową (*<A>*, *<B>*)	Zwraca wynik zewnętrznego wywołania biblioteki kodów. Aby wywołanie funkcji niestandardowej powiodło się, należy podać bibliotekę kodów do motoru określającego.

Логические соединители(English)

Синтаксис	Описание
если	Необязательный элемент, который может появиться в конце строки заключения, имеющей следующее подтверждение
и	Логическая конъюнкция между двумя ***attributes***
или	Логическая дизъюнкция между двумя ***attributes***
либо **один из** **одно из** **любой** **хотя бы одно из следующих утверждений истинно** **удовлетворяется любое из следующих условий**	Элемент группировки, используемый с дизъюнкцией в случае, когда требуется сгруппировать два или более ***attributes***
оба **все** **все следующие утверждения истинны** **все следующие условия удовлетворяются**	Элемент группировки, используемый с конъюнкцией в случае, когда требуется сгруппировать два или более ***attributes***
иначе	Элемент, который появляется в конце табличного правила для указания оператора, выполняемого в остальных случаях
является	Элемент, который используется в записи условных обозначений между сокращенной фразой и полным ***attribute text***

Логические функции(English)

Синтаксис	Описание
неверно, что *<expr>*	Оператор, используемый для возвращения значения "истина", если ***attribute*** имеет значение "ложь"
<var> **является достоверным**	Оператор, используемый для возвращения значения "Истина", если ***attribute*** имеет значение, отличное от ***uncertain***

Синтаксис	Описание
достоверно известно, *<expr>*	
<var> **является сомнительным** **достоверно известно, что** *<expr>* **недостоверно известно,** *<expr>* **нет уверенности, что** *<expr>* **не достоверно**	Оператор, используемый для возвращения значения "истина", если ***attribute*** имеет значение ***uncertain***
<var> **является известным** **известно,** *<expr>*	Оператор, используемый для возвращения значения "истина", если ***attribute*** имеет любое значение
<var> **не является известным** **не известно,** *<expr>* **не известно**	Оператор, используемый для возвращения значения "истина", если ***attribute*** не имеет значений

Логические константы(English)

Синтаксис	Описание
истина	Постоянное значение "истина", используемое для правил таблицы.
ложь	Постоянное значение "ложь", используемое для табличных правил.
не достоверно	Постоянное значение ***uncertain***, используемое для табличных правил.

Операторы сравнения(English)

Синтаксис	Описание
<lhs> **<** *<rhs>*	Меньше Примечание: для использования этого оператора с числовыми и валютными значениями не предусмотрено формы на естественном языке.
<lhs> **>** *<rhs>*	Больше Примечание: для использования этого оператора с числовыми и валютными значениями не предусмотрено формы на естественном языке.
<lhs> **<=** *<rhs>*	Меньше или равно

Синтаксис	Описание
<lhs> **>=** *<rhs>*	Больше или равно
<lhs> **=** *<rhs>*	Равно
<lhs> **<>** *<rhs>*	Не равно

Числовые функции(English)

Синтаксис	Описание
Число(*<numText>*)	Преобразование указанной строки в числовое значение
<x> **+** *<y>*	Математическое сложение
<x> **-** *<y>*	Математическое вычитание
<lhs> * *<rhs>*	Математическое умножение
<lhs> **/** *<rhs>*	Математическое деление
<lhs> \ *<rhs>*	Целочисленное деление
<lhs> **modulo** *<rhs>*	Остаток от целочисленного деления
Максимум(*<x>*, *<y>*) **Максимум**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Возвращает большее из двух значений
Минимум(*<x>*, *<y>*) **Минимум**(*<date/time/datetime1>*, *<date/time/datetime2>*)	Возвращает меньшее из двух значений
ХстепY(*<x>*, *<y>*)	x в степени y
ЕстепХ(*<x>*)	е (основание натуральных логарифмов) в степени x
Модуль(*<x>*) **|***<val>***|**	Абсолютное значение x
НатЛог(*<x>*)	Натуральный логарифм x
ДесЛог(*<x>*)	Десятичный логарифм x
КвКорень(*<x>*)	Квадратный корень из x
Округл(*<x>*, *<n>*)	Округляет x до n десятичных знаков
Усеч(*<x>*, *<n>*)	Усекает x до n десятичных знаков
Синус(*<x>*)	Синус x

Синтаксис	Описание
Косинус(*<x>*)	Косинус x
Тангенс(*<x>*)	Тангенс x
Арксинус(*<x>*)	Арксинус x
Арккосинус(*<x>*)	Арккосинус x
Арктангенс(*<x>*)	Арктангенс x

Функции даты(English)

Синтаксис	Описание
ТекущаяДата()	Возвращает текущее значение "***date***" в начале сеанса.
Дата(*<text>*)	Преобразование указанной строки в значение ***date***
ДатаВыпуска(*<year>*, *<month>*, *<day>*)	Возвращает значение ***date***, образованное из указанного года, месяца и дня.
ИзвлечьДень(*<date/datetime>*)	Возвращает компонент дня для ***date/datetime attribute***.
ИзвлечьМесяц(*<date/datetime>*)	Возвращает компонент месяца для ***date/datetime attribute***.
ИзвлечьГод(*<date/datetime>*)	Возвращает компонент года для ***date/datetime attribute***.
СледДеньНедели(*<date/datetime>*, *<day>*)	Возвращает ***date*** следующего буднего дня до или после ***date*** включительно (в зависимости от используемого синтаксиса).
СледующаяДата(*<date>*, *<day>*, *<month>*)	Возвращает следующий экземпляр данного дня и месяца после ***date***.
ДобДн(*<date/datetime>*, *<num_days>*)	Прибавляет к ***date*** или вычитает указанное число дней. При использовании краткой синтаксической формы для прибавления дней к входному значению ***date*** необходимо указывать целое положительное число, а для вычитания дней из ***date*** - отрицательное число.
ДобНедели(*<date/datetime>*, *<num_weeks>*)	Прибавляет число недель к ***date***. При использовании краткой синтаксической формы число должно быть положительным целым числом для прибавления недель к вводу ***date***.
ДобМесяцы(*<date/datetime>*, *<num_months>*)	Прибавляет число месяцев к ***date***. При использовании краткой синтаксической формы число должно быть положительным целым числом для прибавления месяцев к вводу ***date***.
ДобГоды(*<date/datetime>*, *<num_years>*)	Прибавляет число лет к ***date***. При использовании краткой синтаксической формы число должно быть положительным целым числом для прибавления лет к вводу ***date***.

Синтаксис	Описание
ЧислоДнейНедели(*<date1>*, *<date2>*)	Подсчет числа будних дней между ***date***1 и ***date***2, т. е. дней с понедельника по пятницу. Примечание: более ранний ***date*** включается, а более поздний ***date*** исключается.
НачалоГода(*<date/datetime>*)	Возвращает первое значение ***date*** в году, в который попадает ***date***.
КонецГода(*<date/datetime>*)	Возвращает последнее значение ***date*** в году, в который попадает ***date***.
РазницаДней(*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых дней между ***date/datetime1*** и ***date/datetime2***. Порядок двух дат не влияет на результат.
РазницаДнейВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых дней между ***date/datetime1*** и ***date/datetime2*** (включая граничные даты). В этом расчете включаются обе конечные точки. Если граничные даты совпадают, возвращается результат 1. Порядок этих двух дат не влияет на результат.
РазницаДнейНеВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых дней между ***date/datetime1*** и ***date/datetime2*** (не включая граничные даты). В этом расчете исключаются обе конечные точки. Если граничные даты совпадают, возвращается результат 0. Порядок этих двух дат не влияет на результат.
РазницаНедель(*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших недель между ***date/datetime1*** и ***date/datetime2***. Порядок двух дат не влияет на результат.
РазницаНедельВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших недель между ***date/datetime1*** и ***date/datetime2*** (включая граничные точки). Порядок двух дат не влияет на результат.
РазницаНедельНеВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших недель между ***date/datetime1*** и ***date/datetime2*** (не включая граничные точки). Порядок двух дат не влияет на результат.
РазницаМесяцев(*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших месяцев между ***date/datetime1*** и ***date/datetime2***. Порядок двух дат не влияет на результат.
РазницаМесяцевВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших месяцев между ***date/datetime1*** и ***date/datetime2*** (включая граничные точки). Порядок двух дат не влияет на результат.
РазницаМесяцевНеВключительно (*<date/datetime1>*, *<date/datetime2>*)	Возвращает число целых прошедших месяцев между ***date/datetime1*** и ***date/datetime2*** (не включая граничные точки). Порядок двух дат не влияет на результат.
РазницаЛет(*<date/datetime1>*, *<date/datetime2>*)	Возвращает число лет между ***date/datetime1*** и ***date/datetime2***. Порядок двух дат не влияет на результат.
РазницаЛетВключительно(*<date/datetime1>*, *<date/datetime2>*)	Возвращает число лет между ***date/datetime1*** и ***date/datetime2*** (включая граничные точки). Порядок двух дат не влияет на результат.

Синтаксис	Описание
РазницаЛетНеВключительно (<*date/datetime1*>, <*date/datetime2*>)	Возвращает число лет между ***date/datetime1*** и ***date/datetime2*** (не включая граничные точки). Порядок двух дат не влияет на результат.

Функции времени суток(English)

Синтаксис	Описание
ВремяСуток(<*text*>)	Преобразование указанной строки во время дня
ИзвлечьСекунды(<*time/datetime*>)	Возвращает компонент секунд для ***timeofday/datetime attribute***.
ИзвлечьМинуты(<*time/datetime*>)	Возвращает компонент минут для ***timeofday/datetime attribute***.
ИзвлечьЧасы(<*time/datetime*>)	Возвращает компонент часа для ***timeofday/datetime attribute***.

Функции дат и времени(English)

Синтаксис	Описание
ТекущаяДатаВремя()	Возвращает текущее значение "***date*** и время" в начале сеанса.
ДатаВремя(<*text*>)	Преобразование указанной строки в значение ***datetime***
УсечьДатуВремя(<*date*>, <*time*>)	Устанавливает время ***date***, объединяя ***date*** и время дня.
РазницаСекунд(<*datetime1*>, <*datetime2*>) **РазницаСекунд**(<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число секунд между ***datetime1*** и ***datetime2***.
РазницаСекундВключительно (<*datetime1*>, <*datetime2*>) **РазницаСекундВключительно** (<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число секунд между ***datetime1*** и ***datetime2***, включая граничные точки.
РазницаСекундНеВключительно (<*datetime1*>, <*datetime2*>) **РазницаСекундНеВключительно** (<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число секунд между ***datetime1*** и ***datetime2***, исключая граничные точки.
РазницаМинут(<*datetime1*>, <*datetime2*>) **РазницаМинут**(<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число минут между ***datetime1*** и ***datetime2***.
РазницаМинутВключительно (<*datetime1*>, <*datetime2*>)	Возвращает число минут между ***datetime1*** и ***datetime2***, включая граничные точки.

Синтаксис	Описание
РазницаМинутВключительно (<*timeOfDay1*>, <*timeOfDay2*>)	
РазницаМинутНеВключительно (<*datetime1*>, <*datetime2*>) **РазницаМинутНеВключительно** (<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число минут между ***datetime1*** и ***datetime2***, исключая граничные точки.
РазницаЧасов(<*datetime1*>, <*datetime2*>) **РазницаЧасов**(<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число часов между ***datetime1*** и ***datetime2***.
РазницаЧасовВключительно (<*datetime1*>, <*datetime2*>) **РазницаЧасовВключительно** (<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число часов между ***datetime1*** и ***datetime2***, включая граничные точки.
РазницаЧасовНеВключительно (<*datetime1*>, <*datetime2*>) **РазницаЧасовНеВключительно** (<*timeOfDay1*>, <*timeOfDay2*>)	Возвращает число часов между ***datetime1*** и ***datetime2***, исключая граничные точки.
ИзвлечьДату(<*datetime*>)	Извлекает ***date*** из ***datetime attribute***.
ИзвлечьВремяСуток(<*datetime*>)	Извлекает время дня из ***datetime attribute***. Может использоваться для установки значения ***timeofday attribute*** на время выполнения правила, извлекая время из текущего значения ***date*** и времени.
ДобЧасы(<*datetime*>, <*num_hours*>) **ДобЧасы**(<*timeOfDay*>, <*num_hours*>)	Прибавляет число часов к времени ***date***.
ДобМинуты(<*datetime*>, <*num_minutes*>) **ДобМинуты**(<*timeOfDay*>, <*num_minutes*>)	Прибавляет число минут к времени ***date***.
ДобСекунды(<*datetime*>, <*num_seconds*>) **ДобСекунды**(<*timeOfDay*>, <*num_seconds*>)	Прибавляет число секунд к времени ***date***.

Текстовые функции(English)

Синтаксис	Описание
<text1> **&** *<text2>*	Объединяет ***text1*** с ***text2*** и т. д., образуя единое значение ***text***. Можно использовать переменные любого типа. Значения форматируются с помощью средства форматирования, устанавливаемого в сеансе работы с правилами.
	Объединяет ***text1*** с ***text2*** и т. д., образуя единое значение ***text***. Можно использовать переменные любого типа. Значения форматируются с помощью средства форматирования, устанавливаемого в сеансе работы с правилами.
Содержит(*<text>*, *<substring>*)	Возвращает логическое значение, указывающее, содержит ли указанное значение ***text*** данную подстроку ***text***. Сравнение ***text*** выполняется без учета регистра.
ОкончаниеС (*<text>*, *<substring>*)	Возвращает логическое значение, указывающее, оканчивается ли указанное значение ***text*** данной подстрокой ***text***. Сравнение ***text*** выполняется без учета регистра.
ЯвляетсяЧислом (*<text>*)	Возвращает логическое значение, указывающее, представляет ли данное значение ***text*** допустимое число.
Длина(*<text>*)	Возвращает длину в символах для данного значения ***text***.
НачинаетсяС (*<text>*, *<substring>*)	Возвращает логическое значение, указывающее, начинается ли указанное значение ***text*** данной подстрокой ***text***. Сравнение ***text*** выполняется без учета регистра.
Подстрока(*<text>*, *<offset>*, *<length>*)	Возвращает подстроку для ***text***, которая начинается с данным отступом и указывает длину в символах. Если достигнут конец строки, возвращается меньше символов.
Текст(*<number>*) **Текст**(*<date>*) **Текст**(*<datetime>*) **Текст**(*<timeOfDay>*)	Преобразование указанного числа или ***date attribute*** в значение ***text***.

Функции логических объектов и отношений(English)

Синтаксис	Описание
Для(*<relationship>*, *<Exp>*)	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа "один к одному", "многие к одному" или "многие ко многим", если имеется только одно условие.
ДляОбласти(*<relationship>*, *<alias>*) **ДляОбласти**(*<relationship>*)	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа "один к одному", "многие к одному" или "многие ко многим", если имеется одно или несколько условий.
ДляВсех(*<relationship>*, *<Exp>*)	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа "один ко многим" или "многие ко многим", когда требуется определить, должны ли все члены группы-адресата ***entity*** соответствовать правилу. Эта форма используется, если в правиле имеется только одно условие.
ДляВсейОбласти(*<rela-*	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа

Синтаксис	Описание
tionship>) **ДляВсейОбласти**(*<relationship>*, *<alias>*)	"один ко многим" или "многие ко многим", когда требуется определить, должны ли все члены группы-адресата ***entity*** соответствовать правилу. Эта форма используется, если в правиле имеется одно или несколько условий.
Существует(*<relationship>*, *<Exp>*)	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа "один ко многим" или "многие ко многим", когда требуется определить, должны ли какие-либо члены группы-адресата ***entity*** соответствовать правилу. Эта форма используется, если в правиле имеется только одно условие.
СуществуетОбласть(*<relationship>*) **СуществуетОбласть**(*<relationship>*, *<alias>*)	Используется для ссылки из одного ***entity*** на другое ***entity*** в ***relationship*** типа "один ко многим" или "многие ко многим", когда требуется определить, должны ли какие-либо члены группы-адресата ***entity*** соответствовать правилу. Эта форма используется, если в правиле имеется одно или несколько условий.
ЯвляетсяЧленом(*<target>*, *<relationship>*) **ЯвляетсяЧленом**(*<target>*, *<alias>*, *<relationship>*)	Используется в качестве заключения о том, что экземпляр ***entity*** является элементом ***relationship***. Используется в качестве условия проверки того, что экземпляр ***entity*** является конечным для ***relationship***, для которого второй экземпляр ***entity*** является исходным.
НеЯвляетсяЧленом (*<target>*, *<relationship>*)	Используется в качестве условия проверки того, что экземпляр ***entity*** не является конечным для ***relationship***, для которого второй экземпляр ***entity*** является исходным.
ЧислоЭкземпляров(*<relationship>*)	Считает количество экземпляров, существующих для ***entity***.
ЧислоЭкземпляровЕсли (*<relationship>*, *<Exp>*)	Считает количество имеющихся экземпляров ***entity***, для которых определенный ***entity-level attribute*** имеет конкретное значение.
ЭкземпляровМаксимум (*<relationship>*, *<number-attr>*) **ЭкземпляровМаксимум** (*<relationship>*, *<date-attr>*) **ЭкземпляровМаксимум** (*<relationship>*, *<datetime-attr>*) **ЭкземпляровМаксимум** (*<relationship>*, *<time-attr>*)	Получает самое высокое/самое последнее значение переменной ***entity-level*** для всех экземпляров ***entity***.
ЭкземпляровМаксимумЕсли (*<relationship>*, *<number-attr>*, *<condition>*) **ЭкземпляровМаксимумЕсли** (*<relationship>*, *<date-attr>*, *<condition>*) **ЭкземпляровМаксимумЕсли** (*<relationship>*, *<datetime-attr>*,	Получает самое высокое/самое последнее значение переменной ***entity-level*** для всех экземпляров ***entity***, для которых каждое ***entity-level attribute*** имеет отдельное значение.

Синтаксис	Описание
<condition>) **ЭкземпляровМаксимумЕсли** (*<relationship>*, *<time-attr>*, *<condition>*)	
ЭкземпляровМинимум(*<relationship>*, *<number-attr>*) **ЭкземпляровМинимум**(*<relationship>*, *<date-attr>*) **ЭкземпляровМинимум**(*<relationship>*, *<datetime-attr>*) **ЭкземпляровМинимум**(*<relationship>*, *<time-attr>*)	Получает самое низкое/самое давнее значение переменной ***entity-level*** для всех экземпляров ***entity***.
ЭкземпляровМинимумЕсли (*<relationship>*, *<number-attr>*, *<condition>*) **ЭкземпляровМинимумЕсли** (*<relationship>*, *<date-attr>*, *<condition>*) **ЭкземпляровМинимумЕсли** (*<relationship>*, *<datetime-attr>*, *<condition>*) **ЭкземпляровМинимумЕсли** (*<relationship>*, *<time-attr>*, *<condition>*)	Получает самое низкое/самое давнее значение переменной ***entity-level*** для всех экземпляров ***entity***, для которых каждое ***entity-level attribute*** имеет отдельное значение.
СуммаЭкземпляров(*<relationship>*, *<number-attr>*)	Получает сумму всех экземпляров переменной ***entity-level***.
СуммаЭкземпляровЕсли (*<relationship>*, *<number-attr>*, *<condition>*)	Получает сумму всех экземпляров переменной ***entity-level***, для которых является "истиной" для ***entity***, что определенное значение ***attribute*** логического условия ***entity-level*** является "истиной".
ЗначениеЭкземпляровЕсли (*<relationship>*, *<number-attr>*, *<condition>*) **ЗначениеЭкземпляровЕсли** (*<relationship>*, *<text-attr>*, *<condition>*) **ЗначениеЭкземпляровЕсли** (*<relationship>*, *<date-attr>*, *<condition>*) **ЗначениеЭкземпляровЕсли** (*<relationship>*, *<datetime-attr>*, *<condition>*)	Получает значение из уникального экземпляра ***entity***, идентифицированного из экземпляров-адресатов ***entity*** для ***relationship*** в соответствии с условием. • Если условие идентифицирует единственный экземпляр-адресат ***entity***, то значение вычисляется по этому экземпляру ***entity***. • Если условию соответствуют более одного экземпляра-адресата, то возвращается значение ***uncertain***. • Если никакие экземпляры-адресаты не соответствуют условию, и ***relationship*** известно, то возвращается значение ***uncertain***.

Синтаксис	Описание
ЗначениеЭкземпляровЕсли (*<relationship>*, *<time-attr>*, *<condition>*)	
ЭкземплярыРавны (*<instance1>*, *<instance2>*)	Определяет, являются ли два экземпляра ***entity*** одним и тем же экземпляром.
ЭкземплярыНеРавны (*<instance1>*, *<instance2>*)	Определяет, являются ли два экземпляра ***entity*** разными экземплярами.
InferInstance(*<relationship>*, *<identity>*)	Используется в качестве заключения для вывода о том, что экземпляр ***entity*** существует и является членом ***relationship***.

Функции временного вывода заключений(English)

Синтаксис	Описание
ЧислоИнтеваловБезПовторов(*<start-date>*, *<end-date>*, *<variable>*) **ЧислоИнтеваловБезПовторов**(*<start-date>*, *<end-date>*, *<condition>*)	Считает количество известных разных значений для переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
ЧислоИнтеваловБезПовторовЕсли (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Считает количество известных разных значений для переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), причем включается только время, когда логический фильтр дает значение "истина".
ЕжедневнСуммарныйИнтервал (*<start-date>*, *<end-date>*, *<number-attr>*)	Вычисляет сумму для переменной с типом Currency или Number в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку). Подразумевается, что ***attribute*** является количеством за день.
ЕжедневнСуммарныйИнтервалЕсли (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Вычисляет сумму всех ежедневных значений для переменной с типом Currency или Number в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), причем включается только время, когда условие является "истиной".
МаксимальныйИнтервал(*<start-date>*, *<end-date>*, *<number-attr>*) **МаксимальныйИнтервал**(*<start-date>*, *<end-date>*, *<date-attr>*) **МаксимальныйИнтервал**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **МаксимальныйИнтервал**(*<start-date>*, *<end-date>*, *<time-attr>*)	Выбирает максимальное значение переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).

Синтаксис	Описание
МаксимальныйИнтервалЕсли (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **МаксимальныйИнтервалЕсли** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **МаксимальныйИнтервалЕсли** (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **МаксимальныйИнтервалЕсли** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Выбирает максимальное значение переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), причем включается только время, когда условие является "истиной".
МинимальныйИнтервал(*<start-date>*, *<end-date>*, *<number-attr>*) **МинимальныйИнтервал**(*<start-date>*, *<end-date>*, *<date-attr>*) **МинимальныйИнтервал**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **МинимальныйИнтервал**(*<start-date>*, *<end-date>*, *<time-attr>*)	Выбирает минимальное значение переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
МинимальныйИнтервалЕсли(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **МинимальныйИнтервалЕсли**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **МинимальныйИнтервалЕсли**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **МинимальныйИнтервалЕсли**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Выбирает минимальное значение переменной в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), причем включается только время, когда условие является "истиной".
ВзвешенныйСреднийИнтервал (*<start-date>*, *<end-date>*, *<number-attribute>*)	Вычисляет среднее значение для переменной с типом Currency или Number в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), взвешенное по тому промежутку времени, к которому применяется каждое значение.
ВзвешенныйСреднийИнтервалЕсли (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Вычисляет среднее значение для переменной с типом Currency или Number в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку), причем включается только время, когда логическое условие дает значение "истина" (взвешивается по тому промежутку времени, к которому применяется каждое значение и в котором фильтр дает значение "истина").

Синтаксис	Описание
ИнтервалВсегда(*<start-date>*, *<end-date>*, *<condition>*)	Возвращает значение "истина", только если логическое условие является "истиной" во всем интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
ИнтервалНеМенееДней(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Возвращает значение "истина", только если логическое условие является "истиной" по крайней мере для указанного числа дней (необязательно последовательных) в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
ИнтервалПоследовательныхДней (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Возвращает значение "истина", только если логическое условие является "истиной" по крайней мере для указанного числа последовательных дней в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
ИнтервалИногда(*<start-date>*, *<end-date>*, *<condition>*)	Возвращает значение "истина", только если логическое условие является "истиной" в какой-то момент в интервале от начала ***date*** (включительно) до конца ***date*** (не включая конечную точку).
ЗначениеВ(*<date>*, *<value>*)	Возвращает значение данного ***attribute*** в указанном ***date***.
КогдаПоследний(*<date>*, *<condition>*)	Возвращает значение ***date***, на которое логическое условие было "истиной" в последний раз, отсчитывая в обратную сторону от указанного ***date*** (включительно).
КогдаДалее(*<date>*, *<condition>*)	Возвращает значение ***date***, на которое логическое условие будет "истиной" в следующий раз, отсчитывая вперед от указанного ***date*** (включительно).
СамоеПозднее()	Возвращает значение ***date***, эквивалентное самому позднему из возможных ***date***, а именно: ***date*** гарантированно является более поздним, чем любое другое значение ***date***, которое может принимать ***date attribute*** или до которого может быть оценено выражение.
СамоеРаннее()	Возвращает значение ***date***, эквивалентное самому раннему из возможных ***date***, а именно: ***date*** гарантированно является более ранним, чем любое другое значение ***date***, которое может принимать ***date attribute*** или до которого может быть оценено выражение.
ВременнДнейС(*<date>*, *<end-date>*)	Возвращает числовую переменную, которая изменяется каждый день и является числом полных дней, начиная с ***date***.
ВременнНедельС(*<date>*, *<end-date>*)	Возвращает числовую переменную, которая изменяется каждую неделю и является числом полных недель, начиная с ***date***.
ВременнМесяцевС(*<date>*, *<end-date>*)	Возвращает числовую переменную, которая изменяется каждый месяц и является числом полных месяцев, начиная с ***date***. Примечание. Если предоставленное значение ***date*** попадает после 28-го дня месяца, и в последующем месяце меньше дней, чем в предоставленном месяце, то

Синтаксис	Описание
	точка изменения будет создана в последний день того месяца. Например, если предоставленное значение ***date*** соответствует 28, 29, 30 или 31 января 2007 года, то первая точка изменения будет 28 февраля 2007 г.
ВременнЛетС(*<date>*, *<end-date>*)	Возвращает числовую переменную, которая изменяется каждый год и является числом полных лет, начиная с ***date***.
ВременнВсегдаДней(*<days>*, *<condition>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной", только если логическое условие является "истиной" для всего данного числа предыдущих дней, не включая текущий день.
ВременнПоследовательнДней (*<minDays>*, *<days>*, *<condition>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной", только если логическое условие является "истиной" по крайней мере для минимального числа последовательных предыдущих дней в любое время в заданном интервале предыдущих дней, не включая текущий день.
ВременнИногдаДней(*<days>*, *<condition>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной", только если логическое условие является "истиной" в какой-то момент в интервале заданного числа предыдущих дней, не включая текущий день.
ВременнПосле(*<date>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной" после ***date*** и "ложью" на эту дату и до нее.
ВременнДо(*<date>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной" до ***date*** и "ложью" на эту дату и после нее.
ВременнВМомент(*<date>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной" на дату ***date*** и "ложью" после нее.
ВременнВМоментИлиПосле(*<date>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной" после ***date*** и "ложью" до этой даты.
ВременнВМоментИлиДо(*<date>*)	Возвращает логическое значение ***attribute***, которое изменяется со временем и является "истиной" на дату ***date*** и до нее и "ложью" после нее.
ВременнСДатыНачала(*<relationship>*, *<date>*, *<value>*)	Возвращает единый временной ряд ***attribute*** (на уровне источника ***entity***) из ***relationship*** и значения ***attribute*** на логических объектах, со значениями, которые действуют с начала ***date attribute***.
ВременнСДатыОкончания(*<relationship>*, *<date>*, *<value>*)	Возвращает единый временной ряд ***attribute*** (на уровне источника ***entity***) из ***relationship*** и значения ***attribute*** на логических объектах, со значениями, которые действуют вплоть до конца ***date attribute***.
ВременнОтДиапазона(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Возвращает единый временной ряд ***attribute*** (на уровне источника ***entity***) из ***relationship*** и значения ***attribute*** на логических объектах, со значениями, которые действуют с начала ***date attribute*** (включительно)

Синтаксис	Описание
	до конца ***date attribute*** (не включая эту точку). Значение равно ***uncertain***, если оно истекает до следующего начала ***date***.
ВременнВыходные(*<startdate>*, *<enddate>*)	Возвращает значение "истина" на даты, которые являются будними днями, и "ложь" на даты, которые являются выходными днями, в интервале от указанного начала ***date*** (включительно) до конца ***date*** (не включая эту точку). Возвращает ***uncertain*** вне диапазона ***date***.
ВременнСЕжемесячно(*<startdate>*, *<enddate>*, *<dayofmonth>*)	Возвращает значение "истина", если день равен параметру "день месяца", и "ложь" во все другие дни месяца от указанного начала ***date*** (включительно) до конца ***date*** (не включая эту точку). Возвращает ***uncertain*** вне диапазона ***date***. Когда "день месяца" превышает число дней в текущем месяце, значение является "истиной" в последний день того месяца, чтобы функция возвращала истинное значение только один раз в месяц.

Функции события проверки(English)

Синтаксис	Описание
Ошибка(*<text>*)	Событие ошибки используется для передачи пользователю сообщения, после которого тому запрещается продолжать исследование до устранения условия, вызвавшего ошибку.
Предупреждение(*<text>*)	Событие предупреждения используется для передачи пользователю сообщения, после которого тому разрешается продолжить работу, невзирая на условие, вызвавшее предупреждение.

Функции, не рекомендуемые к использованию(English)

Синтаксис	Описание
ВызвНестандФункцию(*<A>*, *<B>*)	Возвращает результат из внешнего вызова в библиотеку кодов. Для успешного вызова пользовательской функции необходимо предоставить ядру определений библиотеку кодов.

Conectores lógicos(English)

Sintaxis	Descripción
si	Condición que puede aparecer al final de una línea de conclusión que tiene una prueba siguiente
y	Conjunción lógica entre dos ***attributes***
o	Disyunción lógica entre dos ***attributes***

Sintaxis	Descripción
uno de **una de** **o bien** **cualquier de los dos** **cualquier de las dos** **cualquier de los** **cualquier de las** **cualquier** **cualquiera** **cualquiera de** **cualquiera de estas** **cualquiera de las siguientes** **por lo menos uno de los siguientes es verdadero** **cualquier de los siguientes ha sido cumplido**	Elemento de agrupación utilizado con disyunciones en las que necesitan agruparse dos o más ***attributes***
ambos **ambas** **todos** **todo** **tanto** **todas** **todos los siguientes son verdaderos** **todos los siguientes han sido cumplidos**	Elemento de agrupación utilizado con conjunciones en las que necesitan agruparse dos o más ***attributes***
de otro modo	Condición que aparece al final de una regla de tabla, para indicar la cláusula "de otro modo"
es	Condición que se utiliza en una entrada de leyenda, entre la frase abreviada y ***attribute text*** completos.

Funciones lógicas(English)

Sintaxis	Descripción
no es verdad que *<expr>*	Operador utilizado para devolver un valor verdadero, si ***attribute*** tiene un valor que es falso.
<var> **es seguro** *<var>* **es segura** **es seguro** [**o no**] **si** *<expr>*	Operador utilizado para devolver un valor verdadero, si ***attribute*** tiene un valor que no es ***uncertain***
<var> **no es seguro**	Operador utilizado para devolver un valor verdadero, si el valor de ***attribute*** es ***uncertain***

Sintaxis	Descripción
<*var*> **no es segura** <*var*> **es inseguro** <*var*> **es insegura** **es inseguro que** <*expr*> **es inseguro** [**o no**] **si** <*expr*> **no es seguro que** <*expr*> **inseguro** **insegura**	
<*var*> **es conocido** <*var*> **es conocida** **es conocido** [**o no**] **si** <*expr*>	Operador utilizado para devolver un valor verdadero, si ***attribute*** tiene algún valor.
<*var*> **es desconocido** <*var*> **es desconocida** **es desconocido** [**o no**] **si** <*expr*> **desconocido** **desconocida**	Operador utilizado para devolver un valor verdadero, si ***attribute*** no tiene un valor

Constantes lógicas(English)

Sintaxis	Descripción
verdad **verdadero**	Valor verdadero de constante utilizado para reglas de tablas.
falso	Valor falso de constante utilizado para reglas de tablas.
inseguro **incierto**	Valor ***uncertain*** de constante utilizado para reglas de tablas.

Operadores de comparación(English)

Sintaxis	Descripción
<*lhs*>**<**<*rhs*> <*lhs*> **es menor que** <*rhs*> <*lhs*> **es antes que** <*rhs*>	Menor que Nota: no hay una forma de idioma natural cuando este operador se utiliza con valores numéricos y de moneda.
<*lhs*> **>** <*rhs*> <*lhs*> **es mayor que** <*rhs*> <*lhs*> **es después que** <*rhs*>	Mayor que Nota: no hay una forma de idioma natural cuando este operador se utiliza con valores numéricos y de moneda.
<*lhs*>**<=**<*rhs*> <*lhs*> **es menor que o igual a**	Menor o igual que

Sintaxis	Descripción
<rhs>	
<lhs> **>=** *<rhs>* *<lhs>* **es mayor que o igual a** *<rhs>*	Mayor o igual que
*<lhs>***=***<rhs>* *<lhs>* **es igual a** *<rhs>* *<lhs>* **iguala** *<rhs>*	Igual a
<lhs> **no es igual a** *<rhs>* *<lhs>* **no iguala** *<rhs>* *<lhs>* **<>** *<rhs>* *<lhs>* **!=** *<rhs>*	Distinto de

Funciones numéricas(English)

Sintaxis	Descripción
Número(*<numText>*)	Convierte la cadena especificada en un valor numérico
<x> **+** *<y>*	Suma
<x> **-** *<y>*	Resta
<x> ***** *<y>*	Multiplicación
<x> **/** *<y>*	División
<x> **** *<y>*	División de enteros
<x> **modulo** *<y>*	El resto tras la división de enteros
Máximo(*<x>*, *<y>*) **Máximo**(*<date/time/datetime1>*, *<date/time/datetime2>*) **el más grande de** *<val1>* **y** *<val2>* **el máximo de** *<val1>* **y** *<val2>* **el mayor de** *<val1>* **y** *<val2>* **el más reciente de** *<val1>* **y** *<val2>* **la fecha más reciente de** *<val1>* **y** *<val2>*	Devuelve el mayor de dos valores
Mínimo(*<x>*, *<y>*) **Mínimo**(*<date/time/datetime1>*, *<date/time/datetime2>*) **el más pequeño de** *<val1>* **y** *<val2>* **el mínimo de** *<val1>* **y** *<val2>* **el menor de** *<val1>* **y** *<val2>* **el más temprano de** *<val1>* **y** *<val2>* **la fecha más temprana de** *<val1>* **y** *<val2>*	Devuelve el menor de dos valores

Sintaxis	Descripción
Xy(*<x>*, *<y>*)	x a la potencia de y
Ex(*<x>*)	Constante e a la potencia de x
Abs(*<x>*)	Valor absoluto de x
Ln(*<x>*)	Logaritmo natural de x
Log(*<x>*)	Logaritmo base 10 de x
Sqrt(*<x>*) **la raíz cuadrada de** *<val>*	Raíz cuadrada de x
Round(*<x>*, *<n>*) *<val>* **redondeado a** *<num_places>* **decimal** *<val>* **redondeado a** *<num_places>* **decimales**	Redondea x a n decimales
Trunc(*<x>*, *<n>*)	x truncado a n decimales
Sin(*<x>*)	Seno de x
Cos(*<x>*)	Coseno de x
Tan(*<x>*)	Tangente de x
Asin(*<x>*)	Arcoseno de x
Acos(*<x>*)	Arcoseno de x
Atan(*<x>*)	Arcotangente de x

Funciones de fecha(English)

Sintaxis	Descripción
FechaActual() **la fecha actual**	Devuelve la ***date*** actual al inicio de la sesión.
Fecha(*<text>*)	Convierte la cadena especificada en un valor de ***date***
FechaCreación(*<year>*, *<month>*, *<day>*)	Devuelve una ***date*** formada a partir del año, mes y día indicados.
DíaExtracción(*<date/datetime>*)	Devuelve el componente de día de una ***date/datetime attribute***.
MesExtracción (*<date/datetime>*)	Devuelve el componente de mes de una ***date/datetime attribute***.
AñoExtracción (*<date/datetime>*)	Devuelve el componente de año de una ***date/datetime attribute***.

Sintaxis	Descripción
SiguienteDíadelaSemana(*<date/datetime>*, *<day>*) **el** [**próximo**] **lunes durante o después de** *<from-date>* **el** [**próximo**] **lunes en o después de** *<from-date>* **el** [**próximo**] **martes durante o después de** *<from-date>* **el** [**próximo**] **martes en o después de** *<from-date>* **el** [**próximo**] **miércoles durante o después de** *<from-date>* **el** [**próximo**] **miércoles en o después de** *<from-date>* **el** [**próximo**] **jueves durante o después de** *<from-date>* **el** [**próximo**] **jueves en o después de** *<from-date>* **el** [**próximo**] **viernes durante o después de** *<from-date>* **el** [**próximo**] **viernes en o después de** *<from-date>* **el** [**próximo**] **sábado durante o después de** *<from-date>* **el** [**próximo**] **sábado en o después de** *<from-date>* **el** [**próximo**] **domingo durante o después de** *<from-date>* **el** [**próximo**] **domingo en o después de** *<from-date>*	Devuelve la ***date*** del siguiente día de la semana antes/después de una ***date*** (en función de la sintaxis que se utilice).
FechaSiguiente(*<date>*, *<day>*, *<month>*)	Devuelve la siguiente instancia del día y mes indicados, después de ***date***.
AgregarDías(*<date/datetime>*, *<num_days>*) **la fecha** *<num_days>* **día después de** *<date>* **la fecha** *<num_days>* **días después de** *<date>* **el tiempo** *<num_days>* **día después de** *<date>* **el tiempo** *<num_days>* **días después de** *<date>*	Agrega/resta un número de días a una ***date***. Cuando se utiliza una forma sintáctica concisa, el número debe ser un entero positivo, para agregar días a la ***date*** de entrada, o un número negativo, para restar días de la ***date*** de entrada.

Sintaxis	Descripción
AgregarSemanas(*<date/datetime>*, *<num_weeks>*) **la fecha** *<num_weeks>* **semana después de** *<date>* **la fecha** *<num_weeks>* **semanas después de** *<date>* **el tiempo** *<num_weeks>* **semana después de** *<date>* **el tiempo** *<num_weeks>* **semanas después de** *<date>*	Agrega un número de semanas a una ***date***. Cuando se utiliza una forma sintáctica concisa, el número debe ser un entero positivo, para agregar semanas a la ***date*** de entrada.
AgregarMeses(*<date/datetime>*, *<num_months>*) **la fecha** *<num_months>* **mes después de** *<date>* **la fecha** *<num_months>* **meses después de** *<date>* **el tiempo** *<num_months>* **mes después de** *<date>* **el tiempo** *<num_months>* **meses después de** *<date>*	Agrega un número de meses a una ***date***. Cuando se utiliza una forma sintáctica concisa, el número debe ser un entero positivo, para agregar meses a la ***date*** de entrada.
AgregarAños(*<date/datetime>*, *<num_years>*) **la fecha** *<num_years>* **año después de** *<date>* **la fecha** *<num_years>* **años después de** *<date>* **el tiempo** *<num_years>* **año después de** *<date>* **el tiempo** *<num_years>* **años después de** *<date>*	Agrega un número de años a una ***date***. Cuando se utiliza una forma sintáctica concisa, el número debe ser un entero positivo, para agregar años a la ***date*** de entrada.
RecuentodeDíasdelaSemana (*<date1>*, *<date2>*) **el número de días** [**enteros**] **de semana de** *<date1>* **a** *<date2>* **el número de días** [**enteros**] **de semana desde** *<date1>* **hasta** *<date2>* **el número de días** [**enteros**] **de semana entre** *<date1>* **y** *<date2>*	Hace un recuento del número de días de la semana entre ***date***1 y ***date***2. Es decir, el número de días comprendido entre el lunes y el viernes. Nota: está incluida la ***date*** anterior y excluida la ***date*** posterior.
IniciodeAño(*<date/datetime>*)	Devuelve la primera ***date*** del año en la que queda incluida una ***date***.

Sintaxis	Descripción
el primer día del año en el que *<from-date>* **cae** **el primer día del año en el cual cae** *<from-date>*	
FindeAño(*<date/datetime>*) **el último día del año en el que** *<from-date>* **cae** **el último día del año en el cual cae** *<from-date>*	Devuelve la última ***date*** del año en el que queda incluida una ***date***.
DiferenciaDías(*<date/datetime1>*, *<date/datetime2>*) **el número de días** [**enteros**] **de** *<date1>* **a** *<date2>* **el número de días** [**enteros**] **desde** *<date1>* **hasta** *<date2>* **el número de días** [**enteros**] **entre** *<date1>* **y** *<date2>*	Devuelve el número de días completos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaDíasIncluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de días** [**enteros**] **(inclusivo) de** *<date1>* **a** *<date2>* **el número de días** [**enteros**] **(inclusivo) desde** *<date1>* **hasta** *<date2>* **el número de días** [**enteros**] **(inclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de días (incluidos) entre ***date/datetime1*** y ***date/datetime2***. Este cálculo excluye ambos extremos. Cuando las fechas son iguales, el resultado es 1. El orden de las dos fechas no afecta al resultado.
DiferenciaDíasExcluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de días** [**enteros**] **(exclusivo) de** *<date1>* **a** *<date2>* **el número de días** [**enteros**] **(exclusivo) desde** *<date1>* **hasta** *<date2>* **el número de días** [**enteros**] **(exclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de días (excluidos) entre ***date/datetime1*** y ***date/datetime2***. Este cálculo excluye ambos extremos. Cuando las fechas son iguales, el resultado es 0. El orden de las dos fechas no afecta al resultado.

Sintaxis	Descripción
DiferenciaSemanas(*<date/datetime1>*, *<date/datetime2>*) **el número de semanas** [**enteras**] **de** *<date1>* **a** *<date2>* **el número de semanas** [**enteras**] **desde** *<date1>* **hasta** *<date2>* **el número de semanas** [**enteras**] **entre** *<date1>* **y** *<date2>*	Devuelve el número total de semanas transcurridas entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaSemanasIncluyendo(*<date/datetime1>*, *<date/datetime2>*) **el número de semanas** [**enteras**] **(inclusivo) de** *<date1>* **a** *<date2>* **el número de semanas** [**enteras**] **(inclusivo) desde** *<date1>* **hasta** *<date2>* **el número de semanas** [**enteras**] **(inclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de semanas incluidas transcurridas entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaSemanasExluyendo(*<date/datetime1>*, *<date/datetime2>*) **el número de semanas** [**enteras**] **(exclusivo) de** *<date1>* **a** *<date2>* **el número de semanas** [**enteras**][**(exclusivo)**] **desde** *<date1>* **hasta** *<date2>* **el número de semanas** [**enteras**] **(exclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de semanas transcurridas entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaMeses(*<date/datetime1>*, *<date/datetime2>*) **el número de meses** [**enteros**] **de** *<date1>* **a** *<date2>* **el número de meses** [**enteros**] **desde** *<date1>* **hasta** *<date2>* **el número de meses** [**enteros**] **entre** *<date1>* **y** *<date2>*	Devuelve el número total de meses transcurridos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.

Sintaxis	Descripción
DiferenciaMesesIncluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de meses** [**enteros**] **(inclusivo) de** *<date1>* **a** *<date2>* **el número de meses** [**enteros**] **(inclusivo) desde** *<date1>* **hasta** *<date2>* **el número de meses** [**enteros**] **(inclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de meses transcurridos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaMesesExcluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de meses** [**enteros**] **(exclusivo) de** *<date1>* **a** *<date2>* **el número de meses** [**enteros**] [**(exclusivo)**] **desde** *<date1>* **hasta** *<date2>* **el número de meses** [**enteros**] **(exclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número total de meses transcurridos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaAños(*<date/datetime1>*, *<date/datetime2>*) **el número de años** [**enteros**] **de** *<date1>* **a** *<date2>* **el número de años** [**enteros**] **desde** *<date1>* **hasta** *<date2>* **el número de años** [**enteros**] **entre** *<date1>* **y** *<date2>*	Devuelve el número de años entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.
DiferenciaAñosIncluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de años** [**enteros**] **(inclusivo) de** *<date1>* **a** *<date2>* **el número de años** [**enteros**] **(inclusivo) desde** *<date1>* **hasta** *<date2>* **el número de años** [**enteros**]	Devuelve el número de años incluidos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.

Sintaxis	Descripción
(inclusivo) entre *<date1>* **y** *<date2>*	
DiferenciaAñosExcluyendo (*<date/datetime1>*, *<date/datetime2>*) **el número de años** [**enteros**] **(exclusivo) de** *<date1>* **a** *<date2>* **el número de años** [**enteros**] **(exclusivo) desde** *<date1>* **hasta** *<date2>* **el número de años** [**enteros**] **(exclusivo) entre** *<date1>* **y** *<date2>*	Devuelve el número de años excluidos entre ***date/datetime1*** y ***date/datetime2***. El orden de las dos fechas no afecta al resultado.

Funciones de hora del día(English)

Sintaxis	Descripción
HoraDelDía(*<text>*)	Convierte la cadena indicada en una hora del día
SegundoExtracción(*<time/datetime>*)	Devuelve el componente de segundos de una ***timeofday/datetime attribute***.
MinutoExtracción(*<time/datetime>*)	Devuelve el componente de minuto de una ***timeofday/datetime attribute***.
HoraExtracción(*<time/datetime>*)	Devuelve el componente de hora de una ***timeofday/datetime attribute***.

Funciones de fecha y hora(English)

Sintaxis	Descripción
FechaHoraActual()	Devuelve la ***date*** y hora actuales al inicio de la sesión.
FechaHora(*<text>*)	Convierte la cadena especificada en un valor de ***datetime***
HoraDíaConcatenar(*<date>*, *<time>*)	Define la hora de una ***date***, mediante la unión de la ***date*** y la hora del día.
DiferenciaSegundos(*<datetime1>*, *<datetime2>*) **DiferenciaSegundos** (*<timeOfDay1>*, *<timeOfDay2>*) **el número de secondes de** *<date1>* **à** *<date2>* **el número de secondes desde**	Devuelve el número de segundos entre ***datetime1*** y ***datetime2***.

Sintaxis	Descripción
<date1> **hasta** *<date2>* **el número de secondes entre** *<date1>* **y** *<date2>*	
DiferenciaSegundosIncluyendo (*<datetime1>*, *<datetime2>*) **DiferenciaSegundosIncluyendo** (*<timeOfDay1>*, *<timeOfDay2>*) **el número de secondes (inclusif) de** *<date1>* **à** *<date2>* **el número de secondes (inclusif) desde** *<date1>* **hasta** *<date2>* **el número de secondes (inclusif) entre** *<date1>* **y** *<date2>*	Devuelve el número de segundos incluidos entre ***datetime1*** y ***datetime2***.
DiferenciaSegundosExcluyendo (*<datetime1>*, *<datetime2>*) **DiferenciaSegundosExcluyendo** (*<timeOfDay1>*, *<timeOfDay2>*) **el número de secondes (exclusif) de** *<date1>* **à** *<date2>* **el número de secondes (exclusif) desde** *<date1>* **hasta** *<date2>* **el número de secondes (exclusif) entre** *<date1>* **y** *<date2>*	Devuelve el número de segundos excluidos entre ***datetime1*** y ***datetime2***.
DiferenciaMinutos(*<datetime1>*, *<datetime2>*) **DiferenciaMinutos** (*<timeOfDay1>*, *<timeOfDay2>*) **el número de minutos de** *<date1>* **à** *<date2>* **el número de minutos desde** *<date1>* **hasta** *<date2>* **el número de minutos entre** *<date1>* **y** *<date2>*	Devuelve el número de minutos entre ***datetime1*** y ***datetime2***.
DiferenciaMinutosIncluyendo (*<datetime1>*, *<datetime2>*) **DiferenciaMinutosIncluyendo** (*<timeOfDay1>*, *<timeOfDay2>*)	Devuelve el número de minutos incluidos entre ***datetime1*** y ***datetime2***.

Sintaxis	Descripción
el número de minutos (inclusif) de *<date1>* **à** *<date2>* **el número de minutos (inclusif) desde** *<date1>* **hasta** *<date2>* **el número de minutos (inclusif) entre** *<date1>* **y** *<date2>*	
DiferenciaMinutosExcluyendo(*<datetime1>*, *<datetime2>*) **DiferenciaMinutosExcluyendo**(*<timeOfDay1>*, *<timeOfDay2>*) **el número de minutos (exclusif) de** *<date1>* **à** *<date2>* **el número de minutos (exclusif) desde** *<date1>* **hasta** *<date2>* **el número de minutos (exclusif) entre** *<date1>* **y** *<date2>*	Devuelve el número de minutos excluidos entre ***datetime1*** y ***datetime2***.
DiferenciaHoras(*<datetime1>*, *<datetime2>*) **DiferenciaHoras**(*<timeOfDay1>*, *<timeOfDay2>*) **el número de horas de** *<date1>* **à** *<date2>* **el número de horas desde** *<date1>* **hasta** *<date2>* **el número de horas entre** *<date1>* **y** *<date2>*	Devuelve el número de horas entre ***datetime1*** y ***datetime2***.
DiferenciaHorasIncluyendo(*<datetime1>*, *<datetime2>*) **DiferenciaHorasIncluyendo**(*<timeOfDay1>*, *<timeOfDay2>*) **el número de horas (inclusif) de** *<date1>* **à** *<date2>* **el número de horas (inclusif) desde** *<date1>* **hasta** *<date2>* **el número de horas (inclusif) entre** *<date1>* **y** *<date2>*	Devuelve el número de horas incluidas entre ***datetime1*** y ***datetime2***.
DiferenciaHorasExcluyendo	Devuelve el número de horas excluidas entre ***datetime1*** y ***datetime2***.

Sintaxis	Descripción
(*<datetime1>*, *<datetime2>*) **DiferenciaHorasExcluyendo** (*<timeOfDay1>*, *<timeOfDay2>*) **el número de horas (exclusif) de** *<date1>* **à** *<date2>* **el número de horas (exclusif) desde** *<date1>* **hasta** *<date2>* **el número de horas (exclusif) entre** *<date1>* **y** *<date2>*	
FechaExtracción(*<datetime>*)	Extrae la ***date*** desde una ***datetime attribute***.
HoraDíaExtracción(*<datetime>*)	Extrae la hora del día desde una ***datetime attribute***. Se puede utilizar para definir el valor de una ***timeofday attribute*** a la hora de ejecución de la regla, mediante la extracción de la hora y la ***date*** actuales.
AgregarHoras(*<datetime>*, *<num_hours>*) **AgregarHoras**(*<timeOfDay>*, *<num_hours>*) **el tiempo** *<num_hours>* **hora después de** *<date>* **el tiempo** *<num_hours>* **horas después de** *<date>* **el tiempo de día** *<num_hours>* **hora después de** *<time-of-day>* **el tiempo de día** *<num_hours>* **horas después de** *<time-of-day>*	Agrega un número de horas a la hora de una ***date***.
AgregarMinutos(*<datetime>*, *<num_minutes>*) **AgregarMinutos**(*<timeOfDay>*, *<num_minutes>*) **el tiempo** *<num_minutes>* **minuto después de** *<date>* **el tiempo** *<num_minutes>* **minutos después de** *<date>* **el tiempo de día** *<num_minutes>* **minuto después de** *<time-of-day>* **el tiempo de día** *<num_minutes>* **minutos después de** *<time-of-day>*	Agrega un número de minutos a la hora de una ***date***.
AgregarSegundos(*<datetime>*,	Agrega un número de segundos a la hora de una ***date***.

Sintaxis	Descripción
<num_seconds>) **AgregarSegundos**(*<timeOfDay>*, *<num_seconds>*) **el tiempo** *<num_seconds>* **segundo después de** *<date>* **el tiempo** *<num_seconds>* **segundos después de** *<date>* **el tiempo de día** *<num_seconds>* **segundo después de** *<time-of-day>* **el tiempo de día** *<num_seconds>* **segundos después de** *<time-of-day>*	

Funciones de texto(English)

Sintaxis	Descripción
<text1> **&** *<text2>*	Combina ***text1*** con ***text2*** y así sucesivamente, para formar un valor de ***text*** único. Nota: puede utilizar variables de cualquier tipo. El formato de los valores se aplica con el formateador instalado en la sesión de regla.
la concatenación de *<text1>* **&** *<text2>*	Combina ***text1*** con ***text2*** y así sucesivamente, para formar un valor de ***text*** único. Nota: puede utilizar variables de cualquier tipo. El formato de los valores se aplica con el formateador instalado en la sesión de regla.
Contiene(*<text>*, *<substring>*) *<text>* **contiene** *<substring>*	Devuelve un valor booleano que indica si el valor indicado de ***text*** contiene la subcadena de ***text*** indicada. La comparación de ***text*** no distingue mayúsculas y minúsculas.
FinalizaCon (*<text>*, *<substring>*) *<text>* **termina con** *<substring>*	Devuelve un valor booleano que indica si el valor indicado de ***text*** finaliza con la subcadena de ***text*** indicada. La comparación de ***text*** no distingue mayúsculas y minúsculas.
EsNúmero(*<text>*) *<text>* **es un número**	Devuelve un valor booleano que indica si el valor indicado de ***text*** representa un número válido.
Longitud(*<text>*)	Devuelve la longitud de caracteres del valor indicado de ***text***.
ComienzaCon (*<text>*, *<sub-*	Devuelve un valor booleano que indica si el valor indicado de ***text*** comienza con la subcadena de ***text*** indicada. La comparación de ***text*** no distingue mayúsculas y minúsculas.

Sintaxis	Descripción
string>) *<text>* **comienza con** *<substring>*	
Subcadena (*<text>*, *<offset>*, *<length>*)	Devuelve la subcadena del ***text*** que comienza en el desfase indicado, que corresponde a la longitud de caracteres especificada. Si se alcanza el final de la cadena, se devolverá un número menor de caracteres.
Texto(*<number>*) **Texto**(*<date>*) **Texto**(*<datetime>*) **Texto** (*<timeOfDay>*)	Convierte el número especificado o el ***date attribute*** en un valor de ***text***.

Funciones de entidad y relación(English)

Sintaxis	Descripción
Para(*<relationship>*, *<Exp>*) **en el caso de** *<ent>*, *<attr>* *<val>*, **en el caso de** *<ent>*	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a uno", "Muchos a uno" o "Muchos a muchos", en la que sólo hay una condición.
ParaÁmbito(*<relationship>*, *<alias>*) **ParaÁmbito**(*<relationship>*) **en el caso de** *<ent>* **en el caso de** *<ent>* **(**<alias>**)**	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a uno", "Muchos a uno" o "Muchos a muchos", en la que hay una condición o varias.
ParaTodo(*<relationship>*, *<Exp>*) **para cada uno de** *<ent>*, *<attr>* **para cada una de** *<ent>*, *<attr>* **para todos** *<ent>*, *<attr>* **para todas** *<ent>*, *<attr>*	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a muchos" o de "Muchos a muchos", cuando es necesario determinar si todos los miembros del grupo de ***entity*** de destino deben cumplir la regla. Esta forma se utiliza cuando en la regla sólo hay una condición.
ParaTodoslosÁmbitos(*<relationship>*) **ParaTodoslosÁmbitos**(*<relationship>*, *<alias>*) **para todos** *<ent>* **para todas** *<ent>* [**para**] **cada uno de** *<ent>*	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a muchos" o de "Muchos a muchos", cuando es necesario determinar si todos los miembros del grupo de ***entity*** de destino deben cumplir la regla. Esta forma se utiliza cuando en la regla hay una condición o varias.

Sintaxis	Descripción
[**para**] **cada una de** *<ent>* [**para**] **todos** *<ent>* **(**<alias>**)** [**para**] **todas** *<ent>* **(**<alias>**)** [**para**] **cada uno de** *<ent>* **(**<alias>**)** [**para**] **cada una de** *<ent>* **(**<alias>**)**	
Existe(*<relationship>*, *<Exp>*) **por lo menos uno de** *<ent>***,** *<attr>* **por lo menos una de** *<ent>***,** *<attr>* **como mínimo uno de** *<ent>***,** *<attr>* **como mínimo una de** *<ent>***,** *<attr>*	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a muchos" o de "Muchos a muchos", cuando es necesario determinar si algún miembro del grupo de ***entity*** de destino debe cumplir la regla. Esta forma se utiliza cuando en la regla sólo hay una condición.
ExisteÁmbito(*<relationship>*) **ExisteÁmbito**(*<relationship>*, *<alias>*) [**por**] **lo menos uno de** *<ent>* [**por**] **lo menos una de** *<ent>* **como mínimo uno de** *<ent>* **como mínimo una de** *<ent>* [**por**] **lo menos uno de** *<ent>* **(**<alias>**)** [**por**] **lo menos una de** *<ent>* **(**<alias>**)** **como mínimo uno de** *<ent>* **(**<alias>**)** **como mínimo una de** *<ent>* **(**<alias>**)**	Se utiliza para hacer referencia de una ***entity*** a otra ***entity***, en una relación ***relationship*** de "Uno a muchos" o de "Muchos a muchos", cuando es necesario determinar si algún miembro del grupo de ***entity*** de destino debe cumplir la regla. Esta forma se utiliza cuando en la regla hay una condición o varias.
EsMiembroDe(*<target>*, *<relationship>*) **EsMiembroDe**(*<target>*, *<alias>*, *<relationship>*) <ent-target> **est un membre de** *<ent>* <ent-target> **(**<alias>**) est un membre de** *<ent>* **IsMemberOf(**<ent-target>**,**	Se utiliza como conclusión para inferir que una instancia de ***entity*** es miembro de ***relationship***. Se utiliza como condición para probar que una instancia de ***entity*** es el destino de una ***relationship***, para el cual una segunda instancia de ***entity*** es el origen.

Sintaxis	Descripción
*<ent>***)** **IsMemberOf(**<ent-target>**,** <alias>**,** *<ent>***)** <ent-target> **es un miembro de** *<relationship>* <ent-target> **(**<alias>**) es un miembro de** *<relationship>*	
NoEsMiembroDe(*<target>*, *<relationship>*)	Se utiliza como condición para comprobar que una instancia ***entity*** no es un destino de ***relationship*** para el cual una segunda instancia de ***entity*** es el origen.
RecuentoInstancia(*<relationship>*) **el número de** *<ent>*	Hace un recuento del número de instancias que existen para una ***entity***.
RecuentoInstanciaSi(*<relationship>*, *<Exp>*) **el número de** *<ent>* **en el caso** [**de**] **que** *<condition>*	Hace un recuento del número de instancias que existen de una ***entity***, para las que un ***entity-level attribute*** concreto tiene un valor concreto.
MáximoInstancias(*<relationship>*, *<number-attr>*) **MáximoInstancias**(*<relationship>*, *<date-attr>*) **MáximoInstancias**(*<relationship>*, *<datetime-attr>*) **MáximoInstancias**(*<relationship>*, *<time-attr>*) **el más grande de** *<attr>* **para todos** *<ent>* **el más grande de** *<attr>* **para todas** *<ent>* **el máximo de** *<attr>* **para todos** *<ent>* **el máximo de** *<attr>* **para todas** *<ent>* **el mayor de** *<attr>* **para todos** *<ent>* **el mayor de** *<attr>* **para todas** *<ent>* **el más reciente de** *<attr>* **para todos** *<ent>* **el más reciente de** *<attr>* **para todas** *<ent>*	Obtiene el valor superior/más reciente de una variable ***entity-level*** para todas las instancias de ***entity***.

Sintaxis	Descripción
la fecha más reciente de *<attr>* **para todos** *<ent>* **la fecha más reciente de** *<attr>* **para todas** *<ent>* *<attr>* **que sea más reciente para todos** *<ent>* *<attr>* **que sea más reciente para todas** *<ent>* *<attr>* **que es el más grande para todos** *<ent>* *<attr>* **que es la más grande para todos** *<ent>* *<attr>* **que es el más grande para todas** *<ent>* *<attr>* **que es la más grande para todas** *<ent>* *<attr>* **que es el máximo para todos** *<ent>* *<attr>* **que es la máxima para todos** *<ent>* *<attr>* **que es el máximo para todas** *<ent>* *<attr>* **que es la máxima para todas** *<ent>* *<attr>* **que es el mayor para todos** *<ent>* *<attr>* **que es la mayor para todos** *<ent>* *<attr>* **que es el mayor para todas** *<ent>* *<attr>* **que es la mayor para todas** *<ent>*	
MáximoInstanciasSi(*<relationship>*, *<number-attr>*, *<condition>*) **MáximoInstanciasSi**(*<relationship>*, *<date-attr>*, *<condition>*) **MáximoInstanciasSi**(*<rela-*	Obtiene el valor superior/más reciente de una variable de ***entity-level***, para todas las instancias de ***entity***, para las que un ***entity-level attribute*** concreto tiene un valor concreto.

Sintaxis	Descripción
tionship>, *<datetime-attr>*, *<condition>*) **MáximoInstanciasSi**(*<relationship>*, *<time-attr>*, *<condition>*) **el más grande de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más grande de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el máximo de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el máximo de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el mayor de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el mayor de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más reciente de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más reciente de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **la fecha más reciente de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **la fecha más reciente de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más reciente para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más reciente para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el más grande**	

Sintaxis	Descripción
para todos *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la más grande para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el más grande para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la más grande para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el máximo para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la máxima para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el máximo para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la máxima para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el mayor para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la mayor para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el mayor para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la mayor para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>*	
MínimoInstancias(*<relationship>*, *<number-attr>*) **MínimoInstancias**(*<relationship>*, *<date-attr>*) **MínimoInstancias**(*<rela-*	Obtiene el valor inferior/menos reciente de una variable ***entity-level*** para todas las instancias de ***entity***.

Sintaxis	Descripción
tionship>, <*datetime-attr*>) **MínimoInstancias**(<*relationship>*, <*time-attr*>) **el más pequeño de** <*attr*> **para todos** <*ent*> **el más pequeño de** <*attr*> **para todas** <*ent*> **el mínimo de** <*attr*> **para todos** <*ent*> **el mínimo de** <*attr*> **para todas** <*ent*> **el menor de** <*attr*> **para todos** <*ent*> **el menor de** <*attr*> **para todas** <*ent*> **el más temprano de** <*attr*> **para todos** <*ent*> **el más temprano de** <*attr*> **para todas** <*ent*> **la fecha más temprana de** <*attr*> **para todos** <*ent*> **la fecha más temprana de** <*attr*> **para todas** <*ent*> <*attr*> **que sea más temprano para todos** <*ent*> <*attr*> **que sea más temprana para todos** <*ent*> <*attr*> **que sea más temprano para todas** <*ent*> <*attr*> **que sea más temprana para todas** <*ent*> <*attr*> **que es el más pequeño para todos** <*ent*> <*attr*> **que es la más pequeña para todos** <*ent*> <*attr*> **que es el más pequeño para todas** <*ent*> <*attr*> **que es la más pequeña para todas** <*ent*> <*attr*> **que es el mínimo para todos** <*ent*> <*attr*> **que es la mínima para todos** <*ent*>	

Sintaxis	Descripción
<attr> **que es el mínimo para todas** *<ent>* *<attr>* **que es la mínima para todas** *<ent>* *<attr>* **que es el menor para todos** *<ent>* *<attr>* **que es la menor para todos** *<ent>* *<attr>* **que es el menor para todas** *<ent>* *<attr>* **que es la menor para todas** *<ent>*	
MínimoInstanciasSi(*<relationship>*, *<number-attr>*, *<condition>*) **MínimoInstanciasSi**(*<relationship>*, *<date-attr>*, *<condition>*) **MínimoInstanciasSi**(*<relationship>*, *<datetime-attr>*, *<condition>*) **MínimoInstanciasSi**(*<relationship>*, *<time-attr>*, *<condition>*) **el más pequeño de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más pequeño de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el mínimo de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el mínimo de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el menor de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el menor de** *<attr>* **para**	Obtiene el valor inferior/menos reciente de una variable de ***entity-level***, para todas las instancias de ***entity***, para las que un ***entity-level attribute*** concreto tiene un valor concreto.

Sintaxis	Descripción
todas *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más temprano de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **el más temprano de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **la fecha más temprana de** *<attr>* **para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* **la fecha más temprana de** *<attr>* **para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más temprano para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más temprana para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más temprano para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que sea más temprana para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el más pequeño para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la más pequeña para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el más pequeño para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la más pequeña para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el mínimo**	

Sintaxis	Descripción
para todos *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la mínima para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el mínimo para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la mínima para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el menor para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la menor para todos** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es el menor para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>* *<attr>* **que es la menor para todas** *<ent>* **en el caso** [**de**] **que** *<ent-test>*	
TotalInstancias(*<relationship>*, *<number-attr>*) **el total para todos** *<ent>*, *<attr>* **el total para todas** *<ent>*, *<attr>* **la cantidad total para todos** *<ent>*, *<attr>* **la contidad total para todas** *<ent>*, *<attr>* *<attr>* **sumado para todos** *<ent>* *<attr>* **sumada para todos** *<ent>* *<attr>* **sumado para todas** *<ent>* *<attr>* **sumada para todas** *<ent>*	Obtiene la suma de todas las instancias de una variable de ***entity-level***.

Sintaxis	Descripción
TotalInstanciasSi(*<relationship>*, *<number-attr>*, *<condition>*) **el total para todos** *<ent>***,** *<attr>* **solo cuando** *<condition>* **el total para todas** *<ent>***,** *<attr>* **solo cuando** *<condition>* **la cantidad total para todos** *<ent>***,** *<attr>* **solo cuando** *<condition>* **la cantidad total para todas** *<ent>***,** *<attr>* **solo cuando** *<condition>* **el total para todos** *<ent>***,** *<attr>* **en el caso** [**de**] **que** *<condition>* **el total para todas** *<ent>***,** *<attr>* **en el caso** [**de**] **que** *<condition>* **la cantidad total para todos** *<ent>***,** *<attr>* **en el caso** [**de**] **que** *<condition>* **la cantidad total para todas** *<ent>***,** *<attr>* **en el caso** [**de**] **que** *<condition>* *<attr>* **sumado para todos** *<ent>* **en el caso** [**de**] **que** *<condition>* *<attr>* **sumada para todos** *<ent>* **en el caso** [**de**] **que** *<condition>* *<attr>* **sumado para todas** *<ent>* **en el caso** [**de**] **que** *<condition>* *<attr>* **sumada para todas** *<ent>* **en el caso** [**de**] **que** *<condition>*	Obtiene la suma de todas las instancias de una variable de ***entity-level***, para el cual es el caso de la ***entity*** que un booleano de ***entity-level*** específico ***attribute*** es verdadero.
ValorInstanciaSi(*<relationship>*, *<number-attr>*, *<condition>*) **ValorInstanciaSi**(*<rela-*	Obtiene un valor de una instancia única de ***entity***, identificada desde las instancias de destino ***entity*** de ***relationship***, mediante una condición. • Si la condición identifica una instancia de destino única de ***entity***, el valor será el valor calculado respecto a esa instancia de ***entity***.

Sintaxis	Descripción
tionship>, *<text-attr>*, *<condition>*) **ValorInstanciaSi**(*<relationship>*, *<date-attr>*, *<condition>*) **ValorInstanciaSi**(*<relationship>*, *<datetime-attr>*, *<condition>*) **ValorInstanciaSi**(*<relationship>*, *<time-attr>*, *<condition>*)	• Si varias instancias de destino cumplen la condición, se devuelve ***uncertain***. • Si ninguna instancia cumple la condición y se conoce ***relationship***, el valor será ***uncertain***.
InstanciaIgualA (*<instance1>*, *<instance2>*)	Determina si dos instancias de una ***entity*** son la misma instancia.
InstancciaNoIgualA (*<instance1>*, *<instance2>*)	Determina si dos instancias de una ***entity*** no son la misma instancia.
Inferir instancia(*<relationship>*, *<identity>*) *<rel>* **(** *<identity>* **)** (**existe** | **existen**)	Se utiliza como conclusión para inferir que existe una instancia de ***entity*** y es miembro de ***relationship***.

Funciones de razonamiento temporal(English)

Sintaxis	Descripción
RecuentoIntervalosDistintoDe (*<start-date>*, *<end-date>*, *<variable>*) **RecuentoIntervalosDistintoDe** (*<start-date>*, *<end-date>*, *<condition>*)	Hace un recuento del número de valores distintos conocidos para la variable, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
RecuentoIntervalosDistintoSi (*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Hace un recuento del número de valores distintos conocidos para la variable, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), y sólo incluye horas cuando un filtro booleano es verdadero.
TotalIntervaloDiario(*<start-date>*, *<end-date>*, *<number-attr>*)	Calcula la suma de una moneda o variable numérica, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida). Se supone que ***attribute*** es una cantidad diaria.
TotalIntervaloDiarioSi(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Calcula la suma de todos los valores diarios de una moneda o variable numérica, comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), y sólo incluye horas cuando una condición es verdadera.
MáximoIntervalos(*<start-date>*,	Selecciona el valor máximo de una variable, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).

Sintaxis	Descripción
<end-date>, *<number-attr>*) **MáximoIntervalos**(*<start-date>*, *<end-date>*, *<date-attr>*) **MáximoIntervalos**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MáximoIntervalos**(*<start-date>*, *<end-date>*, *<time-attr>*)	
MáximoIntervalosSi(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MáximoIntervalosSi**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MáximoIntervalosSi**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MáximoIntervalosSi**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Selecciona el valor máximo de una variable, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), y sólo incluye horas cuando una condición es verdadera.
MínimoIntervalos(*<start-date>*, *<end-date>*, *<number-attr>*) **MínimoIntervalos**(*<start-date>*, *<end-date>*, *<date-attr>*) **MínimoIntervalos**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **MínimoIntervalos**(*<start-date>*, *<end-date>*, *<time-attr>*)	Selecciona el valor mínimo de una variable, comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
MínimoIntervalosSi(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **MínimoIntervalosSi**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **MínimoIntervalosSi**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **MínimoIntervalosSi**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Selecciona el valor mínimo de una variable, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), y sólo incluye horas cuando una condición es verdadera.
MediaPonderadaIntervalo(*<start-date>*, *<end-date>*, *<number-attribute>*)	Calcula el valor medio de una moneda o variable numérica, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), ponderado por el período de tiempo al que se aplica cada valor.
MediaPonderadaIntervaloSi(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Calcula el valor medio de una moneda o variable numérica, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida), y sólo incluye horas cuando una condición booleana es verdadera (ponderada por el per-

Sintaxis	Descripción
	íodo de tiempo al que se aplica cada valor y donde el filtro es verdadero).
IntervaloSiempre(*<start-date>*, *<end-date>*, *<condition>*)	Devuelve un valor verdadero, sólo si una condición booleana es verdadera en todas las horas, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
IntervaloAlMenosDías(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Devuelve un valor verdadero, sólo si una condición booleana es verdadera al menos para el número de días especificado (no necesariamente consecutivos), en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
DíasConsecutivosIntervalo(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Devuelve un valor verdadero, sólo si una condición booleana es verdadera al menos para un número indicado de días consecutivos, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
IntervaloAlgunasVeces(*<start-date>*, *<end-date>*, *<condition>*)	Devuelve un valor verdadero sólo si una condición booleana es siempre verdadera, en el intervalo comprendido entre la ***date*** de inicio (incluida) y la ***date*** de finalización (excluida).
ValorEn(*<date>*, *<value>*)	Devuelve el valor del ***attribute*** indicado en la ***date*** especificada.
AlÚltimo(*<date>*, *<condition>*)	Devuelve la ***date*** en la que una condición booleana fue verdadera al final, buscan hacia atrás desde una ***date*** especificada (incluida).
AlSiguiente(*<date>*, *<condition>*)	Devuelve la ***date*** en la que una condición booleana será verdadera, buscando hacia atrás desde una ***date*** especificada (incluida).
Más Reciente()	Devuelve un valor de ***date*** equivalente a la más reciente ***date*** posible, con el propósito de garantizar que ***date*** sea más reciente que otra ***date*** que pueda tener una ***date attribute*** o que una expresión pueda dar como resultado.
Menos Reciente()	Devuelve un valor de ***date*** equivalente a la menos reciente posible ***date***, con el propósito de garantizar que ***date*** sea más reciente que otra ***date*** que pueda tener una ***date attribute*** o que una expresión pueda dar como resultado.
TemporalDíasDesde(*<date>*, *<end-date>*)	Devuelve una variable numérica que cambia cada día y que corresponde al número de días completos desde la ***date***.
TemporalSemanasDesde(*<date>*, *<end-date>*)	Devuelve una variable numérica que varía cada semana y que corresponde al número de semanas completas desde la ***date***.
TemporalMesesDesde(*<date>*, *<end-date>*)	Devuelve una variable numérica que varía cada mes y que corresponde al número de meses completos desde la ***date***. Nota: Cuando la ***date*** indicada es posterior al día 28 del mes, y el mes posterior tiene menos días que el mes indicado, el punto de cambio para el mes de aniversario se creará en el último día de ese mes. Por ejemplo, si la ***date*** indicada es el 28, 29, 30 o 31 de enero de 2007, el primer punto de cambio será el 28 de febrero de 2007.
TemporalAñosDesde(*<date>*, *<end-*	Devuelve una variable numérica que varía cada año y que corresponde al número

Sintaxis	Descripción
date>)	de años completos desde la ***date***.
TemporalDíasSiempre(*<days>*, *<condition>*)	Devuelve un ***attribute*** booleano que varía con el tiempo y que será verdadero sólo si una condición booleana es verdadera para todos los días anteriores concretos indicados, excluido el día actual.
TemporalDíasConsecutivos (*<minDays>*, *<days>*, *<condition>*)	Devuelve un ***attribute*** booleano que varía con el tiempo y que será verdadero sólo si una condición booleana es verdadera para, al menos, un número mínimo de días consecutivos, en cualquier momento correspondiente a un número de días anteriores establecido, excluido el día actual.
TemporalDíasAveces(*<days>*, *<condition>*)	Devuelve un ***attribute*** booleano que varía con el tiempo, que será verdadero sólo si una condición booleana es siempre verdadera en un número de días anteriores especificado, sin incluir el día actual.
TemporalDespués(*<date>*)	Devuelve un ***attribute*** booleano que varía con el tiempo y que será verdadero después de una ***date*** y falso en esa fecha y antes de ella.
TemporalAntesDe(*<date>*)	Devuelve un ***attribute*** booleano que varía con el tiempo, que será verdadero antes de una ***date*** y falso en esa fecha y después de ella.
TemporalEn(*<date>*)	Devuelve un ***attribute*** booleano que varía con el tiempo, que será verdadero en una ***date*** y falso tanto antes como después de ella.
TemporalEnODespuésDe(*<date>*)	Devuelve un ***attribute*** booleano que varía con el tiempo, que será verdadero en una ***date*** o después de ella, y falsa antes ella.
TemporalEnOAntesDe(*<date>*)	Devuelve un ***attribute*** booleano que varía con el tiempo, que será verdadero en una ***date*** y antes de ella, y falso después de ella.
TemporalDesdeFechaInicio(*<relationship>*, *<date>*, *<value>*)	Devuelve un temporal único ***attribute*** (en el nivel de ***entity*** de origen) desde una ***relationship*** y un valor de ***attribute*** en las entidades, con valores que entrarán en vigor desde una ***date attribute*** de inicio.
TemporalDesdeFechadeFinalización (*<relationship>*, *<date>*, *<value>*)	Devuelve un temporal único ***attribute*** (en el nivel de ***entity*** de origen) desde una ***relationship*** y un valor de ***attribute*** en las entidades, con valores que entrarán en vigor hasta un final ***date attribute***.
TemporalDesdeRango (*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Devuelve un temporal único ***attribute*** (en el nivel de ***entity*** de origen) desde una ***relationship*** y un valor de ***attribute*** en las entidades, con valores que entrarán en vigor desde una ***date attribute*** de inicio (incluida) hasta un ***date attribute*** de finalización (excluida). El valor será ***uncertain*** si caduca antes del comienzo siguiente ***date***.
TemporalEsDíaSemana(*<startdate>*, *<enddate>*)	Devuelve un valor verdadero en las fechas que son días de la semana, y uno falso en las fechas que son fines de semana desde la ***date*** de inicio especificada (incluida), hasta la ***date*** de finalización (excluida). Devuelve ***uncertain*** fuera del rango de ***date***.

Sintaxis	Descripción
TemporalUnaVezPorMes (*<startdate>*, *<enddate>*, *<dayofmonth>*)	Devuelve un valor verdadero, si el día es igual al parámetro de día del mes y falso en todos los demás días del mes, desde la ***date*** de inicio especificada (incluida), hasta la ***date*** de finalización (excluida). Devuelve ***uncertain*** fuera del rango de ***date***. Cuando el día del mes supera el número de días del mes actual, el valor será verdadero en el último día de ese mes, de modo que la función devuelva un valor que será verdadero exactamente un día por mes.

Funciones de eventos de validación(English)

Sintaxis	Descripción
Error (*<text>*)	Un evento de error se utiliza para transferir un mensaje al usuario e impedirle que continúe con una investigación, hasta que deje de aplicarse la condición que ha disparado el error.
Advertencia (*<text>*)	Un evento de advertencia se utiliza para transferir un mensaje al usuario, pero permitiéndole continuar, a pesar de la condición que ha disparado la advertencia.

Funciones anticuadas(English)

Sintaxis	Descripción
LlamarFunciónPersonalizada (*<A>*, *<B>*)	Devuelve el resultado de una llamada externa a una biblioteca de códigos. La biblioteca de códigos debe indicarse al motor de determinaciones, para que la llamada a la función personalizada sea correcta.

Logiska operatorer(English)

Syntax	Beskrivning
om	Valfri term som eventuellt visas i slutet av en slutsatsrad som har ett efterföljande bevis
och	Logisk konjunktion mellan två ***attributes***
eller	Logisk disjunktion mellan två ***attributes***
antingen **ett av** **en av** **någon** **något** **några** **alls** **minst en av de följande är kor-**	Grupperingselement som används med disjunktioner där två eller fler ***attributes*** måste grupperas

Syntax	Beskrivning
rekt **minst ett av de följande är korrekt** **ett av de följande är korrekt** **någon av de följande är tillfredsställande** **något av de följande är tillfredsställande** **några av de följande är tillfredsställande** **någon av de följande är korrekt** **något av de följande är korrekt** **några av de följande är korrekt**	
båda **både** **all** **allt** **alla** **allt som följer är korrekt** **allt som följer är tillfredsställande**	Grupperingselement som används med konjunktioner där två eller fler ***attributes*** måste grupperas
annars **i annat fall**	Term som visas i slutet av en tabellregel för att ange annars-delsatsen
är	Term som används i en förklaringspost mellan den förkortade frasen och den fullständiga ***attribute text***

Logiska funktioner(English)

Syntax	Beskrivning
det är inte sant att *<expr>*	Operatorn som används för att returnera sant om ***attribute*** har ett värde som är falskt
<var> **är säker** **det är säkert antingen** [**eller inte**]*<expr>*	Operatorn som används för att returnera sant om ***attribute*** har ett värde som inte är ***uncertain***
<var> **är osäker** *<var>* **är inte säker** **det är osäkert att** *<expr>* **det är osäkert om** [**eller inte**]*<expr>* **det är inte säkert att** *<expr>* **osäkert**	Operatorn som används för att returnera sant om värdet för ***attribute*** är ***uncertain***

Syntax	Beskrivning
<var> **är bekant** *<var>* **är känd** *<var>* **är känt** *<var>* **är för närvarande bekant** *<var>* **är för närvarande känd** *<var>* **är för närvarande känt** **det är bekant antingen** [**eller inte**]*<expr>* **det är för närvarande bekant om** [**eller inte**]*<expr>*	Operatorn som används för att returnera sant om ***attribute*** har något värde
<var> **är** [**för närvarande**] **obekant** *<var>* **är** [**för närvarande**] **okänd** **det är** [**för närvarande**] **okänd att** [**eller inte**]*<expr>*	Operatorn som används för att returnera sant om ***attribute*** inte har något värde

Logiska konstanter(English)

Syntax	Beskrivning
sant	Värdet sant för konstant som används för tabellregler.
falskt	Värdet falskt för konstant som används för tabellregler.
osäkert	Värdet ***uncertain*** för konstant som används för tabellregler.

Jämförelseoperatorer(English)

Syntax	Beskrivning
*<lhs>***<***<rhs>* *<lhs>* **är lägre än** *<rhs>* *<lhs>* **är tidigare än** *<rhs>*	Mindre än Obs! Det finns ingen form för naturligt språk när den här operatorn används med numeriska värden och valutavärden.
<lhs> **>** *<rhs>* *<lhs>***är större än** *<rhs>* *<lhs>* **är senare än** *<rhs>*	Större än Obs! Det finns ingen form för naturligt språk när den här operatorn används med numeriska värden och valutavärden.
*<lhs>***<=***<rhs>*	Mindre än eller lika med
<lhs> **>=** *<rhs>*	Större än eller lika med

Syntax	Beskrivning
*<lhs>***=***<rhs>*	Lika med
<lhs> **<>** *<rhs>*	Inte lika med

Numeriska funktioner(English)

Syntax	Beskrivning
Tal(*<numText>*)	Konvertera den angivna strängen till ett talvärde
<x> **+** *<y>*	Addition
<x> **-** *<y>*	Subtraktion
<lhs> ***** *<rhs>*	Multiplikation
<lhs> **/** *<rhs>*	Division
<lhs> **** *<rhs>*	Heltalsdivision
<lhs> **modulo** *<rhs>*	Återstod efter heltalsdivision
Max(*<x>*, *<y>*) **Max**(*<date/time/datetime1>*, *<date/time/datetime2>*) **större än** *<val1>* **och** *<val2>* **senast av** *<val1>* **och** *<val2>*	Returnerar det största värdet av två värden
Min(*<x>*, *<y>*) **Min**(*<date/time/datetime1>*, *<date/time/datetime2>*) **minst av** *<val1>* **och** *<val2>* **tidigast av** *<val1>* **och** *<val2>*	Returnerar det minsta värdet av två värden
Xy(*<x>*, *<y>*)	x upphöjt till y
Ex(*<x>*)	Konstanten e upphöjt till x
Abs(*<x>*)	Absolut värde för x
Ln(*<x>*)	Naturlig logaritm för x
Log(*<x>*)	Logaritmbasen 10 för x
Rot(*<x>*) **kvadratroten av** *<val>*	Kvadratrot ur x
Avrunda(*<x>*, *<n>*) *<x>* **avrunda till** *<n>* **decimaler**	Avrundar x till n decimaler
Kapa(*<x>*, *<n>*)	x kapat till n decimaler
Sin(*<x>*)	Sinus för x

Syntax	Beskrivning
Cos(<*x*>)	Cosinus för x
Tan(<*x*>)	Tangens för x
Asin(<*x*>)	Arcus sinus för x
Acos(<*x*>)	Arcus cosinus för x
Atan(<*x*>)	Arcus tangens för x

Datumfunktioner(English)

Syntax	Beskrivning
AktuelltDatum() **dagens datum**	Returnerar aktuell ***date*** när sessionen påbörjas.
Datum(<*text*>)	Konverterar den angivna strängen till ett värde för ***date***
SkapaDatum(<*year*>, <*month*>, <*day*>)	Returnerar ett ***date*** som har skapats utifrån året, månaden och dagen som har angetts.
ExtraheraDag(<*date/datetime*>)	Returnerar dagkomponenten för ett attribut av typen ***date/datetime attribute***.
ExtraheraMånad(<*date/datetime*>)	Returnerar månadskomponenten för ett attribut av typen ***date/datetime attribute***.
ExtraheraÅr(<*date/datetime*>)	Returnerar årkomponenten för ett attribut av typen ***date/datetime attribute***.
NästaDagIVeckan(<*date/datetime*>, <*day*>) **på måndagen eller den närmast kommande måndagen** <*from-date*> **på tisdagen eller den närmast kommande tisdagen** <*from-date*> **på onsdagen eller den närmast kommande onsdagen** <*from-date*> **på torsdagen eller den närmast kommande torsdagen** <*from-date*> **på fredagen eller den närmast kommande fredagen** <*from-date*> **på lördagen eller den närmast kommande lördagen** <*from-date*> **på söndagen eller den närmast kommande söndagen** <*from-date*>	Returnerar ***date*** för nästa veckodag på eller före/efter ett ***date*** (beroende på syntaxen som används).
NästaDatum(<*date*>, <*day*>,	Returnerar nästa instans för dagen och månaden som angetts efter ett ***date***.

Syntax	Beskrivning
<month>) **sista dagen på året då** *<from-date>* **falls** **det svenska skatteårets startdatum för i år om i dag eller det föregående skatteårets startdatum** *<from-date>*	
LäggtillDagar(*<date/datetime>*, *<num_days>*) **datum** *<num_days>* **dagar efter** *<date>*	Lägger till/tar bort ett antal dagar i/från ***date***. När den kortfattade syntaktiska formen används måste antalet vara ett positivt heltal om dagar ska kunna läggas till i ***date***, eller ett negativt tal om dagar ska kunna tas bort från ***date***.
LäggtillVeckor(*<date/datetime>*, *<num_weeks>*) **datum** *<num_weeks>* **veckor efter** *<date>*	Lägger till ett antal veckor i ***date***. När den kortfattade syntaktiska formen används måste antalet vara ett positivt heltal om veckor ska kunna läggas till i ***date***.
LäggtillMånader(*<date/datetime>*, *<num_months>*) **datum** *<num_months>* **månader efter** *<date>*	Lägger till ett antal månader i ***date***. När den kortfattade syntaktiska formen används måste antalet vara ett positivt heltal om månader ska kunna läggas till i ***date***.
LäggtillÅr(*<date/datetime>*, *<num_years>*) **datum** *<num_years>* **år efter** *<date>*	Lägger till ett antal år i ***date***. När den kortfattade syntaktiska formen används måste antalet vara ett positivt heltal om år ska kunna läggas till i ***date***.
VeckodagAntal(*<date1>*, *<date2>*) **antal veckodagar (inklusive) mellan** *<date1>* **och** *<date2>*	Räknar antalet veckodagar mellan ***date***1 och ***date***2, dvs. antalet dagar som infaller mellan måndag och fredag. Obs! Tidigare ***date*** är inklusive och senare ***date*** är exklusive.
ÅrStart(*<date/datetime>*)	Returnerar ett första ***date*** på året som ett ***date*** infaller i.
ÅrSlut(*<date/datetime>*)	Returnerar ett sista ***date*** på året som ett ***date*** infaller i.
DagSkillnad(*<date/datetime1>*, *<date/datetime2>*) **antal dagar från** *<date1>* **till** *<date2>* **antal dagar (inklusive) från** *<date1>* **till** *<date2>* **antal dagar (exklusive) från** *<date1>* **till** *<date2>*	Returnerar antalet hela dagar mellan ***date/datetime1*** och ***date/datetime2***. Ordningen på de två datumen påverkar inte resultatet.
DagSkillnadInklusive(*<date/datetime1>*, *<date/datetime2>*)	Returnerar antalet hela dagar (inklusive) mellan ***date/datetime1*** och ***date/datetime2***. I den här beräkningen inkluderas båda slutpunkterna. I de fall datumen är samma blir resultatet 1. Ordningen på de två datumen påverkar inte resultatet.
DagSkillnadExklusive(*<date/d-*	Returnerar antalet hela dagar (exklusive) mellan ***date/datetime1*** och ***date/d-***

Syntax	Beskrivning
atetime1>, *<date/datetime2>*)	***atetime2***. I den här beräkningen exkluderas båda slutpunkterna. I de fall datumen är samma blir resultatet 0. Ordningen på de två datumen påverkar inte resultatet.
VeckaSkillnad(*<date/datetime1>*, *<date/datetime2>*) **antal veckor från** *<date1>* **till** *<date2>*	Returnerar tidsåtgången i antal hela veckor mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
VeckaSkillnadInklusive(*<date/datetime1>*, *<date/datetime2>*)	Returnerar tidsåtgången i antal hela veckor inklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
VeckaSkillnadExklusive(*<date/datetime1>*, *<date/datetime2>*)	Returnerar tidsåtgången i antal hela veckor exklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
MånadSkillnad(*<date/datetime1>*, *<date/datetime2>*) **antal månader från** *<date1>* **till** *<date2>*	Returnerar tidsåtgången i antal hela månader mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
MånadSkillnadInklusive(*<date/datetime1>*, *<date/datetime2>*)	Returnerar tidsåtgången i antal hela månader inklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
MånadSkillnadExklusive(*<date/datetime1>*, *<date/datetime2>*)	Returnerar tidsåtgången i antal hela månader exklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
ÅrSkillnad(*<date/datetime1>*, *<date/datetime2>*) **antal år (inklusive) mellan** *<date1>* **och** *<date2>* **antal hela år vilka** *<date2>***är efter** *<date1>*	Returnerar antalet år mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
ÅrSkillnadInklusive (*<date/datetime1>*, *<date/datetime2>*)	Returnerar antalet år inklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.
ÅrSkillnadExklusive (*<date/datetime1>*, *<date/datetime2>*)	Returnerar antalet år exklusive mellan ***date/datetime1*** och ***date/datetime2***. Ordningen för de två datumen påverkar inte resultatet.

Funktioner för klockslag(English)

Syntax	Beskrivning
Klockslag(*<text>*)	Konverterar den angivna strängen till ett klockslag
ExtraheraSekund(*<time/datetime>*)	Returnerar sekundkomponenten för ett attribut av typen ***timeofday/datetime attribute***.

Syntax	Beskrivning
ExtraheraMinut (*<time/datetime>*)	Returnerar minutkomponenten för ett attribut av typen ***timeofday/datetime attribute***.
ExtraheraTimme(*<time/datetime>*)	Returnerar timkomponenten för ett attribut av typen ***timeofday/datetime attribute***.

Funktioner för datum och tid(English)

Syntax	Beskrivning
AktuelltDatumTid()	Returnerar aktuell ***date*** och tid när sessionen påbörjas.
DatumTid(*<text>*)	Konverterar den angivna strängen till ett värde för ***datetime***
SammanfogaDatumTid (*<date>*, *<time>*)	Ställer in ***date*** genom att koppla ***date*** och klockslaget.
SekundSkillnad(*<datetime1>*, *<datetime2>*) **SekundSkillnad** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet sekunder mellan ***datetime1*** och ***datetime2***.
SekundSkillnadInklusive (*<datetime1>*, *<datetime2>*) **SekundSkillnadInklusive** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet sekunder inklusive mellan ***datetime1*** och ***datetime2***.
SekundSkillnadExklusive (*<datetime1>*, *<datetime2>*) **SekundSkillnadExklusive** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet sekunder exklusive mellan ***datetime1*** och ***datetime2***.
MinutSkillnad(*<datetime1>*, *<datetime2>*) **MinutSkillnad** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet minuter mellan ***datetime1*** och ***datetime2***.
MinutSkillnadInklusive (*<datetime1>*, *<datetime2>*) **MinutSkillnadInklusive**	Returnerar antalet minuter inklusive mellan ***datetime1*** och ***datetime2***.

Syntax	Beskrivning
(*<timeOfDay1>*, *<timeOfDay2>*)	
MinutSkillnadExklusive (*<datetime1>*, *<datetime2>*) **MinutSkillnadExklusive** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet minuter exklusive mellan ***datetime1*** och ***datetime2***.
TimmeSkillnad(*<datetime1>*, *<datetime2>*) **TimmeSkillnad** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet timmar mellan ***datetime1*** och ***datetime2***.
TimmeSkillnadInklusive (*<datetime1>*, *<datetime2>*) **TimmeSkillnadInklusive** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet timmar inklusive mellan ***datetime1*** och ***datetime2***.
TimmeSkillnadExklusive (*<datetime1>*, *<datetime2>*) **TimmeSkillnadExklusive** (*<timeOfDay1>*, *<timeOfDay2>*)	Returnerar antalet timmar exklusive mellan ***datetime1*** och ***datetime2***.
ExtraheraDatum(*<datetime>*)	Extraherar ***date*** från ***datetime attribute***.
ExtraheraKlockslag(*<datetime>*)	Extraherar klockslaget från ett attribut av typen ***datetime attribute***. Detta kan användas för att ställa in värdet för ett attribut av typen ***timeofday attribute*** på tiden som regeln utförs genom att tiden extraheras från aktuellt ***date*** och tid.
LäggtillTimmar(*<datetime>*, *<num_hours>*) **LäggtillTimmar** (*<timeOfDay>*, *<num_hours>*)	Lägger till ett antal timmar i ***date***.
LäggtillMinuter(*<datetime>*, *<num_minutes>*) **LäggtillMinuter** (*<timeOfDay>*, *<num_*	Lägger till ett antal minuter i ***date***.

Syntax	Beskrivning
minutes>)	
LäggtillSekunder(*<datetime>*, *<num_seconds>*) **LäggtillSekunder** (*<timeOfDay>*, *<num_seconds>*)	Lägger till ett antal sekunder i ***date***.

Textfunktioner(English)

Syntax	Beskrivning
<text1> **&** *<text2>*	Kombinerar ***text1*** med ***text2*** osv. så att ett enstaka värde för ***text*** skapas. Obs! Du kan använda variabler av valfri typ. Värdena formateras med hjälp av formateraren som är installerad i regelsessionen.
teckensträng av *<text1>* **&** *<text2>*	Kombinerar ***text1*** med ***text2*** osv. så att ett enstaka värde för ***text*** skapas. Obs! Du kan använda variabler av valfri typ. Värdena formateras med hjälp av formateraren som är installerad i regelsessionen.
Innehåller(*<text>*, *<substring>*) *<text>* **innehåller** *<substring>*	Returnerar ett booleskt värde som anger om det angivna värdet för ***text*** innehåller den angivna delsträngen för ***text***. Jämförelsen för ***text*** är skiftlägesokänslig.
SlutarMed(*<text>*, *<substring>*) *<text>* **slutar med** *<substring>*	Returnerar ett booleskt värde som anger om det angivna värdet för ***text*** slutar med den angivna delsträngen för ***text***. Jämförelsen för ***text*** är skiftlägesokänslig.
ÄrTal(*<text>*) *<text>* **är ett tal**	Returnerar ett booleskt värde som anger om det angivna värdet för ***text*** motsvarar ett giltigt tal.
Längd(*<text>*)	Returnerar teckenlängden för det angivna värdet för ***text***.
BörjarMed(*<text>*, *<substring>*) *<text>* **startar med** *<substring>*	Returnerar ett booleskt värde som anger om det angivna värdet för ***text*** börjar med den angivna delsträngen för ***text***. Jämförelsen för ***text*** är skiftlägesokänslig.
Delsträng(*<text>*, *<offset>*, *<length>*)	Returnerar delsträngen för ***text*** som börjar vid den angivna förskjutningen, dvs. den angivna längden i tecken. Färre tecken returneras om slutet på strängen nås.
Text(*<number>*) **Text**(*<date>*) **Text**(*<datetime>*) **Text**(*<timeOfDay>*)	Konvertera det angivna talet eller ***date attribute*** till ett värde för ***text***.

Funktioner för enhet och relation(English)

Syntax	Beskrivning
För(*<relationship>*, *<Exp>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-en, många-till-en eller många-till-många där det endast finns ett villkor.
FörOmfattning(*<relationship>*, *<alias>*) **FörOmfattning**(*<relationship>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-en, många-till-en eller många-till-många där det finns ett eller fler villkor.
FörAlla(*<relationship>*, *<Exp>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-många eller många-till-många när du måste fastställa om alla medlemmar i målgruppen för ***entity*** måste uppfylla regelvillkoren. Den här formen används när det endast finns ett villkor i regeln.
FörAllaOmfattning(*<relationship>*) **FörAllaOmfattning**(*<relationship>*, *<alias>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-många- eller många-till-många när du måste fastställa om alla medlemmar i målgruppen för ***entity*** måste uppfylla regelvillkoren. Den här formen används när det endast finns ett eller fler villkor i regeln.
Befintlig(*<relationship>*, *<Exp>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-många eller många-till-många när du måste fastställa om några medlemmar i målgruppen för ***entity*** måste uppfylla regelvillkoren. Den här formen används när det endast finns ett villkor i regeln.
FinnsOmfattning(*<relationship>*) **FinnsOmfattning**(*<relationship>*, *<alias>*)	Används för att referera från en ***entity*** till en annan ***entity*** i en ***relationship*** av typen en-till-många- eller många-till-många när du måste fastställa om några medlemmar i målgruppen för ***entity*** måste uppfylla regelvillkoren. Den här formen används när det finns ett eller fler villkor i regeln.
ÄrMedlemAv(*<target>*, *<relationship>*) **ÄrMedlemAv**(*<target>*, *<alias>*, *<relationship>*)	Används som en slutsats för att härleda att en instans för ***entity*** är en medlem i en ***relationship***. Används som ett villkor för att testa att en instans för ***entity*** är målet för en ***relationship*** där den andra instansen för ***entity*** är källan.
ÄrInteMedlemAv(*<target>*, *<relationship>*)	Används som ett villkor för att testa att en instans för ***entity*** inte är målet för en ***relationship*** där den andra instansen för ***entity*** är källan.
InstansAntal (*<relationship>*) **antalet** *<ent>*	Räknar antalet instanser som finns för en ***entity***.
InstansAntalOm(*<relationship>*, *<Exp>*) **antalet** *<ent>* **om fallet är** *<condition>*	Räknar antalet instanser det finns för en ***entity*** som ett specifikt ***entity-level attribute*** har ett specifikt värde för.
InstansMax(*<relationship>*,	Hämtar det högsta/senaste värdet för en variabel på ***entity-level*** för alla instanser för ***entity***.

Syntax	Beskrivning
<number-attr>) **InstansMax**(*<relationship>*, *<date-attr>*) **InstansMax**(*<relationship>*, *<datetime-attr>*) **InstansMax**(*<relationship>*, *<time-attr>*) *<date-attr>* **vilket är det senaste av alla** *<ent>* *<max-attr>* **vilket är det högsta av alla** *<ent>* **senast av alla** *<ent-attr>* **högst av alla** *<ent-attr>* **senast av alla** *<ent-attr>* **högst av** *<attr>* **för alla** *<ent>* **högst av alla** *<attr>* **för** *<ent>* **senast av alla** *<attr>* **för** *<ent>*	
InstansMaxOm(*<relationship>*, *<number-attr>*, *<condition>*) **InstansMaxOm**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansMaxOm**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstansMaxOm**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **vilket är det senaste av alla** *<ent>* **om fallet är** *<ent-test>* *<max-attr>***vilket är det högsta av alla** *<ent>* **om fallet är** *<ent-test>* **senast av alla** *<ent-attr>* **om fallet är** *<ent-test>* **högst av alla** *<ent-attr>* **om fallet är** *<ent-test>* **högst av** *<attr>* **för alla**	Hämtar det högsta/senaste värdet för en variabel på ***entity-level*** för alla instanser för ***entity*** som ett specifikt ***entity-level attribute*** har ett specifikt värde för.

Syntax	Beskrivning
<ent> **om fallet är** *<ent-test>*	
InstansMin(*<relationship>*, *<number-attr>*) **InstansMin**(*<relationship>*, *<date-attr>*) **InstansMin**(*<relationship>*, *<datetime-attr>*) **InstansMin**(*<relationship>*, *<time-attr>*) *<date-attr>* **vilket är det tidigaste av alla** *<ent>* *<attr>* **vilket är det lägsta av alla** *<ent>* **lägst av alla** *<ent-attr>* **tidigast av alla** *<ent-attr>* **lägst av alla** *<attr>* **för** [**alla**]*<ent>* **tidigast av alla** *<attr>* **för** [**alla**]*<ent>* **tidigast av alla** *<attr>* **för** *<ent>*	Hämtar det lägsta/tidigaste värdet för en variabel på ***entity-level*** för alla instanser för ***entity***.
InstansMinOm(*<relationship>*, *<number-attr>*, *<condition>*) **InstansMinOm**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansMinOm**(*<relationship>*, *<datetime-attr>*, *<condition>*) **InstansMinOm**(*<relationship>*, *<time-attr>*, *<condition>*) *<date-attr>* **vilket är det tidigaste av alla** *<ent>* **om fallet är** *<ent-test>* *<num-attr>* **vilket är det lägsta av alla** *<ent>* **om fallet är** *<ent-test>* **lägst av alla** *<ent-attr>* **om fallet är** *<ent-test>*	Hämtar det lägsta/tidigaste värdet för en variabel på ***entity-level*** för alla instanser för ***entity*** som ett specifikt ***entity-level attribute*** har ett specifikt värde för.

Syntax	Beskrivning
tidigast av alla *<ent-attr>* **om fallet är** *<ent-test>* **lägst av alla** *<attr>* **för** *<ent>* **om fallet är** *<ent-test>* **tidigast av alla** *<attr>* **för** *<ent>* **om fallet är** *<ent-test>*	
InstansSumma(*<relationship>*, *<number-attr>*) *<num-attr>* **summan för alla** *<ent>* **summan av** [**alla**]*<ent-attr>* **den sammanlagda summan för allt** *<ent-attr>* **totalt för alla** *<ent>*, *<attr>*	Hämtar summan av alla instanser för en variabel på ***entity-level***.
InstansSummaOm(*<relationship>*, *<number-attr>*, *<condition>*) *<num-attr>* **summman för alla** *<ent>* **om fallet är** *<ent-test>* **den sammanlagda summan för allt** *<ent-attr>* **om fallet är** *<condition>* **det sammanlagda antalet av** [**alla**]*<ent-attr>* **om fallet är** *<condition>* **totalt för alla** *<ent>*, *<attr>* **om fallet är** *<condition>*	Hämtar summan av alla instanser för en variabel på ***entity-level*** som värdet är sant för, för ***entity*** som ett specifikt booleskt ***attribute*** på ***entity-level*** är sant.
InstansVärdeOm(*<relationship>*, *<number-attr>*, *<condition>*) **InstansVärdeOm**(*<relationship>*, *<text-attr>*, *<condition>*) **InstansVärdeOm**(*<relationship>*, *<date-attr>*, *<condition>*) **InstansVärdeOm**(*<relationship>*, *<datetime-attr>*, *<condition>*)	Hämtar ett värde från en unik instans för ***entity*** som har identifierats av ett villkor utifrån målinstanserna för ***entity*** för ***relationship***. • Om villkoret identifierar en enstaka målinstans för ***entity*** används det värde som har beräknats mot den instansen för ***entity***. • Om fler än en målinstans uppfyller villkoret returneras ***uncertain***. • Om inga målinstanser uppfyller villkoret och ***relationship*** är känd är värdet ***uncertain***.

Syntax	Beskrivning
InstansVärdeOm(*<relationship>*, *<time-attr>*, *<condition>*)	
InstansLikaMed (*<instance1>*, *<instance2>*)	Fastställer om två instanser för en ***entity*** är samma instans.
InstansEjLikaMed (*<instance1>*, *<instance2>*)	Fastställer om två instanser för en ***entity*** inte är samma instans.
InferInstance(*<relationship>*, *<identity>*)	Används som en slutsats för att härleda att en instans för ***entity*** finns och att den är en medlem i en ***relationship***.

Tidsbestämda slutledningsfunktioner(English)

Syntax	Beskrivning
IntervallAntalUnikt(*<start-date>*, *<end-date>*, *<variable>*) **IntervallAntalUnikt**(*<start-date>*, *<end-date>*, *<condition>*)	Räknar antalet kända unika värden för variabeln i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive).
IntervallAntalUniktOm(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Räknar antalet kända unika värden för variabeln i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive). Tider inkluderas bara när ett booleskt filter är sant.
IntervallDagligSumma(*<start-date>*, *<end-date>*, *<number-attr>*)	Beräknar summan av en valuta- eller talvariabel i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive). ***attribute*** antas vara en daglig kvantitet.
IntervallDagligSummaOm(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Beräknar summan av alla dagliga värden för en valuta- eller talvariabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive). Tider inkluderas bara när ett villkor är sant.
IntervallMax(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervallMax**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervallMax**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervallMax**(*<start-date>*, *<end-date>*, *<time-attr>*)	Väljer maxvärdet för en variabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive).
IntervallMaxOm(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervallMaxOm**(*<start-date>*,	Väljer maxvärdet för en variabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive). Tider inkluderas bara när ett villkor är sant.

Syntax	Beskrivning
<end-date>, *<date-attr>*, *<condition>*) **IntervallMaxOm**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervallMaxOm**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	
IntervallMin(*<start-date>*, *<end-date>*, *<number-attr>*) **IntervallMin**(*<start-date>*, *<end-date>*, *<date-attr>*) **IntervallMin**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **IntervallMin**(*<start-date>*, *<end-date>*, *<time-attr>*)	Väljer min.värdet för en variabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive).
IntervallMinOm(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **IntervallMinOm**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **IntervallMinOm**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **IntervallMinOm**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Väljer min.värdet för en variabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive). Tider inkluderas bara när ett villkor är sant.
IntervallViktadMedelvärde(*<start-date>*, *<end-date>*, *<number-attribute>*)	Beräknar det genomsnittliga värdet för en valuta- eller talvariabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive), viktat utifrån tidsintervallet som varje värde tillämpas på.
IntervallViktadMedelvärdeOm (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Beräknar det genomsnittliga värdet för en valuta- eller talvariabel i intervallet från ett startdatum ***date*** (inklusive) till ett slutdatum ***date*** (exklusive). Tider inkluderas bara när ett booleskt villkor är sant (viktat utifrån tidsintervallet som varje värde tillämpas på och där filtret är sant).
IntervallAlltid(*<start-date>*, *<end-date>*, *<condition>*)	Returnerar sant om, och endast om, ett booleskt villkor alltid är sant i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive).
IntervallMinstDagar(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerar sant om, och endast om, ett booleskt villkor är sant åtminstone för det angivna antalet dagar (inte nödvändigtvis på varandra följande) i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive).
IntervallPåvarandraföljandeDagar (*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Returnerar sant om, och endast om, ett booleskt villkor är sant åtminstone för ett angivet antal på varandra följande dagar i intervallet från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive).
IntervallIbland(*<start-date>*, *<end-*	Returnerar sant om, och endast om, ett booleskt villkor någonsin är sant i intervallet

Syntax	Beskrivning
date>, *<condition>*)	från startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive).
VärdePå(*<date>*, *<value>*)	Returnerar värdet för det angivna ***attribute*** på det angivna ***date***.
NärSenaste(*<date>*, *<condition>*)	Returnerar ett ***date*** då ett booleskt villkor senast var sant, sett bakåt från (och inklusive) ett angivet ***date***.
NärNästa(*<date>*, *<condition>*)	Returnerar ett ***date*** då ett booleskt villkor nästa gång är sant, sett framåt från (och inklusive) ett angivet ***date***.
Senaste()	Returnerar ett värde för ***date*** som motsvarar senast möjliga ***date*** - nämligen ett ***date*** som garanterat infaller efter övriga ***date*** som ett ***date attribute*** eventuellt använder eller som ett uttryck eventuellt utvärderar till.
Tidigaste()	Returnerar ett värde för ***date*** som motsvarar tidigast möjliga ***date*** - nämligen ett ***date*** som garanterat infaller före övriga ***date*** som ett ***date attribute*** eventuellt använder eller som ett uttryck eventuellt utvärderar till.
TidDagarSedan(*<date>*, *<end-date>*)	Returnerar en talvariabel som varierar varje dag och motsvarar antalet hela dagar sedan ***date***.
TidVeckorSedan(*<date>*, *<end-date>*)	Returnerar en talvariabel som varierar varje vecka och som motsvarar antalet hela veckor sedan ***date***.
TidMånaderSedan(*<date>*, *<end-date>*)	Returnerar en talvariabel som varierar varje månad och som motsvarar antalet hela månader sedan ***date***. Obs! När angivet ***date*** infaller efter den 28:e dagen i månaden, och en efterföljande månad har färre dagar än den angivna månaden, skapas ändringspunkten för årsmånaden på den sista dagen i den månaden. Om angivet ***date*** t.ex. är 28, 29, 30 eller 31 januari 2007 blir den första ändringspunkten 28 februari 2007.
TidÅrSedan(*<date>*, *<end-date>*)	Returnerar en talvariabel som varierar varje år och som motsvarar antalet hela år sedan ***date***.
TidAlltidDagar(*<days>*, *<condition>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant om, och endast om, ett booleskt villkor är sant för alla av de angivna föregående dagarna, inte inklusive den aktuella dagen.
TidPåvarandraföljandeDagar (*<minDays>*, *<days>*, *<condition>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant om, och endast om, ett booleskt villkor är sant för åtminstone ett minsta antal på varandra följande dagar vid någon tidpunkt inom föregående angivna antal dagar, inte inklusive den aktuella dagen.
TidIblandDagar(*<days>*, *<condition>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant om, och endast om, ett booleskt villkor någonsin är sant inom ett angivet antal föregående dagar, inte inklusive den aktuella dagen.
TidEfter(*<date>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant efter ett ***date*** och falskt på och före.

Syntax	Beskrivning
TidFöre(*<date>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant före ett ***date*** och falskt på och efter.
TidPå(*<date>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant på ett ***date*** och falskt före och efter.
TidPåEllerEfter(*<date>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant på eller efter ett ***date*** och falskt före.
TidPåEllerFöre(*<date>*)	Returnerar ett booleskt ***attribute*** som varierar över tiden och är sant på och före ett ***date*** och falskt efter.
TidFrånStartDatum(*<relationship>*, *<date>*, *<value>*)	Returnerar ett enstaka tidsbestämt ***attribute*** (på källnivån för ***entity***) från en ***relationship*** och ett ***attribute*** för värde i enheterna, med värden som gäller fr.o.m. ett startdatum ***date attribute***.
TidFrånSlutDatum(*<relationship>*, *<date>*, *<value>*)	Returnerar ett enstaka tidsbestämt ***attribute*** (på källnivån för ***entity***) från en ***relationship*** och ett ***attribute*** för värde i enheterna, med värden som gäller t.o.m. ett slutdatum ***date attribute***.
TidFrånIntervall(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	Returnerar ett enstaka tidsbestämt ***attribute*** (på källnivån för ***entity***) från en ***relationship*** och ett ***attribute*** för värde i enheterna, med värden som gäller fr.o.m. ett startdatum ***date attribute*** (inklusive) t.o.m. ett slutdatum ***date attribute*** (exklusive). Värdet är ***uncertain*** om det upphör att gälla före nästa startdatum ***date***.
TidÄrVeckodag(*<startdate>*, *<enddate>*)	Returnerar sant för datumen som är veckodagar och falskt för datumen som är helgdagar fr.o.m. det angivna startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive). Returnerar ***uncertain*** utanför intervallet för ***date***.
TidEngångPerMånad(*<startdate>*, *<enddate>*, *<dayofmonth>*)	Returnerar sant om dagen är lika med värdet i parametern dag i månaden och falskt för alla övriga dagar i månaden fr.o.m. det angivna startdatumet ***date*** (inklusive) till slutdatumet ***date*** (exklusive). Returnerar ***uncertain*** utanför intervallet för ***date***. När dagen i månaden överskrider antalet dagar i den aktuella månaden blir värdet sant på den sista dagen i den månaden, så att funktionen returnerar ett värde som är sant exakt en dag per månad.

Funktioner för valideringshändelse(English)

Syntax	Beskrivning
Fel (*<text>*)	En felhändelse används för att överföra ett meddelande till användaren och därmed förhindra användaren från att fortsätta undersökningen tills villkoret som utlöste felet inte längre gäller.
Varning (*<text>*)	En varningshändelse används för att överföra ett meddelande till användaren. Användaren tillåts fortsätta trots villkoret som utlöste varningen.

Inaktuella funktioner(English)

Syntax	Beskrivning
AnropaAnpassadFunktion (*<A>*, *<B>*)	Returnerar resultatet för ett externt anrop till ett kodbibliotek. Kodbiblioteket måste finnas tillgängligt för Oracle Determinations Engine om det anpassade funktionsanropet ska kunna utföras.

ตัวเชื่อมต่อเชิงตรรกศาสตร์ (English)

ซินแท็กซ์	คำอธิบาย
ถ้า	คำที่เลือกได้ซึ่งสามารถปรากฏที่ท้ายบรรทัดข้อสรุปที่มีหลักฐานต่อไปนี้
และ	การเชื่อมเชิงตรรกศาสตร์ระหว่างสอง ***attributes***
หรือ	การเลือกเชิงตรรกศาสตร์ระหว่างสอง ***attributes***
รายการใดรายการหนึ่ง **รายการหนึ่งใน** **รายการใดๆ** **อย่างน้อยหนึ่งเงื่อนไขต่อไปนี้เป็นจริง** **ตรงตามเงื่อนไขใดๆ ต่อไปนี้**	การจัดกลุ่มอีลิเมนต์ที่ใช้กับการเลือกโดยมีสอง ***attributes*** หรือมากกว่าที่ต้องจัดกลุ่ม
ทั้งคู่ **ทั้งหมด** **เงื่อนไขทั้งหมดเป็นจริง** **ตรงตามเงื่อนไขทั้งหมดต่อไปนี้**	การจัดกลุ่มอีลิเมนต์ที่ใช้กับการเชื่อมโดยมีสอง ***attributes*** หรือมากกว่าที่ต้องจัดกลุ่ม
มิฉะนั้น	คำที่ปรากฏที่ท้ายกฎตารางเพื่อแสดงคำสั่งมิฉะนั้น
เป็น	คำที่ใช้ในรายการบันทึกคำอธิบายระหว่างคำย่อและ ***attribute text*** เต็ม

ฟังก์ชันเชิงตรรกศาสตร์ (English)

ซินแท็กซ์	คำอธิบาย
ไม่จริงว่า *<attr>*	เครื่องหมายที่ใช้ส่งคืนจริงหาก ***attribute*** มีค่าซึ่งเป็นเท็จ
<var> **แน่นอนว่า** **แน่นอนว่า** *<attr>*	เครื่องหมายที่ใช้ส่งคืนจริงหาก ***attribute*** มีค่าซึ่งไม่ใช่ ***uncertain***
<var> **ไม่แน่นอนว่า** **ไม่แน่นอนว่า** *<attr>* **ไม่แน่นอนว่า** *<attr>* **ไม่แน่นอนว่า** *<attr>* **ไม่แน่นอน**	เครื่องหมายที่ใช้ส่งคืนจริงหากค่า ***attribute*** เป็น ***uncertain***

ซินแท็กซ์	คำอธิบาย
<var> **รู้ว่า** **รู้ว่า** *<attr>*	เครื่องหมายที่ใช้ส่งคืนจริงหาก ***attribute*** มีค่าใดๆ
<var> **ไม่รู้ว่า** **ไม่รู้ว่า** *<attr>* **ไม่รู้จัก**	เครื่องหมายที่ใช้ส่งคืนจริงหาก ***attribute*** ไม่มีค่า

ค่าคงที่เชิงตรรกศาสตร์ (English)

ซินแท็กซ์	คำอธิบาย
จริง	ค่าคงที่เป็นจริงที่ใช้สำหรับกฎของตาราง
เท็จ	ค่าคงที่เป็นเท็จที่ใช้สำหรับกฎของตาราง
ไม่แน่นอน	ค่าคงที่ ***uncertain*** ที่ใช้สำหรับกฎของตาราง

เครื่องหมายการเปรียบเทียบ (English)

ซินแท็กซ์	คำอธิบาย
*<lhs>***<***<rhs>*	น้อยกว่า หมายเหตุ: ไม่มีรูปแบบภาษากลางเมื่อใช้เครื่องหมายนี้กับค่าตัวเลขและสกุลเงิน
<lhs> **>** *<rhs>*	มากกว่า หมายเหตุ: ไม่มีรูปแบบภาษากลางเมื่อใช้เครื่องหมายนี้กับค่าตัวเลขและสกุลเงิน
*<lhs>***<=***<rhs>*	น้อยกว่าหรือเท่ากับ
<lhs> **>=** *<rhs>*	มากกว่าหรือเท่ากับ
*<lhs>***=***<rhs>*	เท่ากับ
<lhs> **<>** *<rhs>*	ไม่เท่ากับ

ฟังก์ชันตัวเลข (English)

ซินแท็กซ์	คำอธิบาย
ตัวเลข(*<numText>*)	แปลงสตริงที่ระบุเป็นค่าตัวเลข
<x> **+** *<y>*	การบวกทางคณิตศาสตร์
<x> **-** *<y>*	การลบทางคณิตศาสตร์
<lhs> ***** *<rhs>*	การคูณทางคณิตศาสตร์
<lhs> **/** *<rhs>*	การหารทางคณิตศาสตร์

ซินแท็กซ์	คำอธิบาย
<lhs> **** *<rhs>*	การหารจำนวนเต็ม
<lhs> **modulo** *<rhs>*	เศษหลังจากการหารจำนวนเต็ม
สูงสุด(*<x>*, *<y>*) **สูงสุด**(*<date/time/datetime1>*, *<date/time/datetime2>*)	ส่งคืนค่าที่มากกว่าของสองค่า
ต่ำสุด(*<x>*, *<y>*) **ต่ำสุด**(*<date/time/datetime1>*, *<date/time/datetime2>*)	ส่งคืนค่าที่น้อยกว่าของสองค่า
Xy(*<x>*, *<y>*)	x ยกกำลัง y
เอ็กซ์โพเนนเชียล(*<x>*)	ค่าคงที่ e ยกกำลัง x
ค่าสัมบูรณ์ (*<x>*) **|***<val>***|**	ค่าสัมบูรณ์ของ x
ลอการิธึมธรรมชาติ (*<x>*)	ลอการิธึมธรรมชาติของ x
ลอการิธึม(*<x>*)	ลอการิธึมฐาน 10 ของ x
รากที่สอง(*<x>*)	รากที่สองของ x
ปัดเศษ(*<x>*, *<n>*)	ปัดเศษ x เป็นจุดทศนิยม n หลัก
ตัด(*<x>*, *<n>*)	ตัด x เป็นจุดทศนิยม n หลัก
ไซน์ (*<x>*)	ไซน์ของ x
โคไซน์ (*<x>*)	โคไซน์ของ x
แทนเจนต์ (*<x>*)	แทนเจนต์ของ x
อาร์กไซน์ (*<x>*)	อาร์กไซน์ของ x
อาร์กโคไซน์ (*<x>*)	อาร์กโคไซน์ของ x
อาร์กแทนเจนต์ (*<x>*)	อาร์กแทนเจนต์ของ x

ฟังก์ชันเวลา(English)

ซินแท็กซ์	คำอธิบาย
วันที่ปัจจุบัน()	ส่งคืน ***date*** ปัจจุบันเมื่อเริ่มต้นเซสชัน
วันที่ (*<text>*)	แปลงสตริงที่ระบุเป็นค่า ***date***
สร้างวันที่ (*<year>*, *<month>*, *<day>*)	ส่งคืน ***date*** ที่สร้างจากปี เดือนและวันที่ระบุ
ดึงข้อมูลวัน(*<date/datetime>*)	ส่งคืนส่วนวันของ ***date/datetime attribute***

ซินแท็กซ์	คำอธิบาย
ดึงข้อมูลเดือน (*<date/datetime>*)	ส่งคืนส่วนเดือนของ ***date/datetime attribute***
ดึงข้อมูลปี (*<date/datetime>*)	ส่งคืนส่วนปีของ ***date/datetime attribute***
วันถัดไป(*<date/datetime>*, *<day>*)	ส่งคืน ***date*** ของวันทำงานถัดไปที่เป็นวันเดียวกับหรือก่อนหน้า/หลังจาก ***date*** (ขึ้นอยู่กับซินแท็กซ์ที่ใช้)
วันที่ถัดไป(*<date>*, *<day>*, *<month>*)	ส่งคืนอินสแตนซ์ถัดไปของวันที่และเดือนที่ระบุหลังจาก ***date***
เพิ่มวัน(*<date/datetime>*, *<num_days>*)	เพิ่ม/ลบจำนวนวันใน ***date*** เมื่อใช้รูปแบบซินแท็กซ์แบบสั้น จำนวนต้องเป็นจำนวนเต็มบวกเพื่อเพิ่มจำนวนวันในอินพุต ***date*** หรือจำนวนลบเพื่อลบจำนวนวันออกจากอินพุต ***date***
เพิ่มสัปดาห์ (*<date/datetime>*, *<num_weeks>*)	เพิ่มจำนวนสัปดาห์ใน ***date*** เมื่อใช้รูปแบบซินแท็กซ์แบบสั้นเหล่านี้ จำนวนต้องเป็นจำนวนเต็มบวกเพื่อเพิ่มจำนวนสัปดาห์ในอินพุต ***date***
เพิ่มเดือน(*<date/datetime>*, *<num_months>*)	เพิ่มจำนวนเดือนใน ***date*** เมื่อใช้รูปแบบซินแท็กซ์แบบสั้นเหล่านี้ จำนวนต้องเป็นจำนวนเต็มบวกเพื่อเพิ่มจำนวนเดือนในอินพุต ***date***
เพิ่มปี (*<date/datetime>*, *<num_years>*)	เพิ่มจำนวนปีใน ***date*** เมื่อใช้รูปแบบซินแท็กซ์แบบสั้นเหล่านี้ จำนวนต้องเป็นจำนวนเต็มบวกเพื่อเพิ่มจำนวนปีในอินพุต ***date***
นับวันทำงาน (*<date1>*, *<date2>*)	นับจำนวนวันทำงานระหว่าง ***date***1 และ ***date***2 นั่นคือจำนวนวันที่อยู่ระหว่างวันจันทร์และวันศุกร์ หมายเหตุ: รวม ***date*** แรกและไม่รวม ***date*** หลัง
เริ่มต้นปี (*<date/datetime>*)	ส่งคืน ***date*** แรกสุดในปีของ ***date***
สิ้นสุดปี (*<date/datetime>*)	ส่งคืน ***date*** สุดท้ายในปีของ ***date***
ส่วนต่างวัน(*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของวันระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างวันรวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของวัน (รวม) ระหว่าง ***date/datetime1*** และ ***date/datetime2*** การคำนวณนี้รวมจุดสิ้นสุดทั้งสอง หากวันที่เป็นวันเดียวกัน ผลลัพธ์จะเป็น 1 ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างวันไม่รวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของวัน (ไม่รวม) ระหว่าง ***date/datetime1*** และ ***date/datetime2*** การคำนวณนี้ไม่รวมจุดสิ้นสุดทั้งสอง หากวันที่เป็นวันเดียวกัน ผลลัพธ์จะเป็น 0 ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างสัปดาห์	ส่งคืนจำนวนเต็มของสัปดาห์ที่ผ่านไประหว่าง ***date/datetime1*** และ ***date/d-***

ซินแท็กซ์	คำอธิบาย
(*<date/datetime1>*, *<date/datetime2>*)	***atetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างสัปดาห์รวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของสัปดาห์ที่ผ่านไปรวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างสัปดาห์ไม่รวม(*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของสัปดาห์ที่ผ่านไปไม่รวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างเดือน(*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของเดือนที่ผ่านไประหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างเดือนรวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของเดือนที่ผ่านไปรวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างเดือนไม่รวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนเต็มของเดือนที่ผ่านไปไม่รวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างปี (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนปีระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างปีรวม(*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนปีรวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์
ส่วนต่างปีไม่รวม (*<date/datetime1>*, *<date/datetime2>*)	ส่งคืนจำนวนปีไม่รวมระหว่าง ***date/datetime1*** และ ***date/datetime2*** ลำดับของสองวันที่จะไม่มีผลกับผลลัพธ์

ฟังก์ชันเวลาของวัน(English)

ซินแท็กซ์	คำอธิบาย
เวลา(*<text>*)	แปลงสตริงที่ระบุเป็นค่าเวลาของวัน
ดึงข้อมูลวินาที (*<time/datetime>*)	ส่งคืนส่วนวินาทีของ ***timeofday/datetime attribute***
ดึงข้อมูลนาที (*<time/datetime>*)	ส่งคืนส่วนนาทีของ ***timeofday/datetime attribute***
ดึงข้อมูลชั่วโมง(*<time/datetime>*)	ส่งคืนส่วนชั่วโมงของ ***timeofday/datetime attribute***

ฟังก์ชันวันที่และเวลา(English)

ซินแท็กซ์	**คำอธิบาย**
วันที่เวลาปัจจุบัน()	ส่งคืน ***date*** และเวลาปัจจุบันเมื่อเริ่มต้นเซสชัน
วันที่เวลา(*<text>*)	แปลงสตริงที่ระบุเป็นค่า ***datetime***
ต่อวันที่เวลา(*<date>*, *<time>*)	ตั้งค่า ***date*** เวลาโดยการรวม ***date*** และเวลาของวันเข้าด้วยกัน
ส่วนต่างวินาที (*<datetime1>*, *<datetime2>*) **ส่วนต่างวินาที** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนวินาทีระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างวินาทีรวม(*<datetime1>*, *<datetime2>*) **ส่วนต่างวินาทีรวม** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนวินาทีรวมระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างวินาทีไม่รวม (*<datetime1>*, *<datetime2>*) **ส่วนต่างวินาทีไม่รวม** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนวินาทีไม่รวมระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างนาที (*<datetime1>*, *<datetime2>*) **ส่วนต่างนาที** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนนาทีระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างนาทีรวม(*<datetime1>*, *<datetime2>*) **ส่วนต่างนาทีรวม** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนนาทีรวมระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างนาทีไม่รวม(*<datetime1>*, *<datetime2>*) **ส่วนต่างนาทีไม่รวม** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนนาทีไม่รวมระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างชั่วโมง(*<datetime1>*, *<datetime2>*) **ส่วนต่างชั่วโมง** (*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนชั่วโมงระหว่าง ***datetime1*** และ ***datetime2***

ซินแท็กซ์	คำอธิบาย
ส่วนต่างชั่วโมงรวม(*<datetime1>*, *<datetime2>*) **ส่วนต่างชั่วโมงรวม**(*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนชั่วโมงรวมระหว่าง ***datetime1*** และ ***datetime2***
ส่วนต่างชั่วโมงไม่รวม(*<datetime1>*, *<datetime2>*) **ส่วนต่างชั่วโมงไม่รวม**(*<timeOfDay1>*, *<timeOfDay2>*)	ส่งคืนจำนวนชั่วโมงไม่รวมระหว่าง ***datetime1*** และ ***datetime2***
ดึงข้อมูลวันที่(*<datetime>*)	ดึงข้อมูล ***date*** จาก ***datetime attribute***
ดึงข้อมูลเวลา(*<datetime>*)	ดึงข้อมูลเวลาของวันจาก ***datetime attribute*** สามารถใช้เพื่อตั้งค่าของ ***timeofday attribute*** เป็นเวลาที่รันกฎได้ โดยการดึงข้อมูลเวลาจาก ***date*** และเวลาปัจจุบัน
เพิ่มชั่วโมง(*<datetime>*, *<num_hours>*) **เพิ่มชั่วโมง**(*<timeOfDay>*, *<num_hours>*)	เพิ่มจำนวนชั่วโมงใน ***date*** เวลา
เพิ่มนาที(*<datetime>*, *<num_minutes>*) **เพิ่มนาที**(*<timeOfDay>*, *<num_minutes>*)	เพิ่มจำนวนนาทีใน ***date*** เวลา
เพิ่มวินาที(*<datetime>*, *<num_seconds>*) **เพิ่มวินาที**(*<timeOfDay>*, *<num_seconds>*)	เพิ่มจำนวนวินาทีใน ***date*** เวลา

ฟังก์ชันข้อความ (English)

ซินแท็กซ์	คำอธิบาย
<text1> **&** *<text2>*	รวม ***text1*** กับ ***text2*** และอื่นๆ เพื่อสร้างค่า ***text*** ค่าเดียว หมายเหตุ: คุณสามารถใช้ตัวแปรประเภทใดก็ได้ ค่าถูกจัดรูปแบบโดยตัวจัดรูปแบบที่ติดตั้งในเซสชันกฎ
	รวม ***text1*** กับ ***text2*** และอื่นๆ เพื่อสร้างค่า ***text*** ค่าเดียว หมายเหตุ: คุณสามารถใช้ตัวแปรประเภทใดก็ได้ ค่าถูกจัดรูปแบบโดยตัวจัดรูปแบบที่ติดตั้งในเซสชันกฎ

ซินแท็กซ์	คำอธิบาย
มี (*<text>*, *<substring>*)	ส่งคืนค่าบูลีนที่ระบุว่าค่า ***text*** ที่ระบุมีสตริงย่อย ***text*** ที่ระบุหรือไม่ การเปรียบเทียบ ***text*** ต้องเป็นตัวพิมพ์ที่ตรงกัน
ลงท้ายด้วย(*<text>*, *<substring>*)	ส่งคืนค่าบูลีนที่ระบุว่าค่า ***text*** ที่ระบุลงท้ายด้วยสตริงย่อย ***text*** ที่ระบุหรือไม่ การเปรียบเทียบ ***text*** ต้องเป็นตัวพิมพ์ที่ตรงกัน
เป็นตัวเลข(*<text>*)	ส่งคืนค่าบูลีนที่ระบุว่าค่า ***text*** ที่ระบุเป็นค่าตัวเลขที่ถูกต้องหรือไม่
ความยาว(*<text>*)	ส่งคืนความยาวตัวอักษรของค่า ***text*** ที่ระบุ
เริ่มต้นด้วย (*<text>*, *<substring>*)	ส่งคืนค่าบูลีนที่ระบุว่าค่า ***text*** ที่ระบุเริ่มต้นด้วยสตริงย่อย ***text*** ที่ระบุหรือไม่ การเปรียบเทียบ ***text*** ต้องเป็นตัวพิมพ์ที่ตรงกัน
สตริงย่อย(*<text>*, *<offset>*, *<length>*)	ส่งคืนสตริงย่อยของ ***text*** ที่เริ่มต้นตามออฟเซ็ตที่ระบุ ซึ่งเป็นความยาวที่ระบุ เป็นตัวอักษร โดยจะส่งคืนตัวอักษรน้อยกว่าหากถึงจุดสิ้นสุดของสตริง
ข้อความ(*<number>*) **ข้อความ**(*<date>*) **ข้อความ** (*<datetime>*) **ข้อความ** (*<timeOfDay>*)	แปลงตัวเลขหรือ ***date attribute*** ที่ระบุเป็นค่า ***text***

ฟังก์ชันเอนทิตีและความสัมพันธ์ (English)

ซินแท็กซ์	คำอธิบาย
สำหรับ(*<relationship>*, *<Exp>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหนึ่ง""หลายต่อหนึ่ง" หรือ "หลายต่อหลาย" เมื่อมีเงื่อนไขเพียงหนึ่งข้อ
สำหรับขอบเขต(*<relationship>*, *<alias>*) **สำหรับขอบเขต**(*<relationship>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหนึ่ง""หลายต่อหนึ่ง" หรือ "หลายต่อหลาย" เมื่อมีเงื่อนไขอย่างน้อยหนึ่งข้อ
สำหรับทั้งหมด(*<relationship>*, *<Exp>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหลาย" หรือ "หลายต่อหลาย" เมื่อคุณต้องการกำหนดว่าสมาชิกทั้งหมดของกลุ่ม ***entity*** เป้าหมายต้องตรงตามกฎหรือไม่ ใช้รูปแบบนี้เมื่อมีเงื่อนไขในกฎเพียงหนึ่งข้อ
สำหรับขอบเขตทั้งหมด (*<relationship>*) **สำหรับขอบเขตทั้งหมด** (*<relationship>*, *<alias>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหลาย" หรือ "หลายต่อหลาย" เมื่อคุณต้องการกำหนดว่าสมาชิกทั้งหมดของกลุ่ม ***entity*** เป้าหมายต้องตรงตามกฎหรือไม่ ใช้รูปแบบนี้เมื่อมีเงื่อนไขในกฎอย่างน้อยหนึ่งข้อ
มีอยู่ (*<relationship>*, *<Exp>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหลาย" หรือ "หลายต่อหลาย" เมื่อคุณต้องการกำหนดว่าสมาชิกคนใดคนหนึ่งข

ซินแท็กซ์	คำอธิบาย
	องกลุ่ม ***entity*** เป้าหมายต้องตรงตามกฎหรือไม่ ใช้รูปแบบนี้เมื่อมีเงื่อนไขในกฎเพียงหนึ่งข้อ
ขอบเขตที่มีอยู่ (*<relationship>*) **ขอบเขตที่มีอยู่** (*<relationship>*, *<alias>*)	ใช้เพื่ออ้างอิงจาก ***entity*** ไปยังอีก ***entity*** ในความสัมพันธ์ ***relationship*** ประเภท "หนึ่งต่อหลาย" หรือ "หลายต่อหลาย" เมื่อคุณต้องการกำหนดว่าสมาชิกคนใดคนหนึ่งของกลุ่ม ***entity*** เป้าหมายต้องตรงตามกฎหรือไม่ ใช้รูปแบบนี้เมื่อมีเงื่อนไขในกฎอย่างน้อยหนึ่งข้อ
เป็นสมาชิกของ (*<target>*, *<relationship>*) **เป็นสมาชิกของ** (*<target>*, *<alias>*, *<relationship>*) <ent-target> **เป็นส่วนหนึ่งของ** *<relationship>* <ent-target> **(**<alias>**) เป็นส่วนหนึ่งของ** *<relationship>*	ใช้เป็นข้อสรุปเพื่อระบุว่าอินสแตนซ์ ***entity*** เป็นสมาชิกของ ***relationship*** ใช้เป็นเงื่อนไขเพื่อทดสอบว่าอินสแตนซ์ ***entity*** เป็นเป้าหมายของ ***relationship*** ซึ่งอินสแตนซ์ ***entity*** ที่สองเป็นที่มา
ไม่เป็นสมาชิกของ(*<target>*, *<relationship>*)	ใช้เป็นเงื่อนไขเพื่อทดสอบว่าอินสแตนซ์ ***entity*** ไม่ใช่เป้าหมายของ ***relationship*** ซึ่งอินสแตนซ์ ***entity*** ที่สองเป็นที่มา
อินสแตนซ์นับ(*<relationship>*)	นับจำนวนของอินสแตนซ์ที่มีอยู่สำหรับ ***entity***
อินสแตนซ์นับถ้า(*<relationship>*, *<Exp>*)	นับจำนวนอินสแตนซ์ที่เป็นของ ***entity*** ซึ่ง ***entity-level attribute*** นั้นมีค่าเฉพาะ
อินสแตนซ์สูงสุด(*<relationship>*, *<number-attr>*) **อินสแตนซ์สูงสุด**(*<relationship>*, *<date-attr>*) **อินสแตนซ์สูงสุด**(*<relationship>*, *<datetime-attr>*) **อินสแตนซ์สูงสุด**(*<relationship>*, *<time-attr>*)	หาค่าสูงสุด/ใหม่ที่สุดของตัวแปร ***entity-level*** สำหรับอินสแตนซ์ทั้งหมดของ ***entity***
อินสแตนซ์สูงสุดถ้า (*<relationship>*, *<number-attr>*, *<condition>*) **อินสแตนซ์สูงสุดถ้า** (*<relationship>*, *<date-attr>*, *<condition>*) **อินสแตนซ์สูงสุดถ้า** (*<relationship>*, *<datetime-attr>*, *<condition>*) **อินสแตนซ์สูงสุดถ้า**	หาค่าสูงสุด/ใหม่ที่สุดของตัวแปร ***entity-level*** สำหรับอินสแตนซ์ทั้งหมดของ ***entity*** ซึ่ง ***entity-level attribute*** นั้นมีค่าเฉพาะ

ซินแท็กซ์	คำอธิบาย
(*<relationship>*, *<time-attr>*, *<condition>*)	
อินสแตนซ์ต่ำสุด(*<relationship>*, *<number-attr>*) **อินสแตนซ์ต่ำสุด**(*<relationship>*, *<date-attr>*) **อินสแตนซ์ต่ำสุด**(*<relationship>*, *<datetime-attr>*) **อินสแตนซ์ต่ำสุด**(*<relationship>*, *<time-attr>*)	หาค่าต่ำสุด/เก่าที่สุดของตัวแปร ***entity-level*** สำหรับอินสแตนซ์ทั้งหมดของ ***entity***
อินสแตนซ์ต่ำสุดถ้า(*<relationship>*, *<number-attr>*, *<condition>*) **อินสแตนซ์ต่ำสุดถ้า**(*<relationship>*, *<date-attr>*, *<condition>*) **อินสแตนซ์ต่ำสุดถ้า**(*<relationship>*, *<datetime-attr>*, *<condition>*) **อินสแตนซ์ต่ำสุดถ้า**(*<relationship>*, *<time-attr>*, *<condition>*)	หาค่าต่ำสุด/เก่าที่สุดของตัวแปร ***entity-level*** สำหรับอินสแตนซ์ทั้งหมดของ ***entity*** ซึ่ง ***entity-level attribute*** นั้นมีค่าเฉพาะ
อินสแตนซ์ผลรวม(*<relationship>*, *<number-attr>*)	หาผลรวมของอินสแตนซ์ทั้งหมดของตัวแปร ***entity-level***
อินสแตนซ์ผลรวมถ้า(*<relationship>*, *<number-attr>*, *<condition>*)	หาผลรวมของอินสแตนซ์ทั้งหมดของตัวแปร ***entity-level*** ที่เป็นจริงของ ***entity*** ที่ ***attribute*** บูลีน ***entity-level*** เป็นจริง
อินสแตนซ์ค่าถ้า(*<relationship>*, *<number-attr>*, *<condition>*) **อินสแตนซ์ค่าถ้า**(*<relationship>*, *<text-attr>*, *<condition>*) **อินสแตนซ์ค่าถ้า**(*<relationship>*, *<date-attr>*, *<condition>*) **อินสแตนซ์ค่าถ้า**(*<rela-*	หาค่าจากอินสแตนซ์ ***entity*** ที่ไม่ซ้ำกันซึ่งระบุจากอินสแตนซ์ ***entity*** เป้าหมายของ ***relationship*** ตามเงื่อนไข • หากเงื่อนไขระบุอินสแตนซ์ ***entity*** เป้าหมายเดียว ค่าจะเป็นค่าที่คำนวณกับอินสแตนซ์ ***entity*** • หากอินสแตนซ์เป้าหมายมากกว่าหนึ่งอินสแตนซ์ตรงตามเงื่อนไข จะส่งคืน ***uncertain*** • หากไม่มีอินสแตนซ์เป้าหมายที่ตรงตามเงื่อนไขและรู้จัก ***relationship*** ค่าจะเป็น ***uncertain***

ซินแท็กซ์	คำอธิบาย
tionship>, *<datetime-attr>*, *<condition>*) **อินสแตนซ์ค่าถ้า**(*<relationship>*, *<time-attr>*, *<condition>*)	
อินสแตนซ์เท่า (*<instance1>*, *<instance2>*)	พิจารณาว่าสองอินสแตนซ์ของ ***entity*** เป็นอินสแตนซ์เดียวกันหรือไม่
อินสแตนซ์ไม่เท่า (*<instance1>*, *<instance2>*)	พิจารณาว่าสองอินสแตนซ์ของ ***entity*** ไม่ใช่อินสแตนซ์เดียวกันหรือไม่
สรุปอินสแตนซ์ (*<relationship>*, *<identity>*)	ใช้เป็นข้อสรุปเพื่อระบุว่าอินสแตนซ์ ***entity*** มีอยู่และเป็นสมาชิกของ ***relationship***

ฟังก์ชันเหตุผลเกี่ยวกับเวลา(English)

ซินแท็กซ์	คำอธิบาย
ช่วงนับไม่ซ้ำกัน(*<start-date>*, *<end-date>*, *<variable>*) **ช่วงนับไม่ซ้ำกัน**(*<start-date>*, *<end-date>*, *<condition>*)	นับจำนวนค่าไม่ซ้ำกันที่รู้จักสำหรับตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม)
ช่วงนับไม่ซ้ำกันถ้า(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	นับจำนวนค่าไม่ซ้ำกันที่รู้จักสำหรับตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) รวมเวลาเฉพาะเมื่อฟิลเตอร์บูลีนเป็นจริง
ช่วงผลรวมรายวัน (*<start-date>*, *<end-date>*, *<number-attr>*)	คำนวณผลรวมของตัวแปรสกุลเงินหรือจำนวนในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) ***attribute*** ถูกกำหนดเป็นปริมาณรายวัน
ช่วงผลรวมรายวันถ้า(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	คำนวณผลรวมของค่ารายวันสำหรับตัวแปรสกุลเงินหรือจำนวนในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) รวมเวลาเฉพาะเมื่อเงื่อนไขเป็นจริง

ซินแท็กซ์	คำอธิบาย
ช่วงสูงสุด (*<start-date>*, *<end-date>*, *<number-attr>*) **ช่วงสูงสุด** (*<start-date>*, *<end-date>*, *<date-attr>*) **ช่วงสูงสุด** (*<start-date>*, *<end-date>*, *<datetime-attr>*) **ช่วงสูงสุด** (*<start-date>*, *<end-date>*, *<time-attr>*)	เลือกค่าสูงสุดของตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม)
ช่วงสูงสุดถ้า (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **ช่วงสูงสุดถ้า** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **ช่วงสูงสุดถ้า** (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **ช่วงสูงสุดถ้า** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	เลือกค่าสูงสุดของตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) รวมเวลาเมื่อเงื่อนไขเป็นจริงเท่านั้น
ช่วงต่ำสุด (*<start-date>*, *<end-date>*, *<number-attr>*) **ช่วงต่ำสุด** (*<start-date>*, *<end-date>*, *<date-attr>*) **ช่วงต่ำสุด** (*<start-date>*, *<end-date>*, *<datetime-attr>*) **ช่วงต่ำสุด** (*<start-date>*, *<end-*	เลือกค่าต่ำสุดของตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม)

ซินแท็กซ์	คำอธิบาย
date>, *<time-attr>*)	
ช่วงต่ำสุดถ้า (*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **ช่วงต่ำสุดถ้า** (*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **ช่วงต่ำสุดถ้า** (*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **ช่วงต่ำสุดถ้า** (*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	เลือกค่าต่ำสุดของตัวแปรในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) รวมเวลาเมื่อเงื่อนไขเป็นจริงเท่านั้น
ช่วงค่าเฉลี่ยถ่วงน้ำหนัก(*<start-date>*, *<end-date>*, *<number-attribute>*)	คำนวณค่าเฉลี่ยของตัวแปรสกุลเงินหรือจำนวนในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) ถ่วงน้ำหนักตามช่วงเวลาที่ใช้แต่ละค่า
ช่วงค่าเฉลี่ยถ่วงน้ำหนักถ้า (*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	คำนวณค่าเฉลี่ยของตัวแปรสกุลเงินหรือจำนวนในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) รวมเวลาเฉพาะเมื่อเงื่อนไขบูลีนเป็นจริง (ถ่วงน้ำหนักตามช่วงเวลาที่ใช้แต่ละค่าและฟิลเตอร์เป็นจริง)
ช่วงเสมอ(*<start-date>*, *<end-date>*, *<condition>*)	ส่งคืนจริงหากเงื่อนไขบูลีนเป็นจริงตลอดเวลาในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) เท่านั้น
ช่วงวันอย่างน้อย(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	ส่งคืนจริงหากเงื่อนไขบูลีนเป็นจริงอย่างน้อยสำหรับจำนวนวันที่ระบุ (ไม่จำเป็นต้องติดต่อกัน) ในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) เท่านั้น
ช่วงวันที่ติดต่อกัน(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	ส่งคืนจริงถ้าเงื่อนไขบูลีนเป็นจริงอย่างน้อยสำหรับจำนวนวันติดต่อกันที่ระบุในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) เท่านั้น

ซินแท็กซ์	คำอธิบาย
ช่วงบางครั้ง (*<start-date>*, *<end-date>*, *<condition>*)	ส่งคืนจริงถ้าเงื่อนไขบูลีนเป็นจริงในช่วงจาก ***date*** เริ่มต้น (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) เท่านั้น
ค่าอยู่ที่ (*<date>*, *<value>*)	ส่งคืนค่าของ ***attribute*** ที่ระบุใน ***date*** ที่ระบุ
เมื่อสุดท้าย (*<date>*, *<condition>*)	ส่งคืน ***date*** ที่เงื่อนไขบูลีนเป็นจริงครั้งสุดท้าย ค้นหาย้อนหลังจาก (และรวม) ***date*** ที่ระบุ
เมื่อถัดไป (*<date>*, *<condition>*)	ส่งคืน ***date*** ที่เงื่อนไขบูลีนเป็นจริงครั้งถัดไป ค้นหาไปข้างหน้าจาก (และรวม) ***date*** ที่ระบุ
ท้ายสุด()	ส่งคืนค่า ***date*** เทียบเท่ากับ ***date*** ที่เป็นไปได้ล่าสุด - โดย ***date*** ต้องอยู่หลัง ***date*** อื่นๆ ที่ ***date attribute*** สามารถใช้ได้ หรือหาค่าจากนิพจน์ได้
แรกสุด()	ส่งคืนค่า ***date*** เทียบเท่ากับ ***date*** ที่เป็นไปได้แรกสุด - โดย ***date*** ต้องอยู่ก่อน ***date*** อื่นๆ ที่ ***date attribute*** สามารถใช้ได้ หรือหาค่าจากนิพจน์ได้
เวลาตั้งแต่วัน (*<date>*, *<end-date>*)	ส่งคืนตัวแปรจำนวนที่เปลี่ยนแปลงทุกวันและเป็นจำนวนวันเต็มนับตั้งแต่ ***date***
เวลาตั้งแต่สัปดาห์ (*<date>*, *<end-date>*)	ส่งคืนตัวแปรจำนวนที่เปลี่ยนแปลงทุกสัปดาห์และเป็นจำนวนสัปดาห์เต็มนับตั้งแต่ ***date***
เวลาตั้งแต่เดือน (*<date>*, *<end-date>*)	ส่งคืนตัวแปรจำนวนที่เปลี่ยนแปลงทุกเดือนและเป็นจำนวนเดือนเต็มนับตั้งแต่ ***date*** หมายเหตุ: เมื่อ ***date*** ที่ป้อนอยู่หลังวันที่ 28 ของเดือนและเดือนต่อมามีจำนวนวันน้อยกว่าเดือนที่ป้อน จุดเปลี่ยนแปลงของเดือนครบรอบจะถูกจัดทำขึ้นในวันสุดท้ายของเดือนนั้น ตัวอย่างเช่น หาก ***date*** ที่ป้อนเป็น 28, 29, 30 หรือ 31 มกราคม 2007 จุดเปลี่ยนแปลงครั้งแรกสุดจะเป็น 28 กุมภาพันธ์ 2007
เวลาตั้งแต่ปี (*<date>*, *<end-date>*)	ส่งคืนตัวแปรจำนวนที่เปลี่ยนแปลงทุกปีและเป็นจำนวนปีเต็มนับตั้งแต่ ***date***
เวลาวันเสมอ (*<days>*, *<condition>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลาและเป็นจริงหากเงื่อนไขบูลีนเป็นจริงสำหรับจำนวนก่อนหน้าที่ระบุทั้งหมดไม่รวมวันปัจจุบันเท่านั้น
เวลาวันติดต่อกัน(*<minDays>*, *<days>*, *<condition>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลาและเป็นจริงหากเงื่อนไขบูลีนเป็นจริงอย่างน้อยสำหรับจำนวนวันที่ติดต่อกันในระยะเวลาใดๆ ภายในจำนวนวันก่อนหน้าที่ระบุ ไม่รวมวันปัจจุบันเท่านั้น
เวลาวันบางครั้ง	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลาและเป็นจริงหากเงื่อนไขบูลีนเป็นจริงภา

ซินแท็กซ์	คำอธิบาย
(*<days>*, *<condition>*)	ยในจำนวนวันก่อนหน้าที่ระบุ ไม่รวมวันปัจจุบันเท่านั้น
เวลาหลัง(*<date>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลา และเป็นจริงหลังจาก ***date*** และเป็นเท็จในวันที่นั้นและก่อนหน้านั้น
เวลาก่อน(*<date>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลา และเป็นจริงก่อนหน้า ***date*** และเป็นเท็จในวันที่นั้นและหลังจากนั้น
เวลาเมื่อ(*<date>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลา และเป็นจริงใน ***date*** และเป็นเท็จก่อนหน้าหรือหลังจากนั้น
เวลาเมื่อหรือหลัง(*<date>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลา และเป็นจริงในวันเดียวกับหรือหลังจาก ***date*** และเป็นเท็จก่อนหน้านั้น
เวลาเมื่อหรือก่อน(*<date>*)	ส่งคืน ***attribute*** บูลีนที่เปลี่ยนแปลงตามเวลา และเป็นจริงในวันเดียวกับและก่อนหน้า ***date*** และเป็นเท็จหลังจากนั้น
เวลาจากวันที่เริ่มต้น(*<relationship>*, *<date>*, *<value>*)	ส่งคืน ***attribute*** เวลาเดียว (ที่ระดับ ***entity*** ที่มา) จาก ***relationship*** และค่า ***attribute*** ในเอนทิตีซึ่งมีค่าที่ได้รับผลจาก ***date attribute*** เริ่มต้น
เวลาจากวันที่สิ้นสุด (*<relationship>*, *<date>*, *<value>*)	ส่งคืน ***attribute*** เวลาเดียว (ที่ระดับ ***entity*** ที่มา) จาก ***relationship*** และ ***attribute*** ค่าในเอนทิตีซึ่งมีค่าที่มีผลจนถึง ***date attribute*** สิ้นสุด
เวลาจากช่วง(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	ส่งคืน ***attribute*** เวลาเดียว (ที่ระดับ ***entity*** ที่มา) จาก ***relationship*** และ ***attribute*** ค่าในเอนทิตีซึ่งมีค่าที่ได้รับผลจาก ***date attribute*** เริ่มต้น (รวม) ถึง ***date attribute*** สิ้นสุด (ไม่รวม) ค่าจะเป็น ***uncertain*** หากหมดอายุก่อน ***date*** เริ่มต้นถัดไป
เวลาวันทำงาน (*<startdate>*, *<enddate>*)	ส่งคืนจริงในวันที่ซึ่งเป็นวันทำงาน และเท็จในวันที่ซึ่งเป็นวันหยุดสุดสัปดาห์ จาก ***date*** เริ่มต้นที่ระบุ (รวม) ถึง ***date*** สิ้นสุด (ไม่รวม) ส่งคืน ***uncertain*** นอกช่วง ***date***
เวลาหนึ่งครั้งต่อเดือน(*<startdate>*, *<enddate>*, *<dayofmonth>*)	ส่งคืนจริงหากวันเท่ากับพารามิเตอร์วันของเดือน และเท็จในวันของเดือนอื่นๆ จาก ***date*** เริ่มต้นที่ระบุ (รวม) ถึง ***date*** ที่สิ้นสุด (ไม่รวม) ส่งคืน ***uncertain*** นอกช่วง ***date*** เมื่อวันของเดือนเกินจำนวนวันในเดือนปัจจุบัน ค่าจะเป็นจริงในวันสุดท้ายของเดือนนั้น เพื่อให้ฟังก์ชันส่งคืนค่าที่เป็นจริงหนึ่งวันต่อเดือนพอดี

ฟังก์ชันของกิจกรรมการตรวจสอบ (English)

ซินแท็กซ์	คำอธิบาย
ข้อผิดพลาด(*<text>*)	กิจกรรมข้อผิดพลาดจะใช้เพื่อส่งข้อความถึงผู้ใช้ และทำให้ผู้ใช้ไม่สามารถตรวจสอบต่อจนกว่าเงื่อนไขที่ทริกเกอร์ข้อผิดพลาดนั้นไม่มีผลอีก

ซินแท็กซ์	คำอธิบาย
คำเตือน (*<text>*)	กิจกรรมการเตือนจะใช้เพื่อส่งข้อความถึงผู้ใช้ แต่ยังอนุญาตให้ผู้ใช้สามารถดำเนินการต่อขณะที่มีเงื่อนไขที่ทริกเกอร์คำเตือนนั้น

ฟังก์ชันที่เลิกใช้ (English)

ซินแท็กซ์	คำอธิบาย
เรียกฟังก์ชันที่กำหนดเอง(*<A>*, *<B>*)	ส่งคืนผลลัพธ์ของการเรียกภายนอกไปยังไลบรารีรหัส ต้องระบุไลบรารีรหัสให้กับ Determinations Engine เพื่อให้เรียกฟังก์ชันที่กำหนดได้

Mantıksal bağlayıcılar(English)

Sözdizimi	Tanımlama
eğer	Şu ispata sahip olan bir sonuç satırının sonunda görünmesi olası isteğe bağlı terim
ve	İki ***attributes*** arasındaki mantıksal bağlaç
veya	İki ***attributes*** arasındaki mantıksal ayraç
biri ya da öbürü **aralarından biri** **herhangi biri** **aşağıdakilerden en az biri doğru** **aşağıdakilerin herhangi biri yerine geldi**	İki veya daha fazla ***attributes*** özelliğinin gruplanması gerektiğinde ayraçlarla birlikte kullanılan gruplama bileşeni
her ikisi **tümü** **aşağıdakilerin tümü doğru** **aşağıdakilerin tümü yerine geldi**	İki veya daha çok ***attributes*** özelliğinin gruplanması gerektiğinde bağlaçlarla birlikte kullanılan gruplama bileşeni
aksi halde	aksi halde tümceciğini göstermek için tablo kuralının sonunda görünen terim
=	Kısaltılmış deyim ile tam ***attribute text*** arasındaki altyazı girişinde kullanılan terim

Mantıksal fonksiyonlar(English)

Sözdizimi	Tanımlama
<attr>(**olmaması | önergesinin yanlış olması**)	***attribute*** yanlış değerine sahipse doğru değerini döndürmek için kullanılan işleç
<var> **kesinse**	***attribute***, ***uncertain*** olmayan bir değer içerirse doğru

Sözdizimi	Tanımlama
<var>[**değeri**] **kesin** [**ise**] *<var>*[**değeri**] **belirli** [**ise**] *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirli** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirliyse** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirli ise**	değerini döndürmek için kullanılan işleç
<var> **kesin değil** [**ise**] *<var>* **kesin değilse** *<var>* **belirsizse** *<var>* **belirsiz** [**ise**] *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirsiz** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirsizse** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **belirsiz ise**	İşleç ***attribute*** değeri ***uncertain*** ise doğru değerini döndürür
<var>(**bilinirse** | **biliniyor** | **biliniyor ise** | **biliniyorsa** | **bilinir** | **bilinir ise**) *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **biliniyor** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **biliniyorsa** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **biliniyor ise**	***attribute*** özelliği değer içeriyorsa doğru değerini döndürmek için kullanılan işleç
<var>(**bilinmezse** | **bilinmez** | **bilinmez ise** | **bilinmiyorsa** | **bilinmiyor** | **bilinmiyor ise**) *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **bilinmiyor** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **bilinmiyorsa** *<attr>*(**mu** | **mi** | **mü** | **mı** | **değil mi**) **bilinmiyor ise**	İşleç ***attribute***, bir değer içermezse doğru değerini döndürür

Mantıksal sabitler(English)

Sözdizimi	Tanımlama
doğru	Tablo kuralları için sabit doğru değerini kullanılır.
yanlış	Tablo kuralları için sabit yanlış değeri kullanılır.
belirsiz	Tablo kuralları için sabit ***uncertain*** değeri kullanılır.

Karşılaştırma işleçleri(English)

Sözdizimi	Tanımlama
*<lhs>***<***<rhs>* *<lhs><rhs>***'den daha azsa** *<lhs><rhs>***'den önceyse**	Küçüktür Not: Bu işleç sayısal değerler veya para birimi değerleriyle birlikte kullanıldığında doğal dil formu yoktur.
<lhs> **>** *<rhs>* *<lhs><rhs>***'dan fazlaysa**	Büyüktür Not: Bu işleç sayısal değerler veya para birimi değerleriyle birlikte kullanıldığında doğal

Sözdizimi	Tanımlama
<lhs><rhs>'**den daha geç oluyorsa**	dil formu yoktur.
*<lhs>***<=***<rhs>* *<lhs><rhs>*'**den daha az veya eşitse**	Küçüktür veya eşittir
<lhs> **>=** *<rhs>* *<lhs><rhs>*'**dan daha büyük veya eşitse**	Büyüktür veya eşittir
*<lhs>***=***<rhs>* *<lhs><rhs>* **eşitse** *<lhs><rhs>* **eşitliyorsa**	Eşittir
<lhs><rhs> **eşit değilse** *<lhs><rhs>* **eşitlemiyorsa** *<lhs>* **<>** *<rhs>* *<lhs>* **!=** *<rhs>*	Eşit değildir

Sayısal fonksiyonlar(English)

Sözdizimi	Tanımlama
Sayı(*<numText>*)	Belirtilen dizeyi sayı değerine dönüştürür
<x> **+** *<y>*	Toplama işlemi
<x> **-** *<y>*	Çıkarma işlemi
<x> * *<y>*	Çarpım işlemi
<x> **/** *<y>*	Bölme işlemi
<x> **** *<y>*	Tam sayı bölme
<x> **modulo** *<y>*	Tam sayı bölümünde kalan
Maksimum(*<x>*, *<y>*) **Maksimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) *<val1>*'**nin ve** *<val2>*'**nün büyüğü** *<val1>*'**nin ve** *<val2>*'**nün en sonu**	İki değerden büyük olanı döndürür
Minimum(*<x>*, *<y>*) **Minimum**(*<date/time/datetime1>*, *<date/time/datetime2>*) *<val1>*'**nin ve** *<val2>*'**nün küçüğü** *<val1>*'**nin ve** *<val2>*'**nün en erkeni**	İki değerden küçük olanı döndürür
Xy(*<x>*, *<y>*)	x üzeri y

Sözdizimi	Tanımlama
<val> **yükseltilmiş** *<power>* **iktidara**	
Eks(*<x>*) **e to the power of** *<log-val>*	Sabit e üzeri x
Mut.(*<x>*) *<val>* **mutlak değeri** **|***<val>***|**	x değerinin mutlak değeri
Ln(*<x>*) *<log-val>* **doğal logaritma**	x değerinin doğal logaritması
Log(*<x>*) *<log-val>* **logaritmik tabanı 10**	Logaritma 10 tabanına göre x
Karekök(*<x>*) *<val>* **kare kökü**	x değerinin karekökü
Yuvarla(*<x>*, *<n>*) *<val><num_places>* **ondalık yuvarlanır**	x değerini n ondalık basamağa yuvarlar
Kes(*<x>*, *<n>*) *<val>* **kesilmiş için** *<num_places>* **ondalık**	x değeri n sayıda ondalık basamağa kesilir
Sin(*<x>*)	x değerinin sinüsü
Cos(*<x>*)	x değerinin kosinüsü
Tan(*<x>*)	x değerinin tanjantı
Asin(*<x>*)	x değerinin ark sinüsü
Acos(*<x>*)	x değerinin ark kosinüsü
Atan(*<x>*)	x değerinin ark tanjantı

Tarih fonksiyonları(English)

Sözdizimi	Tanımlama
GeçerliTarih() **günün tarihi**	Oturumun başlangıcında geçerli ***date*** öğesini döndürür.
Tarih(*<text>*)	Belirtilen dizeyi ***date*** değerine dönüştürür
TarihYap(*<year>*, *<month>*, *<day>*)	Belirtilen yıl, ay ve günden oluşturulan bir ***date*** özelliği değeri döndürür.
GünÇıkar(*<date/datetime>*)	***date/datetime attribute*** özelliğinin gün bileşenini döndürür.

Sözdizimi	Tanımlama
AyÇıkar (*<date/datetime>*)	***date/datetime attribute*** özelliğinin ay bileşenini döndürür.
YılÇıkar (*<date/datetime>*)	***date/datetime attribute*** özelliğinin yıl bileşenini döndürür.
SonrakiGün(*<date/datetime>*, *<day>*) *<from-date>***'e denk gelen yada sonraki Pazartesi** *<from-date>***'e denk gelen yada önceki Pazartesi** *<from-date>***'e denk gelen yada sonraki Salı** *<from-date>***'e denk gelen yada önceki Salı** *<from-date>***'e denk gelen yada sonraki Çarşamba** *<from-date>***'e denk gelen yada önceki Çarşamba** *<from-date>***'e denk gelen yada sonraki Perşembe** *<from-date>***'e denk gelen yada önceki Perşembe** *<from-date>***'e denk gelen yada sonraki Cumartesi** *<from-date>***'e denk gelen yada önceki Cumartesi** *<from-date>***'e denk gelen yada sonraki Cuma** *<from-date>***'e denk gelen yada önceki Cuma** *<from-date>***'e denk gelen yada sonraki Pazar**	Sonraki hafta içi günün ***date*** değerini, belirli bir ***date*** değerine sahip veya bu değerden önce ya da sonra olacak şekilde döndürür (kullanılan sözdizimine bağlı olarak).

Sözdizimi	Tanımlama
*<from-date>***'e denk gelen yada önceki Pazar**	
SonrakiGün(*<date>*, *<day>*, *<month>*)	***date*** sonrasında bir sonraki belirtilen gün ve ay örneğini döndürür.
GünEkle(*<date/datetime>*, *<num_days>*) *<date><num_days>* **gün sonra tarih** *<date><num_days>* **gün önce tarih** (**saat** | **tarih**)*<datetime>* (**'da** | **'de** | **'ta** | **'te**)*<num_days>* **günden sonra**	Bir ***date*** özellik tipine gün sayısı ekler veya bu değerden gün sayısını çıkarır. Kısa sözdizimsel biçim kullanıldığında, ***date*** girdisine gün sayısını eklemek için bu sayının pozitif bir tamsayı olması gerekir; ***date*** girdisinden gün sayısını çıkarmak içinse bu sayının negatif bir tamsayı olması gerekir.
HaftaEkle(*<date/datetime>*, *<num_weeks>*) *<date><num_weeks>* **hafta sonra tarih** *<date><num_weeks>* **hafta önce tarih** (**saat** | **tarih**)*<datetime>* (**'da** | **'de** | **'ta** | **'te**)*<num_weeks>* **haftadan sonra**	***date*** özellik değerine hafta ekler. Kısa sözdizim formu kullanılırken ***date*** girdisine hafta eklenmesi için sayının pozitif tam sayı olması gerekir.
AyEkle(*<date/datetime>*, *<num_months>*) *<date><num_months>* **ay sonra tarih** *<date><num_months>* **ay önce tarih** (**saat** | **tarih**)*<datetime>* (**'da** | **'de** | **'ta** | **'te**)*<num_months>* **aydan sonra**	***date*** özellik değerine ay ekler. Kısa sözdizim formu kullanılırken ***date*** girdisine ay eklenmesi için sayının pozitif tam sayı olması gerekir.
YılEkle(*<date/datetime>*, *<num_years>*) *<date><num_years>* **yıl sonra tarih** *<date><num_years>* **yıl önce tarih**	***date*** özellik değerine yıl ekler. Kısa sözdizim formu kullanılırken ***date*** girdisine yıl eklenmesi için sayının pozitif tam sayı olması gerekir.

Sözdizimi	Tanımlama
(**saat** | **tarih**)*<datetime>* (**'da** | **'de** | **'ta** | **'te**)*<num_years>* **yıldan sonra**	
GünSayısı(*<date1>*, *<date2>*) *<date1>***'den** *<date2>***'a kadar kalan** [**(içeren)**] [**tam**] **haftalar** *<date1>* **ve** *<date2>***'un arasında kalan** [**(içeren)**][**tam**] **hafta içi günler** *<date1>*(**'yla** | **'yle**)*<date2>* **arasında kalan** [**dahil**] **hafta içi günlerin sayısı**	***date***1 ile ***date***2 arasındaki hafta içi günleri sayar. Bunlar Pazartesi ile Cuma arasındaki günlerdir. Not: Önceki ***date*** dahil, sonraki ***date*** hariçtir.
YılBaşlangıcı(*<date/datetime>*) **senenin ilk olan** *<from-date>***'e denk gelen günü**	Yıl içindeki ilk ***date*** değerini (***date*** öğesinin içinde bulunduğu) döndürür.
YılSonu (*<date/datetime>*) **senenin son olan** *<from-date>***'e denk gelen günü**	Yıl içindeki son ***date*** değerini (***date*** öğesinin içinde bulunduğu) döndürür.
GünFarkı(*<date/datetime1>*, *<date/datetime2>*) *<date1>***'dan** *<date2>***'a kadar** [**(içeren)**] **kalan** [**tam**] **günler** *<date1>* **ve** *<date2>***'un arasında kalan** [**tam**] [**(içeren)**] **günler** **gün sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ve ***date/datetime2*** arasındaki tam gün sayısını döndürür. İki günün sırası sonucu etkilemez.
GünFarkıDahil(*<date/datetime1>*, *<date/d-*	***date/datetime1*** ve ***date/datetime2*** dahil, aralarındaki tam gün sayısını döndürür. Bu hesaplama her iki ucu da dahil eder. Tarihler aynı olduğunda sonuç 1'dir. İki günün sırası sonucu etkile-

Sözdizimi	Tanımlama
atetime2>) **dahil gün sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	mez.
GünFarkıHariç(*<date/datetime1>*, *<date/datetime2>*) *<date1>***'dan** *<date2>***'a kadar (dışlayan) kalan** [**tam**] **günler** *<date1>* **ve** *<date2>***'un arasında kalan (dışlayan)** [**tam**] **günler** **hariç gün sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ve ***date/datetime2*** hariç, aralarındaki tam gün sayısını döndürür. Bu hesaplama her iki ucu da hariç tutar. Tarihler aynı olduğunda sonuç 0'dır. İki günün sırası sonucu etkilemez.
HaftaFarkı(*<date/datetime1>*, *<date/datetime2>*) *<date1>***'den** *<date2>***'a kadar** [**(içeren)**][**tam**] **haftalar** *<date1>* **ve** *<date2>***'un arasında kalan** [**(içeren)**][**tam**] **haftalar** **hafta sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ile ***date/datetime2*** arasındaki geçmiş tam haftaların sayısını döndürür. İki tarihin sırası sonucu etkilemez.
HaftaFarkıDahil(*<date/datetime1>*, *<date/datetime2>*) **dahil hafta sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ile ***date/datetime2*** dahil, aralarındaki geçmiş tam haftaların sayısını döndürür. İki tarihin sırası sonucu etkilemez.
HaftaFarkıHariç(*<date/datetime1>*, *<date/datetime2>*)	***date/datetime1*** ile ***date/datetime2*** hariç, aralarındaki geçmiş tam haftaların sayısını döndürür. İki tarihin sırası sonucu etkilemez.

Sözdizimi	Tanımlama
hariç hafta sayısı *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	
AyFarkı(*<date/datetime1>*, *<date/datetime2>*) *<date1>***'den** *<date2>***'a kadar kalan** [**(içeren)**] [**tam**] **aylar** *<date1>* **ve** *<date2>***'un arasında kalan** [**(içeren)**][**tam**] **ayları** **ay sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ve ***date/datetime2*** arasındaki tam geçmiş ay sayısını döndürür. İki tarihin sırası sonucu etkilemez.
AyFarkıDahil(*<date/datetime1>*, *<date/datetime2>*) **dahil ay sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ve ***date/datetime2*** dahil, aralarındaki tam geçmiş ay sayısını döndürür. İki tarihin sırası sonucu etkilemez.
AyFarkıHariç(*<date/datetime1>*, *<date/datetime2>*) **hariç ay sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***date/datetime1*** ve ***date/datetime2*** hariç, aralarındaki tam geçmiş ay sayısını döndürür. İki tarihin sırası sonucu etkilemez.
YılFarkı(*<date/datetime1>*, *<date/datetime2>*) *<date1>***'den** *<date2>***'a kadar kalan** [**(içeren)**] [**tam**] **yıllar** *<date1>* **ve** *<date2>***'un arasında kalan** [**(içeren)**][**tam**] **yıllar** *<date1>*(**'yla** | **'yle**)*<date2>* **arasında kalan yıl sayısı**	***date/datetime1*** ile ***date/datetime2*** arasındaki yıl sayısını döndürür. İki tarihin sırası sonucu etkilemez.
YılFarkıDahil(*<date/d-*	***date/datetime1*** ile ***date/datetime2*** dahil, aralarındaki yıl sayısını döndürür. İki tarihin sırası

Sözdizimi	Tanımlama
atetime1>, *<date/d-atetime2>*) *<date1>*(**'yla | 'yle**)*<date2>* **arasında kalan dahil yıl sayısı**	sonucu etkilemez.
YılFarkıHariç(*<date/d-atetime1>*, *<date/d-atetime2>*) *<date1>*(**'yla | 'yle**)*<date2>* **arasında kalan hariç yıl sayısı**	***date/datetime1*** ile ***date/datetime2*** hariç, aralarındaki yıl sayısını döndürür. İki tarihin sırası sonucu etkilemez.

Günün saati fonksiyonları(English)

Sözdizimi	Tanımlama
Saat(*<text>*)	Belirtilen dizeyi günün saati değerine dönüştürür
SaniyeAl(*<time/datetime>*)	***timeofday/datetime attribute*** özelliğinin saniye bileşenini döndürür.
DakikaAl(*<time/datetime>*)	***timeofday/datetime attribute*** özelliğinin dakika bileşenini döndürür.
SaatAl(*<time/datetime>*)	***timeofday/datetime attribute*** özelliğinin saat bileşenini döndürür.

Tarih ve saat fonksiyonları(English)

Sözdizimi	Tanımlama
GeçerliTarihSaat() **şimdiki zaman**	Oturumun başlangıcında geçerli ***date*** öğesi ile saati döndürür.
TarihSaat(*<text>*)	Belirtilen dizeyi ***datetime*** değerine dönüştürür
TarihSaatiBirleştir(*<date>*, *<time>*) *<date>* **da** *<time-of-day>* *<time-of-day>* **da** *<date>*	***date*** özellik değerini, ***date*** ve günün saati değerlerini birleştirerek ayarlar.
SaniyeFarkı(*<datetime1>*, *<datetime2>*) **SaniyeFarkı**(*<timeOfDay1>*, *<timeOfDay2>*) **saniye sayısı** *<date1>*(**'dan | 'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** arasındaki saniye sayısını döndürür.

Sözdizimi	Tanımlama
SaniyeFarkıDahil(*<datetime1>*, *<datetime2>*) **SaniyeFarkıDahil** (*<timeOfDay1>*, *<timeOfDay2>*) **dahil saniye sayısı** *<date1>* (**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** dahil, aralarındaki saniye sayısını döndürür.
SaniyeFarkıHariç(*<datetime1>*, *<datetime2>*) **SaniyeFarkıHariç** (*<timeOfDay1>*, *<timeOfDay2>*) **hariç saniye sayısı** *<date1>* (**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** hariç, aralarındaki saniye sayısını döndürür.
DakikaFarkı(*<datetime1>*, *<datetime2>*) **DakikaFarkı**(*<timeOfDay1>*, *<timeOfDay2>*) **dakika sayısı** *<date1>*(**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** arasındaki dakika sayısını döndürür.
DakikaFarkıDahil(*<datetime1>*, *<datetime2>*) **DakikaFarkıDahil** (*<timeOfDay1>*, *<timeOfDay2>*) **dahil dakika sayısı** *<date1>* (**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** dahil, aralarındaki dakika sayısını döndürür.
DakikaFarkıHariç(*<datetime1>*, *<datetime2>*) **DakikaFarkıHariç** (*<timeOfDay1>*, *<timeOfDay2>*) **hariç dakika sayısı** *<date1>* (**'dan** | **'den**)*<date2>*(**'a** | **'e**) **kadar**	***datetime1*** ve ***datetime2*** hariç, aralarındaki dakika sayısını döndürür.
SaatFarkı(*<datetime1>*, *<datetime2>*) **SaatFarkı**(*<timeOfDay1>*,	***datetime1*** ve ***datetime2*** arasındaki saat sayısını döndürür.

Sözdizimi	Tanımlama
<timeOfDay2>) **saat sayısı** *<date1>*(**'dan | 'den**)*<date2>*(**'a | 'e**) **kadar**	
SaatFarkıDahil(*<datetime1>*, *<datetime2>*) **SaatFarkıDahil** (*<timeOfDay1>*, *<timeOfDay2>*) **dahil saat sayısı** *<date1>* (**'dan | 'den**)*<date2>*(**'a | 'e**) **kadar**	***datetime1*** ve ***datetime2*** dahil, aralarındaki saat sayısını döndürür.
SaatFarkıHariç(*<datetime1>*, *<datetime2>*) **SaatFarkıHariç** (*<timeOfDay1>*, *<timeOfDay2>*) **hariç saat sayısı** *<date1>* (**'dan | 'den**)*<date2>*(**'a | 'e**) **kadar**	***datetime1*** ve ***datetime2*** hariç, aralarındaki saat sayısını döndürür.
TarihAl(*<datetime>*)	***date*** özelliğini ***datetime attribute*** özelliğinden çıkartır.
GünAl(*<datetime>*)	***datetime attribute*** özelliğinden günün saatini çıkartır. Saati geçerli ***date*** ve saat değerinden çıkartarak, ***timeofday attribute*** özelliğinin değerini kuralın yürütüldüğü saate ayarlamak için kullanılabilir.
SaatEkle(*<datetime>*, *<num_hours>*) **SaatEkle**(*<timeOfDay>*, *<num_hours>*) **saat** *<datetime>*(**'da | 'de | 'ta | 'te**)*<num_hours>* **saattan sonra**	***date*** özellik değerine saat ekler.
DakikaEkle(*<datetime>*, *<num_minutes>*) **DakikaEkle**(*<timeOfDay>*, *<num_minutes>*) **saat** *<datetime>*(**'da | 'de | 'ta | 'te**)*<num_minutes>* **dakikadan sonra**	***date*** özellik değerine dakika ekler.
SaniyeEkle(*<datetime>*, *<num_seconds>*) **SaniyeEkle**(*<timeOfDay>*,	***date*** özellik değerine saniye ekler.

Sözdizimi	Tanımlama
<num_seconds>) **saat** *<datetime>*(**'da | 'de | 'ta | 'te**)*<num_seconds>* **saniyeden sonra**	

Metin fonksiyonları(English)

Sözdizimi	Tanımlama
<text1> **&** *<text2>*	***text1*** metin değerini ***text2*** metin değeriyle birleştirip bu şekilde devam ederek tek bir ***text*** değeri oluşturur. Herhangi bir tipte değişken kullanabilirsiniz. Değerlerin formatı kural oturumunda yüklü olan formatlayıcı kullanılarak belirlenir.
bağlaması *<text1>* **&** *<text2>*	***text1*** metin değerini ***text2*** metin değeriyle birleştirip bu şekilde devam ederek tek bir ***text*** değeri oluşturur. Herhangi bir tipte değişken kullanabilirsiniz. Değerlerin formatı kural oturumunda yüklü olan formatlayıcı kullanılarak belirlenir.
İçerir(*<text>*, *<substring>*)	Belirtilen ***text*** değerinin belirtilen ***text*** alt dizesini içerip içermediğini gösteren bir mantıksal değer döndürür. ***text*** karşılaştırması büyük küçük harfe duyarlı değildir.
İleBiter(*<text>*, *<substring>*)	Belirtilen ***text*** değerinin belirtilen ***text*** alt dize ile sonlanıp sonlanmadığını gösteren bir mantıksal değer döndürür. ***text*** karşılaştırması büyük küçük harfe duyarlı değildir.
Sayıdır(*<text>*)	Belirtilen ***text*** değerinin geçerli bir sayıyı temsil edip etmediğini gösteren bir mantıksal değer döndürür.
Uzunluk(*<text>*)	Belirtilen ***text*** değerinin karakter uzunluğunu döndürür.
İleBaşlar(*<text>*, *<substring>*)	Belirtilen ***text*** değerinin belirtilen ***text*** alt dize ile başlayıp başlamadığını gösteren bir mantıksal değer döndürür. ***text*** karşılaştırması büyük küçük harfe duyarlı değildir.
Altdize(*<text>*, *<offset>*, *<length>*)	Belirtilen karakter uzunluğu olan, başlangıç noktasında başlayan ***text*** alt dizesini döndürür. Dizenin sonuna ulaşıldıysa daha az sayıda karakter döndürülür.
Metin(*<number>*) **Metin**(*<date>*) **Metin**(*<datetime>*) **Metin**(*<timeOfDay>*)	Belirtilen sayısı veya ***date attribute*** değerini ***text*** değerine dönüştürür.

Öğe ve ilişki fonksiyonları(English)

Sözdizimi	Tanımlama
İçin(*<relationship>*,	Sadece bir koşul olduğunda, bir "Tek - Tek" veya "Çok - Tek" ***relationship*** ilişkisinde ***entity***

Sözdizimi	Tanımlama
<Exp>) *<val>*, *<ent>* **durumda**	öğesinden başka bir ***entity*** öğesine başvuru için kullanılır.
Kapsamİçin(*<relationship>*, *<alias>*) **Kapsamİçin**(*<relationship>*)	Bir veya daha çok koşul olduğunda, bir "Tek - Tek", "Çok - Tek" veya "Çok - Çok" ***relationship*** ilişkisinde ***entity*** öğesinden başka bir ***entity*** öğesine başvuru için kullanılır.
Tümüİçin(*<relationship>*, *<Exp>*)	Bir "Tek - Tek" veya "Çok - Çok" ***relationship*** ilişkisinde bir ***entity*** öğesinden başka bir ***entity*** öğesine başvuru için kullanılır (hedef ***entity*** grubunun tüm üyelerinin ilgili kuralı karşılaması gerekip gerekmediğini belirlemeniz gerektiğinde). Bu form, kuralda sadece tek bir koşul olduğunda kullanılır.
TümKapsamİçin(*<relationship>*) **TümKapsamİçin**(*<relationship>*, *<alias>*)	Bir "Tek - Çok" veya "Çok - Çok" ***relationship*** ilişkisinde bir ***entity*** öğesinden başka bir ***entity*** öğesine başvuru için kullanılır (hedef ***entity*** grubunun tüm üyelerinin ilgili kuralı karşılaması gerekip gerekmediğini belirlemeniz gerektiğinde). Bu form, kuralda bir veya birden fazla koşul olduğunda kullanılır.
Var(*<relationship>*, *<Exp>*)	Bir "Tek - Çok" veya "Çok - Çok" ***relationship*** ilişkisinde bir ***entity*** öğesinden başka bir ***entity*** öğesine başvuru için kullanılır (hedef ***entity*** grubunun herhangi bir üyesinin ilgili kuralı karşılaması gerekip gerekmediğini belirlemeniz gerektiğinde). Bu form, kuralda sadece tek bir koşul olduğunda kullanılır.
KapsamVar(*<relationship>*) **KapsamVar**(*<relationship>*, *<alias>*)	Bir "Tek - Çok" veya "Çok - Çok" ***relationship*** ilişkisinde bir ***entity*** öğesinden başka bir ***entity*** öğesine başvuru için kullanılır (hedef ***entity*** grubunun herhangi bir üyesinin ilgili kuralı karşılaması gerekip gerekmediğini belirlemeniz gerektiğinde). Bu form, kuralda bir veya birden fazla koşul olduğunda kullanılır.
Üyesi:(*<target>*, *<relationship>*) **Üyesi:**(*<target>*, *<alias>*, *<relationship>*)	Bir ***entity*** örneğinin ***relationship*** üyesi olduğu anlamına ulaşmak için sonuç olarak kullanılır. ***entity*** örneğinin ***relationship*** öğesinin bir hedefi olduğunu test etmek için kullanılır (bu öğede ikinci bir ***entity*** örneği kaynak durumundadır).
ÜyesiDeğil:(*<target>*, *<relationship>*)	Bir ***entity*** örneğinin, ikinci bir ***entity*** örneğinin kaynak olduğu ***relationship*** ilişkisinde hedef olmadığını test etmede koşul olarak kullanılır.
KopyaSay(*<relationship>*) *<ent>*(**sayısı** | **sayısının ortalaması**) **sayisi** *<relationship>*	***entity*** için mevcut örnek sayısını sayar.
KopyaSayEğer(*<relationship>*, *<Exp>*) **sayisi** *<relationship>* **durum böyle olduğu için** *<condition>*	Belirli bir ***entity-level attribute*** özelliğinin belirli bir değere sahip olduğu ***entity*** öğesinin örnek sayısını sayar.

Sözdizimi	Tanımlama
MaksimumKopya(*<relationship>*, *<number-attr>*) **MaksimumKopya**(*<relationship>*, *<date-attr>*) **MaksimumKopya**(*<relationship>*, *<datetime-attr>*) **MaksimumKopya**(*<relationship>*, *<time-attr>*) *<ent>* **için en kücük** *<date-attr>* **bütün** *<ent>* **için en yaşlı** *<attr>* **bütün** *<ent>* **için en geç** *<attr>* **bütün** *<ent>* **için en büyük** *<attr>*	Tüm ***entity*** örnekleri için ***entity-level*** değişkeninin en yüksek/en son değerini alır.
KopyaMaksimumEğer (*<relationship>*, *<number-attr>*, *<condition>*) **KopyaMaksimumEğer** (*<relationship>*, *<date-attr>*, *<condition>*) **KopyaMaksimumEğer** (*<relationship>*, *<datetime-attr>*, *<condition>*) **KopyaMaksimumEğer** (*<relationship>*, *<time-attr>*, *<condition>*)	Belirli bir ***entity-level attribute*** özelliğinin belirli bir değere sahip olduğu tüm ***entity*** örnekleri için ***entity-level*** değişkeninin en yüksek/en son değerini alır.
MinimumKopya(*<relationship>*, *<number-attr>*) **MinimumKopya**(*<relationship>*, *<date-attr>*) **MinimumKopya**(*<relationship>*, *<datetime-attr>*) **MinimumKopya**(*<relationship>*, *<time-attr>*) **bütün** *<ent>* **için en küçük** *<attr>* **bütün** *<ent>* **için en erken** *<attr>*	Tüm ***entity*** örnekleri için ***entity-level*** değişkeninin en düşük/en eski değerini alır.
KopyaMinimumEğer	Belirli bir ***entity-level attribute*** özelliğinin belirli bir değere sahip olduğu tüm ***entity*** örnekleri için ***entity-level*** değişkeninin en düşük/en eski değerini alır.

Sözdizimi	Tanımlama
(*<relationship>*, *<number-attr>*, *<condition>*) **KopyaMinimumEğer** (*<relationship>*, *<date-attr>*, *<condition>*) **KopyaMinimumEğer** (*<relationship>*, *<datetime-attr>*, *<condition>*) **KopyaMinimumEğer** (*<relationship>*, *<time-attr>*, *<condition>*) **eğer** *<ent-test>***bütün** *<ent>* **için en küçük** *<attr>*	
KopyaToplam(*<relationship>*, *<number-attr>*) **bütün** *<ent>* **toplam** *<num-attr>* **bütün** *<ent>* **toplam** *<attr>*	***entity-level*** değişkeninin tüm örneklerinin toplamını alır.
KopyaToplamEğer(*<relationship>*, *<number-attr>*, *<condition>*)	Belirli bir ***entity-level*** Mantıksal değer ***attribute*** özelliği değerinin doğru olduğu ***entity*** öğesi için ***entity-level*** değişkeninin tüm örneklerinin toplamını alır.
KopyaDeğeriEğer(*<relationship>*, *<number-attr>*, *<condition>*) **KopyaDeğeriEğer**(*<relationship>*, *<text-attr>*, *<condition>*) **KopyaDeğeriEğer**(*<relationship>*, *<date-attr>*, *<condition>*) **KopyaDeğeriEğer**(*<relationship>*, *<datetime-attr>*, *<condition>*) **KopyaDeğeriEğer**(*<relationship>*, *<time-attr>*, *<condition>*)	Koşul bazında ***relationship*** ile ilişkili hedef ***entity*** örneklerinden belirlenen benzersiz bir ***entity*** örneğinden değer alır. • Koşul tek hedef ***entity*** örneğini tanımlıyorsa değer, söz konusu ***entity*** örneğine göre hesaplanan değerdir. • Koşulu karşılayan birden fazla hedef örnek varsa, ***uncertain*** değeri döndürülür. • Koşulu karşılayan herhangi bir hedef örnek yoksa ve ***relationship*** biliniyorsa, değer ***uncertain*** olur.
KopyaEşittir (*<instance1>*, *<instance2>*)	İki ***entity*** örneğinin aynı örnek olup olmadığını belirler.

Sözdizimi	Tanımlama
KopyaEşitDeğildir(*<instance1>*, *<instance2>*)	İki ***entity*** örneğinin aynı örnek olmadığını belirler.
AnlamKopya(*<relationship>*, *<identity>*) *<rel>***(***<identity>***)** (**şeklindedir** | **olarak tanımlıdır**) *<rel>* **içinde (***<identity>***)** (**bulunmaktadır** | **mevcuttur**)	Bir ***entity*** örneğinin mevcut olduğu ve bir ***relationship*** üyesi olduğu anlamına ulaşmak için sonuç olarak kullanılır.

Zamana dayalı mantık fonksiyonları(English)

Sözdizimi	Tanımlama
AralıkSayıAyrı(*<start-date>*, *<end-date>*, *<variable>*) **AralıkSayıAyrı**(*<start-date>*, *<end-date>*, *<condition>*)	Başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişken ile ilgili bilinen ayrı değer sayısını sayar.
AralıkSayıAyrıEğer(*<start-date>*, *<end-date>*, *<variable>*, *<condition>*)	Sadece mantıksal değer filtresi değerinin doğru olduğu zamanları dahil ederek, başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişken ile ilgili bilinen ayrı değer sayısını sayar.
AralıkGünlükToplam(*<start-date>*, *<end-date>*, *<number-attr>*)	Başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki para birimi veya sayı değişkeninin toplamını hesaplar. ***attribute*** değerinin günlük miktar olduğu varsayılır.
AralıkGünlükToplamEğer(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*)	Sadece koşul değerinin doğru olduğu zamanları dahil ederek, başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki para birimi veya sayı değişkeninin tüm günlük değerlerinin toplamını hesaplar.
AralıkMaksimum(*<start-date>*, *<end-date>*, *<number-attr>*) **AralıkMaksimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **AralıkMaksimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **AralıkMaksimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişkenin maksimum değerini seçer.
AralıkMaksimumEğer(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **AralıkMaksimumEğer**(*<start-date>*,	Sadece koşul değerinin doğru olduğu zamanları dahil ederek, başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişkenin maksimum değerini seçer.

Sözdizimi	Tanımlama
<end-date>, *<date-attr>*, *<condition>*) **AralıkMaksimumEğer**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **AralıkMaksimumEğer**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	
AralıkMinimum(*<start-date>*, *<end-date>*, *<number-attr>*) **AralıkMinimum**(*<start-date>*, *<end-date>*, *<date-attr>*) **AralıkMinimum**(*<start-date>*, *<end-date>*, *<datetime-attr>*) **AralıkMinimum**(*<start-date>*, *<end-date>*, *<time-attr>*)	Başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişkenin minimum değerini seçer.
AralıkMinimumEğer(*<start-date>*, *<end-date>*, *<number-attr>*, *<condition>*) **AralıkMinimumEğer**(*<start-date>*, *<end-date>*, *<date-attr>*, *<condition>*) **AralıkMinimumEğer**(*<start-date>*, *<end-date>*, *<datetime-attr>*, *<condition>*) **AralıkMinimumEğer**(*<start-date>*, *<end-date>*, *<time-attr>*, *<condition>*)	Sadece koşul değerinin doğru olduğu zamanları dahil ederek, başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki değişkenin minimum değerini seçer.
AralıkAğırlıklıOrtalama(*<start-date>*, *<end-date>*, *<number-attribute>*)	Başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki para birimi veya sayı değişkeninin ortalama değerini her değerin geçerli olduğu zaman aralığına göre hesaplar.
AralıkAğırlıklıOrtalamaEğer(*<start-date>*, *<end-date>*, *<number-attribute>*, *<condition>*)	Sadece mantıksal değer koşulunun doğru olduğu (her değerin geçerli olduğu zaman aralığına göre) zamanları dahil ederek, başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki para birimi veya sayı değişkeninin ortalama değerini hesaplar.
ZamanaDayalıHerZaman(*<start-date>*, *<end-date>*, *<condition>*)	Sadece mantıksal değer başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki tüm zamanlar için doğru ise, doğru değerini döndürür.
ZamanaDayalıEnAzGünler(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Sadece mantıksal değer başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki en azından belirlenen gün sayısı kadar bir süre için (sıralı olması gerekmez) doğru ise, doğru değerini döndürür.
ZamanaDayalıTakipEdenGünler(*<start-date>*, *<end-date>*, *<NumDays>*, *<condition>*)	Sadece mantıksal değer başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıktaki en azından belirlenen sıralı gün sayısı kadar bir süre için doğru ise, doğru değerini döndürür.
ZamanaDayalıBazen(*<start-date>*, *<end-date>*, *<condition>*)	Sadece başlangıç ***date*** değerinden (dahil) bitiş ***date*** değerine (hariç) olan aralıkta mantıksal değer koşulu doğru ise, doğru değerini döndürür.
BuradaDeğer(*<date>*, *<value>*)	Belirtilen ***attribute*** için belirtilen ***date*** öğesindeki değeri döndürür.

Sözdizimi	Tanımlama
NezamanEnson(*<date>*, *<condition>*)	Mantıksal değer koşulunun en son kez için doğru olduğu ***date*** değerini döndürür. Belirtilen bir ***date*** değerinden geriye dönüktür ve bu değeri de içerir.
NezamanSonraki(*<date>*, *<condition>*)	Mantıksal değer koşulunun bir sonraki kez için doğru olacağı ***date*** değerini döndürür. Belirtilen bir ***date*** değerinden ileriye dönüktür ve bu değeri de içerir.
EnGeç()	***date*** değerini en geç ***date*** değerine eşdeğer olacak şekilde döndürür, yani ***date*** değerinin herhangi başka bir ***date*** değerinden sonra olacak ve bir ***date attribute*** değerinin alınabileceği veya bir ifadenin değerlendirilebileceği şekilde.
EnErken()	***date*** değerini en erken ***date*** değerine eşdeğer olacak şekilde döndürür, yani ***date*** değerinin herhangi başka bir ***date*** değerinden önce olacak ve bir ***date attribute*** değerinin alınabileceği veya bir ifadenin değerlendirilebileceği şekilde.
ZamanaDayalıİtibarenGün(*<date>*, *<end-date>*)	Her gün değişen bir sayı değişkeni döndürür ve ***date*** sonrasındaki tam gün sayısıdır.
ZamanaDayalıİtibarenHafta(*<date>*, *<end-date>*)	Her hafta değişen bir sayı değişkeni döndürür ve ***date*** sonrasındaki tam hafta sayısıdır.
ZamanaDayalıİtibarenAy(*<date>*, *<end-date>*)	Her ay değişen bir sayı değişkeni döndürür ve bu ***date*** sonrasındaki tam ay sayısıdır. Not: Belirtilen ***date***, ayın 28. gününden sonra olduğunda ve bir sonraki ay belirtilen aydan daha az gün içerdiğinde ayın son gününde yıldönümü ay için değişiklik noktası oluşturulur. Örneğin belirtilen ***date*** 28, 29, 30 veya 31 Ocak 2007 ise ilk değişiklik noktası 28 Şubat 2007 olur.
ZamanaDayalıİtibarenYıl(*<date>*, *<end-date>*)	Her yıl değişen bir sayı değişkeni döndürür ve ***date*** sonrasındaki tam yıl sayısıdır.
ZamanaDayalıHerzamanGünler (*<days>*, *<condition>*)	Zaman içinde değişen ve sadece geçerli gün hariç olmak üzere belirtilen gün sayısı kadar geçmiş günlerin tümünde mantıksal değer koşulu doğru ise doğru değerini veren ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıİzleyenGünler (*<minDays>*, *<days>*, *<condition>*)	Zaman içinde değişen ve sadece geçerli gün hariç olmak üzere, önceki gün sayısı içinde herhangi bir aralıkta, en az belirli bir sayıda sıralı gün için mantıksal değer koşulu doğru ise doğru değerini veren ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıBazenGünler(*<days>*, *<condition>*)	Zaman içinde değişen ve geçerli gün hariç olmak üzere, belirtilen sayıda geçmiş gün içinde sadece mantıksal değer koşulu bir kez doğru ise, doğru olan ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıSonra(*<date>*)	Zaman içinde değişen ve ***date*** tarihinde veya öncesinde yanlış olan sonrasında ise doğru olan ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıÖnce(*<date>*)	Zaman içinde değişen ve ***date*** değerinde veya sonrasında yanlış olan öncesinde ise doğru olan ***attribute*** mantıksal değerini döndürür.

Sözdizimi	Tanımlama
ZamanaDayalıSırasında(*<date>*)	Zaman içinde değişen ve ***date*** öncesinde veya sonrasında yanlış olan tam değerde ise doğru olan ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıSırasındaVeyaSonra (*<date>*)	Zaman içinde değişen ve ***date*** değerinde veya sonrasında doğru, öncesinde yanlış olan ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıSırasındaVeyaÖncesinde (*<date>*)	Zaman içinde değişen ve ***date*** değerinde veya öncesinde doğru olan, sonrasında ise yanlış olan ***attribute*** mantıksal değerini döndürür.
ZamanaDayalıBaşlangıçTarihinden (*<relationship>*, *<date>*, *<value>*)	***relationship*** öğesinden tek zamana dayalı ***attribute*** (kaynak ***entity*** düzeyinde) ve öğelerde değer ***attribute*** döndürür; başlangıç ***date attribute***sonrasında etkin olan değerler içerir.
ZamanaDayalıBitişTarihinden(*<relationship>*, *<date>*, *<value>*)	***relationship*** öğesinden tek zamana dayalı ***attribute*** (kaynak ***entity*** düzeyinde) ve öğelerde değer ***attribute*** döndürür; son ***date attribute*** öncesine kadar etkin olan değerler içerir.
ZamanaDayalıAralıktan(*<relationship>*, *<start-date>*, *<end-date>*, *<Value>*)	***relationship*** öğesinden tek zamana dayalı ***attribute*** (kaynak ***entity*** düzeyinde) ve öğelerde değer ***attribute*** döndürür; başlangıç ***date attribute*** (dahil) ile bitiş ***date attribute*** (hariç) arasında etkin olan değerler içerir. Bir sonraki başlangıç ***date*** öncesinde süresi dolarsa değer ***uncertain*** olur.
ZamanaDayalıHaftaGünü(*<startdate>*, *<enddate>*)	Belirtilen başlangıç ***date*** (dahil) ile bitiş ***date*** (hariç) arasında hafta içi günlerinde doğru ve haftasonu günlerinde yanlış değerini döndürür. Aralık ***date*** dışında ***uncertain*** döndürür.
ZamanaDayalıAydaBir(*<startdate>*, *<enddate>*, *<dayofmonth>*)	Belirtilen başlangıç ***date*** (dahil) ile bitiş ***date*** (hariç) arasında gün, ayın günü parametresi ile aynı ise doğru diğer tüm günlerde yanlış değerini döndürür. Aralık ***date*** dışında ***uncertain*** döndürür. Ayın günü parametresi geçerli aydaki gün sayısını aşarsa değer, her ay mutlaka en az bir gün doğru değeri olabilmesi için o ayın son gününde doğru olur.

Doğrulama etkinliği fonksiyonları(English)

Sözdizimi	Tanımlama
Hata (*<text>*)	Kullanıcıya mesaj aktarmak ve hatayı tetikleyen koşul ortadan kalkana kadar kullanıcıların araştırmaya devam etmesini önlemek amacıyla bir hata olayı kullanıldı.
Uyarı (*<text>*)	Kullanıcıya mesaj aktarmak ve uyarıyı tetikleyen koşula rağmen kullanıcıların ilerlemesini önlemek amacıyla bir uyarı olayı kullanıldı.

Kullanımdan kalkmış fonksiyonlar(English)

Sözdizimi	Tanımlama
ÖzelFonksiyonÇağır	Dış çağrı sonucunu kod kitaplığına döndürür. Özel fonksiyon çağrısının başarılı olması için belir-

Sözdizimi	Tanımlama
(*<A>*, *<B>*)	lenimler altyapısına kod kitaplığı sağlanmalıdır.

Structural configuration settings

The following settings are used for modeling the structure of legislation. These elements must be formatted using the **Configuration** 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. You may specify multiple Default_structural_element entries in a single rule document to apply to all rules following each entry. 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.
Default_structural_globalproof	Default_structural_globalproof[<replacement structural text including structural element ^x>]	Used to bypass the default text ("^x is satisfied") generated for structural elements. You may specify multiple Default_structural_globalproof entries in a single rule document to apply to all rules following each entry. Note that this works in conjunction with Default_structural_element, which is used to define the ^x form.
Default_structural_entityproof	Default_structural_entityproof[<replacement structural text including structural elements ^x and ^entity>]	Used to bypass the default text ("^entity satisfies ^x ") generated for entity-level structural elements.

Element	Syntax	Description
		You may specify multiple Default_structural_entityproof entries in a single rule document to apply to all rules following each entry. 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 ^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	**the pre-2007 rules apply if** the date of claim < 2007-01-01
Greater than	**the new rates apply if** the date of investigation > 2007-06-30
Less than or equal to	**the person can apply for Immunization Allowance for the child if** the date of claim <= the child's second birthday
Greater than or equal to	**the individual qualifies for age pension if** the individual's age >= 65
Equals	**the person is 18 if** the person's age = 18
Not equal to	**the person's salary has been adjusted if** 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	**today = the current date** **the date of the investigation = CurrentDate()**	today's date: 2005-04-15 today's date: 2009-08-31	today = 2005-04-15 the date of the investigation = 2009-08-31	Get today's date
Date	**the date of effect =Date ("2012-01-01")**	2012-01-01	the date of effect = 2012-01-01	Convert a text string into a number or date
MakeDate	**the calculation date =MakeDate(2006, 10, 19)**	yyyy: 2006; mm: 10; dd: 19	the calculation date = 2006-10-19	Get a date formed from a specified year, month and day
ExtractDay	**the day of expiry = ExtractDay (the use-by date on the packet)**	the use-by date on the packet: 2008-06-12	the day of expiry = 12	Get the day component of an input date
ExtractMonth	**the month of expiry = ExtractMonth(the use-by date on the packet)**	the use-by date on the packet: 2007-04-16	the month of expiry = 04	Get the month component of an input date
ExtractYear	**the year of expiry = ExtractYear(the use-by date on the packet)**	the use-by date on the packet: 2009-02-21	the year of expiry = 2009	Get the year component of an input date
NextDayOfTheWeek	**next Monday = NextDayOfTheWeek(the current date,"Monday")**	the current date: 2009-08-09	next Monday = 10 August	Get the date of the next or previous spe-

Function	Example rule	Inputs	Outputs	Further information
	next Tuesday = the next Tuesday on or after the current date **last Thursday = the Thursday on or before the current date**		2009 next Tuesday = 11 August 2009 last Thursday = 6 August 2009	cified day
AddDays	**the date that the library book must be returned by = AddDays(the date of loan,21)** **the date that the library book must be returned by = the date 21 days after the date of loan** **the closing date for the entry = AddDays(the date of the show,-10)**	the date of loan: 2006-01-04 the date of the show: 2007-05-15	the date that the library book must be returned by = 25 the closing date for the entry = 2007-05-05	Add or subtract a specified number of days to an input date
AddWeeks	**the date that the event finishes = AddWeeks(the date that the event begins,the number of weeks in the event)** **the date that the event finishes = the date the number of weeks in the event weeks after the date that the event begins** **the start date of the 5 week promotion = AddWeeks(the end date of the promotion,-5)** **the start date of the 5 week promotion = the date 5 weeks before the end date of the promotion**	the date that the event begins: 2001-08-13 the number of weeks in the event: 12 the end date of the promotion: 2008-12-24	the date that the event finishes = 2001-11-05 the start date of the 5 week promotion = 2008-11-19	Add or subtract a specified number of weeks to an input date
AddMonths	**the date that the player can return from suspension = AddMonths(the date of the suspension,3)** **the date that the player can return from suspension = the date 3 months after the date of the suspension**	the date of the suspension: 2005-12-12 the end date of the player's contract: 2006-06-	the date that the player can return from suspension = 2006-03-12	Add or subtract a specified number of months to an input date

Function	Example rule	Inputs	Outputs	Further information
	the start date of the player's 12 month contract = AddWeeks(the end date of the player's contract,-12) **the start date of the player's 12 month contract = the date 12 months before the end date of the player's contract**	30	the start date of the player's 12 month contract = 2005-06-30	
AddYears	**the date of the trial = AddYears(the date of the crime,3)** **the date of the trial = the date 3 years after the date of the crime** **the date that the prison sentence starts = AddYears(the date that the prison sentence ends,-20)** **the date that the prison sentence starts = the date 20 years before the date that the prison sentence ends**	the date of the crime: 2002-01-01 the date that the prison sentence ends: 1980-03-16	the date of the trial = 2005-01-01 the date that the prison sentence starts = 1960-03-16	Add or subtract a specified number of years to an input date
YearStart	**the start of the first relevant year = YearStart(2009-09-09)** **the start of the second relevant year = the first day of the year in which the date of the grand occasion falls**	the date of the grand occasion: 2007-09-09	the start of the first relevant year = 2009-01-01 the start of the second relevant year = 2007-01-01	Find the first date in the year
YearEnd	**the end of the relevant year = YearEnd(the relevant date)** **the end of the relevant year = the last day of the year in which the relevant date falls**	the relevant date: 2005-10-15	the end of the relevant year = 2005-12-31	Find the last date in the year
NextDate	**the end of the Australian tax year = NextDate(the test date, 30, 6)**	the test date: 2005-07-02	the end of the Australian tax	Find the next instance of the given

Function	Example rule	Inputs	Outputs	Further information
			year = 2006-06-30	day/month
UKTaxYearDates	**the date of effect = the next UK tax year end date on or after the test date** **the assessment date = the previous UK tax year start date on or before the test date**	the test date: 2003-09-21	the date of effect = 2004-04-05 the assessment date = 2003-04-06	Find the start or the end date for the previous or next UK tax year
WeekdayCount	**the number of working days until my holiday = WeekdayCount(2007-12-03, 2007-12-13)** **the number of business days in the specified period = the number of weekdays (inclusive) between 2007-10-15 and 2007-10-31**	date1: 2007-12-03; date2: 2007-12-13 date1: 2007-10-15; date2: 2007-10-31	the number of working days until my holiday = 8 the number of business days in the specified period = 12	Count the number of weekdays between two dates
DayDifference	**the number of days in the assessment period = DayDifference(2006-10-01,2006-10-14)**	date1: 2006-10-01 date2: 2006-10-14	the number of days in the assessment period = 13	Count the number of whole days between two dates
DayDifferenceInclusive	**the number of days in the assessment period = DayDifferenceInclusive(2006-10-01,2006-10-14)**	date1: 2006-10-01 date2: 2006-10-14	the number of days in the assessment period = 14	
DayDifferenceExclusive	**the number of days in the assessment period = DayDifferenceExclusive(2006-10-01,2006-10-14)**	date1: 2006-10-01 date2: 2006-10-14	the number of days in the assessment period = 12	
WeekDifference	**the number of weeks until Christmas = WeekDifference (the current date,2011-12-25)**	the current date: 2011-11-25	the number of weeks remaining = 4	Count the number of whole weeks between two dates

Function	Example rule	Inputs	Outputs	Further information
WeekDifferenceInclusive	**the number of weeks until Christmas = WeekDifferenceInclusive(the current date,2011-12-25)**	the current date: 2011-11-25	the number of weeks remaining = 5	
WeekDifferenceExclusive	**the number of weeks until Christmas = WeekDifferenceExclusive(the current date,2011-12-25)**	the current date: 2011-11-25	the number of weeks remaining = 3	
MonthDifference	**the number of months remaining in the phone contract = MonthDifference(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 = MonthDifferenceInclusive(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 = MonthDifferenceExclusive(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 = YearDifference(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 = YearDifferenceInclusive(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 was assessed:	the age of the tree in years = 4	

Function	Example rule	Inputs	Outputs	Further information
	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	2003-12-12		
YearDifferenceExclusive	**the age of the tree in years = YearDifferenceExclusive(the date the tree was planted, the date the tree was assessed)** **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**	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 = 2	

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	**the latest submission time = TimeOfDay("12:30:00")**	12:30:00	the latest submission time = 12:30:00	Get the time of day from a text string
ExtractSecond	**the second component of the submission time = ExtractSecond(the sub-**	the submission time: 14:42:32	the second component	Get the second com-

Function	Example rule	Inputs	Outputs	Further information
	mission time)		of the submission time = 32	ponent of an input time
ExtractMinute	**the minute component of the submission time = ExtractSecond(the submission time)**	the submission time: 14:42:32	the minute component of the submission time = 42	Get the minute component of an input time
ExtractHour	**the hour component of the submission time = ExtractHour(the submission time)**	the submission time: 14:42:32	the hour component of the submission time = 14	Get the hour component of an input 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

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	**the date and time of the investigation = CurrentDateTime()**	the current date time: 2009-09-15 03:24:12	the date and time of the investigation = 2009-09-15 03:24:12	Get the current date and time
DateTime	**the latest submission date and time= DateTime(the submission date and time spe-**	the submission date and time spe-	the latest submission	Get a date and time from

Function	Example rule	Inputs	Outputs	Further information
	cified on the application form)	cified on the application form: 2012-12-31 18:00:00	date and time= 2012-12-31 18:00:00	a text string
ConcatenateDateTime	**the latest submission time = ConcatenateDateTime(the submission date, the submission closing time)**	the submission date: 2010-01-15 the submission closing time: 17:00:00	the latest submission time = 2010-01-15 17:00:00	Get a date and time by joining together a separate date and time
SecondDifference	**the number of seconds between first place and second place = SecondDifference(the first place time, the second place time)**	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	**the number of seconds between first place and second place = SecondDifferenceInclusive(the first place time, the second place time)**	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	**the number of seconds between first place and second place = SecondDifferenceExclusive(the first place time, the second place time)**	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	**the number of minutes late the plumber is = MinuteDifference(the time the plumber was meant to arrive, the time that the plumber actually arrived)**	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	**the number of minutes late**	the time the	the number	

Function	Example rule	Inputs	Outputs	Further information
	the plumber is = MinuteDifferenceInclusive(the time the plumber was meant to arrive, the time that the plumber actually arrived)	plumber was meant to arrive: 2009-10-18 08:30:00 the time that the plumber actually arrived: 2009-10-18 09:00:40	of minutes late the plumber is = 31	
MinuteDifferenceExclusive	**the number of minutes late the plumber is = MinuteDifferenceExclusive(the time the plumber was meant to arrive, the time that the plumber actually arrived)**	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 = 29	
HourDifference	**the number of hours the plane was delayed by = HourDifference(the scheduled arrival time of the flight, the arrival time of the delayed flight)**	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	Count the number of whole hours between two times
HourDifferenceInclusive	**the number of hours the plane was delayed by = HourDifferenceInclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)**	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	
HourDifferenceExclusive	**the number of hours the plane was delayed by = HourDifferenceExclusive(the scheduled arrival time of the flight, the arrival time of the delayed flight)**	the scheduled arrival time of the flight: 2006-10-13 09:50:00 the arrival time of the delayed flight:	the number of hours the plane was delayed by = 0	

Function	Example rule	Inputs	Outputs	Further information
		2006-10-13 11:00:00		
ExtractDate	**the password expiry date = ExtractDate(the password expiry date time)**	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	**the time of the assessment = ExtractTimeOfDay(the current date time)**	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
AddHours	**the date time that the offer expires = AddHours(the date time that the offer starts, 48)**	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	**the date time that the train is due = the time 25 minutes after the date time that the train departed**	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	**the date time at the start of the recording = the time 40 seconds before the date time at the end of the recording**	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	**the number =Number(the number text)**	the number text: 15	the number = 15
Addition	**the total = the first amount + the second amount**	the first amount: 2; the second amount: 3	the total = 5
Subtraction	**the total = the first amount - the second amount**	the first amount: 100; the second amount: 5	the total = 95
Multiplication	**the total = the first amount * the second amount**	the first amount: 7; the second amount: 2	the total = 14
Division	**the total = the first amount / the second amount**	the first amount: 10; the second amount: 5	the total = 2
Integer Division	**the result = the value \ 5**	the value: 54.25	the result = 10
Remainder after Integer Division	**the total = the first amount modulo the second amount**	the first amount: 9; the second amount: 3	the result = 0
Maximum	**the highest number of fish caught = Maximum (the number of fish caught by Bob, the number of fish caught by Mary)**	the number of fish caught by Bob: 8; the number of fish caught by Mary: 7	the highest number of fish caught =8
Minimum	**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**	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

Function	Example rule	Inputs	Outputs
Exponentiation (xy)	**the result = Xy(the value,3)**	the value: 5	the result = 125
Mathematical Constant (ex)	**the result = Ex(the value)**	the value: 0.3527	the result = 1.42290420813407
Absolute value	**the result = Abs(the value)**	the value: -80	the result = 80
Natural Logarithm	**the result = Ln(the value)**	the value: 0.3527	the result = -1.04213744174013
Logarithm Base	**the result = Log(the value)**	the value: 0.3527	the result = -0.45259454033251
Square Root	**the result = Sqrt(the value)**	the value: 64	the result = 8
Round	**the result = Round(the value,3)**	the value: 2.45678	the result = 2.457
Truncation	**the result = Trunc(the value,1)**	the value: 64.4657	the result = 64.4
Sine	**the result = Sin(the value)**	the value: 0.3527	the result = 0.345432860836779
Cosine	**the result = Cos(the value)**	the value: 0.3527	the result = 0.938443465880667
Tangent	**the result = Tan(the value)**	the value: 0.3527	the result = 0.368091284553421
Inverse Sine	**the result = Asin(the value)**	the value: 0.3527	the result = 0.360454968099581
Inverse Cosine	**the result = Acos(the value)**	the value: 0.3527	the result = 1.21034135869532
Inverse Tangent	**the result = Atan(the value)**	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	**the account may be a benefit account if** Contains(the account name, "benefit")	the account name: "Special Benefits"	the account may be a benefit account = true	Check if a text string contains a particular substring
StartsWith	**the person should be represented if** StartsWith(the person's name, "Sir")	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	**the product uses the ascending sort code if** EndsWith(the product code, "-ASC")	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 true string
IsNumber	**the postcode is a valid Australian postcode if** IsNumber(the postcode) and Length(the postcode) = 4	the postcode: "2612"	the postcode is a valid Australian postcode = true	Check if a text string is a number
Length	**the product code is valid if** Length(the product code) > 8	the product code: "123456789"	the product code is valid = true	Find the length of a text string
Concatenation	**the screen heading variable for the person = the concatenation of the person's**	the person's first name: William	the screen heading	Combine multiple text strings into a

Function	Example rule	Inputs	Outputs	Further information
	first name & ", " & the person's age & ", " & the person's occupation	the person's age: 20 the person's occupation: Student	variable for the person = "William, 20, Student"	single text variable
Substring	**customer reference = Substring(customer name, 4, 4)**	customer name: "maryjane"	customer reference = "jane"	Extract part of a text string
Text	**the customer's age text = Text(the customer's age)**	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	**the child may apply for a scholarship if** For(the child's school, the school has a scholarship program) **the child does not have to go to school if** in the case of the	the child's school: St Mary's; the school has a scholarship program: false the child's	the child may apply for a scholarship = false the child does not have to go to school = true	Refer to entities connected by a to-one relationship

Function	Declarations	Example rule	Inputs	Outputs	Further information
	Relationship text: the child's school	child's school, the school is closed for a pupil free day **the child's school name = the school name, in the case of the child's school**	school: St Joseph's; the school is closed for a pupil free day: true the child's school: St Clare's Public School; the school name: St Clare's Public School	the child's school name = St Clare's Public School	
ForScope	Source entity: the person Target entity: the car Relationship type: One-to-one Relationship text: the person's car	**the person has a reliable car if** in the case of the person's car the number of times the	the car: NSW001; the number of times the car has broken down: 4	the person has a reliable car = false	Extend the For, For All and Exists functions

Function	Declarations	Example rule	Inputs	Outputs	Further information
		car has broken down = 0			
ForScope (Alias)	Source entity: the person Target entity: the person Relationship type: One-to-one Relationship text: the person's spouse	**the person has the highest taxable income if** ForScope(the person’s spouse, the spouse) the person’s income	the person: Fran; the person’s income: 500 the person: Seb; the person’s income: 250	the person has the highest taxable income=true (Fran) the person has the highest taxable income=false (Seb)	Remove ambiguity when reasoning about more than one instance of the same entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
		> the spouse’s income			
ForAll	Source entity: Global Target entity: the child Relationship type: One-to-many Relationship text: the children	**the playground is empty if** ForAll(the children, the child is at home) **the playground is empty if** ForAll(the children, the child's location = "home") **the playground is empty if** each of the children is at home	the child: Sally; the child is at home: true; the child's location: home the child: Molly; the child is at home: true; the child's location: home the child: Elizabeth; the child is at home: false; the child's location: play-	the playground is empty = false	Check that a condition returns true for every instance of an entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
			ground		
ForAllScope	Source entity: the person Target entity: the cat Relationship type: One-to-many Relationship text: the person's cats	**the person is happy if** for all of the person's cats the cat is happy	the cat: Tiger; the cat is happy: false the cat: Kit; the cat is happy: true the cat: Patch; the cat is happy: true	the person is happy = false	Extend the For, For All and Exists functions
ForAllScope (Alias)	Source entity: the person Target entity: the person Relationship type: One-to-many Relationship text: the person's dependents	**the person has one large party if** for all of the person's dependents (the dependent) the person's birthday = the d-	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

Function	Declarations	Example rule	Inputs	Outputs	Further information
		ependent's birthday			
Exists	Source entity: Global Target entity: the child Relationship type: One-to-many Relationship text: the children	**the playground has good equipment if** Exists(the children, the child is happy) **the playground has good equipment if** at least one of the children is happy	the child: Isabelle ; the child is happy: false the child: Xavier; the child is happy: true the child: Phoebe; the child is happy: false the child: Rachel; the child is happy: false	the playground has good equipment = true	Check that a condition returns true for at least one instance of an entity
ExistsScope	Source entity: the plan Target entity: the product Relationship	**the plan has incompatible products if** ExistsScope(the plan's products) the	the plan: Plan 1; the plan's network: Vodafone the plan: Plan 2; the plan's network:	the plan has incompatible products = true	Extend the For, For All and Exists functions

Function	Declarations	Example rule	Inputs	Outputs	Further information
	type: Many-to-many Relationship text: the plan's products	plan's network <-> the product's network	Telstra the product: Product 1; the product's network: Optus the product: Product 2; the product's network: Vodafone		
ExistsScope (Alias)	Source entity: Global Target entity: the child Relationship type: One-	**the child is a twin if** ExistsScope(the children, the other child) th	the child: Kenneth; the child’s date of birth: 2007-10-15; the child's mother:	the child is a twin = true (Kenneth) the child is a twin = true (Benny) the child is	Remove ambiguity when reasoning about more than one instance of the same

Function	Declarations	Example rule	Inputs	Outputs	Further information
	to-many Relationship text: the children	e child’s date of birth = the other child’s date of birt	Samantha Jane Smith the child: Benny; the child’s date of birth: 2007-10-15; the child's mother: Samantha Jane Smith the child: Jenny; the child’s date of birth: 2006-01-02; the child's mother: Samantha Jane Smith	a twin = false (Jenny)	entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
		and the child's mother <> the other child's mother			
IsMemberOf	Source entity: the	**the child is a member of the person's household if**	the child: Sam; the	the child (Sam) is a	Use relationship

Function	Declarations	Example rule	Inputs	Outputs	Further information
	person Target entity: the child Relationship type: Many-to-many Relationship text: the person's household	the child's address = the person's address	child's address: 15 Melbourne Avenue Canberra the person: James; the person's address: 21 Sydney Avenue Canberra	member of the person's (James) household = false	membership as a rule input Infer membership of a relationship
IsNotMemberOf	Source entity: the person Target entity: the bird Relationship type: One-to-many Relationship text: the person's hated birds Reverse relationship text: the bird's owner	**the bird is happy if** in the case of the bird's owner IsNotMemberOf(the cat, the person's hat	the bird: Chirpy; the bird is a member of the person's hated birds: true	the bird (Chirpy) is happy = false	Use relationship membership as a rule input

Function	Declarations	Example rule	Inputs	Outputs	Further information
		ed birds)			
InferInstance	Source entity: the employee Target entity: the location Relationship type: Many-to-one Relationship text: the location in which the employee works Reverse relationship text: the employees at the location	**the location in which the employee works (the employee's local office) exists** **InferInstance(the location in which the employee works, the employee's local office)**	the employee: Gordon; the employee's local office: "London" the employee: Britney; the employee's local office: "London" the employee: Dominique; the employee's local office: "Paris"	new inferred location: London; the location = "London", the 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	Infer existence of entities to satisfy the relationship

Function	Declarations	Example rule	Inputs	Outputs	Further information
				location (London) = false new inferred location: Paris; the location = "Paris", the employee (Gordon) is a member of the 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	**the number of children that the claimant has = InstanceCount(the**	the child: Anthony	the number of chil-	Count the number of

Function	Declarations	Example rule	Inputs	Outputs	Further information
	claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	**claimant's children)**	the child: Peter the child: Rebecca the child: Fiona	dren that the claimant has = 4	instances of an entity
InstanceCountIf	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	**the number of school students that the claimant has = InstanceCountIf (the claimant's children, the child is a school student)**	the child: Anthony; the child is a school student: false the child: Peter; the child is a school student: false the child: Rebecca; the child is a school student: true	the number of school students that the claimant has = 1	Count the number of instances of an entity for which a particular attribute is true
InstanceMaximum	Source entity: the claimant Target entity: the child Relationship	**the highest bank balance for a child of the claimant = InstanceMaximum(the claimant's children, the child's bank balance)**	the child: Max; the child's bank balance: $50 the child: Sophie; the child's	the highest bank balance for a child of the claimant = $175	Get the highest/most recent value of an entity-level variable

Function	Declarations	Example rule	Inputs	Outputs	Further information
	type: One-to-many Relationship text: the claimant's children		bank balance: $175 the child: Katie; the child's bank balance: $120		
InstanceMaximumIf	Source entity: the company Target entity: the employee Relationship type: One-to-many Relationship text: the company's employees	**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)**	the employee: David; the employee's date of employment: 01/01/2006; the employee is a permanent employee: true the employee: Shaun; the employee's date of employment: 24/08/2006; the employee is a permanent employee: false the	the most recent date of employment of a permanent employee by the company = 2006-05-15	Get the highest/most recent value of an entity-level variable for which a particular attribute is true

Function	Declarations	Example rule	Inputs	Outputs	Further information
			employee: Anita; the employee's date of employment: 15/05/2006; the employee is a permanent employee: true		
InstanceMinimum	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	**the lightest weight for a child of the claimant = InstanceMinimum(the claimant's children, the child's weight in kilograms)**	the child: 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 lightest weight for a child of the claimant = 15	Get the lowest/least recent value of an entity-level variable
InstanceMinimumIf	Source entity: the claimant Target entity: the child Relationship	**the youngest of the claimant's female children = InstanceMinimumIf(the claimant's children, the child's age, the child is female)**	the child: Sam; the child's age: 3; the child is female: false the child:	the youngest of the claimant's female children = 4	Get the lowest/least recent value of an entity-level variable for

Function	Declar-ations	Example rule	Inputs	Outputs	Further inform-ation
	type: One-to-many Relationship text: the claimant's children		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 child's age: 8; the child is female: true		which a particular attribute is true
InstanceSum	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	**the total Child Care Benefit payable to the claimant = InstanceSum(the claimant's children, the Child Care Benefit amount for the child)**	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	the total Child Care Benefit payable to the claimant = $900	Add up numerical values gathered from each instance of an entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
			amount for the child: $150		
InstanceSumIf	Source entity: the claimant Target entity: the child Relationship type: One-to-many Relationship text: the claimant's children	**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)**	the child: Sally; the annual school fees for the child: $18000; the child attends a boarding school: true the child: James; the annual school fees for the child: $15000; the child attends a boarding school: true the child: Bob; the annual school fees for the child: $10000; the child attends a boarding school: false	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
InstanceValueIf	Source entity: Global Target entity: the child Relationship type: One-to-many Relationship text: the children	**the name of the oldest child = InstanceValueIf(the children, the child's name, the child's age = the age of the oldest child)**	the age of the oldest child = 8 the child: Sam; the child's age: 3 the child: Alex; the child's age: 4 the child: Sharon; the child's age: 6 the child: Paris; the child's age: 8	the name of the oldest child = Paris	Get a value from a unique entity instance
InstanceEquals	Source entity: the product Target entity: the product Relationship type: Many-to-many Relationship text: the products	**the product is a duplicate if** ExistsScope(the products, the other product) the product's code = the other product's code and InstanceEquals(the product, the other product)	the product: Product A; the product's code: TD2010 the product: Product B; the product's code: SM2031 the product: Product A; the product's code: TD2010	the product (Product A) is a duplicate = true the product (Product B) is a duplicate = false	Compare instances of the same entity

Function	Declarations	Example rule	Inputs	Outputs	Further information
InstanceNotEquals	Source entity: the employee Target entity: the employee Relationship type: Many-to-many Relationship text: the 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 employee: Harry; the employee's ID: RN6710 the employee: Will; the employee's ID: RN5812 the employee: Kate; the employee's ID: RN5812	the employee (Harry) has a conflicting ID = false the employee (Will) has a conflicting ID = true the employee (Kate) has a conflicting ID = true	Compare instances of the same entity

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	**the client's distinct address count = IntervalCountDistinct (2005-07-01,2006-07-01,the client's address)**	Calculate the number of distinct values for a variable in a time

Function	Example	Further information
		period
IntervalCountDistinctIf	the client's distinct address count = IntervalCountDistinctIf (2000-01-01,2007-01-01,the client's address,the client is aged over 18)	Calculate the number of distinct values for a variable in a time period only when a condition is true
IntervalDailySum	the amount of benefit payable for the assessment period = IntervalDailySum(2006-07-05,2006-08-01,the daily rate of benefit)	Calculate the sum of a variable in a time period
IntervalDailySumIf	the total amount spent on weekends in December = IntervalDailySumIf(2006-12-01,2007-01-01,the daily amount spent,the day is a weekend)	Calculate the sum of a variable in a time period only when a condition is true
IntervalMaximum	the maximum rate of benefit during the assessment period = IntervalMaximum(2006-07-05,2006-08-01,the daily rate of benefit)	Find the maximum amount in a period
IntervalMaximumIf	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)	Find the maximum amount in a period when a boolean attribute is true
IntervalMinimum	the minimum rate of benefit during the assessment period = IntervalMinimum(2006-07-05,2006-08-01,the daily rate of benefit)	Find the minimum amount in a period
IntervalMinimumIf	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)	Find the minimum amount in a period when a boolean attribute is true
IntervalWeightedAverage	the average number of children in care = Inter-	Calculate the

Function	Example	Further information
	valWeightedAverage(2007-01-22,2007-01-29,the number of children in care)	average value of a variable in a time period
IntervalWeightedAverageIf	**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)**	Calculate the average value of a variable in a time period when a condition is true
IntervalAlways	**the client was in jail at all times during the assessment period if** IntervalAlways(2006-07-10,2006-07-21,the client was in jail)	Check if a condition is true at all times in the time period
IntervalAtLeastDays	**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)	Check if a condition is true for at least the specified number of days in the time period
IntervalConsecutiveDays	**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)	Check if a condition is true for at least the specified number of consecutive days in the time period
IntervalSometimes	**the client has been in Australia if** IntervalSometimes(2007-01-08,2007-01-23,the client was in Australia)	Check if a condition is ever true in the time period
ValueAt	**the rate of benefit payable on the date of claim = ValueAt (the date of claim,the rate of benefit)**	Determine a rule attribute on a given date

Function	Example	Further information
WhenLast	**the date the customer's bank balance was last over $100 = WhenLast(the current date,the customer's bank balance > 100)**	Find the closest date when an attribute was true
WhenNext	**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)**	Find the closest date when an attribute was true
Latest	**the amount of benefit paid to the applicant since 1/7/2007 = IntervalDailySum(2007-07-01,Latest(),the amount of the monthly payment)**	Get a date value equivalent to the latest possible date
Earliest	**the amount of benefit paid to the applicant up until 1/7/2007 = IntervalDailySum(Earliest(),2007-07-01,the amount of the monthly payment)**	Get a date value equivalent to the earliest possible date
TemporalIsWeekday	**the applicant receives money if** TemporalIsWeekday(2006-07-01,2006-07-15)	Calculate the weekdays in a given time period
TemporalOncePerMonth	**the applicant receives an allowance if** TemporalOncePerMonth(2006-07-01,2006-08-31,15)	Calculate a specific day in a month for a given time period
TemporalDaysSince	**the number of days since it has rained = TemporalDaysSince (the date of the most recent rainfall,the current date)**	Calculate the number of days since a given date
TemporalWeeksSince	**the number of weeks in the assessment period = TemporalWeeksSince(2007-03-12,2007-04-11)**	Calculate the number of weeks since a given date
TemporalMonthsSince	**the number of months the mobile phone contract has been**	Calculate the

Function	Example	Further information
	in effect = TemporalMonthsSince(the start date of the mobile phone contract,the current date)	number of months since a given date
TemporalYearsSince	**the child's age = TemporalYearsSince(the child's date of birth,the child’s fifth birthday)**	Calculate the number of years since a given date
TemporalAlwaysDays	**the employee has been at work for the last 4 days if** TemporalAlwaysDays(4,the employee was working)	Check if a condition is true for all of a specified number of preceding days
TemporalConsecutiveDays	**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)	Check if a condition is true for at least the specified number of consecutive preceding days
TemporalSometimesDays	**the customer's bank balance has exceeded $100 in the last 4 days if** TemporalSometimesDays(4,the customer's bank balance exceeds $100)	Check if a condition is ever true within a specified number of preceding days
TemporalAfter	**the July 2005 rate changes apply if** TemporalAfter(2005-06-30)	Check if a condition is true after a given date and false on and before
TemporalBefore	**the pre-2007 Ministerial Determination is in force if** TemporalBefore(2007-01-01)	Check if a condition is true before a given date and false on and afterwards

Function	Example	Further information
TemporalOn	**the New Millennium Promotion is available to customers if** TemporalOn(2000-01-01)	Check if a condition is true on a given date and false before and afterwards
TemporalOnOrAfter	**the 2007 Ministerial Determination is in force if** TemporalOnOrAfter(2007-01-01)	Check if a condition is true on or after a given date and false before
TemporalOnOrBefore	**the pre-Christmas price list applies if** TemporalOnOrBefore(2007-12-24)	Check if a condition is true on and before a given date and false afterwards
TemporalFromStartDate	**the person's most recent employer = TemporalFromStartDate(the person's jobs, the job's start date, the job's employer)**	Get a temporal attribute from entity instances with values from the start date
TemporalFromEndDate	**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)**	Get a temporal attribute from entity instances with values up until the end date
TemporalFromRange	**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)**	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	**the applicant is eligible for the benefit if** it is certain whether or not the applicant is entitled to a payment or the applicant's eligibility status is certain
uncertain	**the outcome is unclear if** it is uncertain whether or not the means have been achieved or the status of the investigation is uncertain
known	**the interview has been completed if** it is known whether or not the applicant is eligible for a payment or the applicant's rate of benefit is known
unknown	**the generic heading should be shown if** it is unknown whether or not the applicant is eligible or the applicant's rate of entitlement in unknown
currently known	**the income details are available if** the applicant's income is currently known

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

the team's points from round 1	
0	the team's points from round 1 (as recorded by the team) is unknown
the team's points from round 1 (as recorded by the team)	**otherwise**

the team's points from round 2	
0	the team's points from round 2 (as recorded by the team) is unknown
the team's points from round 2 (as recorded by the team)	**otherwise**

the team's points from round 3	
0	the team's points from round 3 (as recorded by the team) is unknown
the team's points from round 3 (as recorded by the team)	**otherwise**

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 of rules. Each translation document also has an associated XGEN file. XGEN files are in XML format.	\Rules or \Translations
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	\output

File type	File extension	Description	Location in Development folder
		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.	
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 (zipped) folder	.zip	Building a project in Oracle Policy Modeling will 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. Additionally, any files in the include folder will be added to the rulebase zip. See Include extra files in the build.	\output

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
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

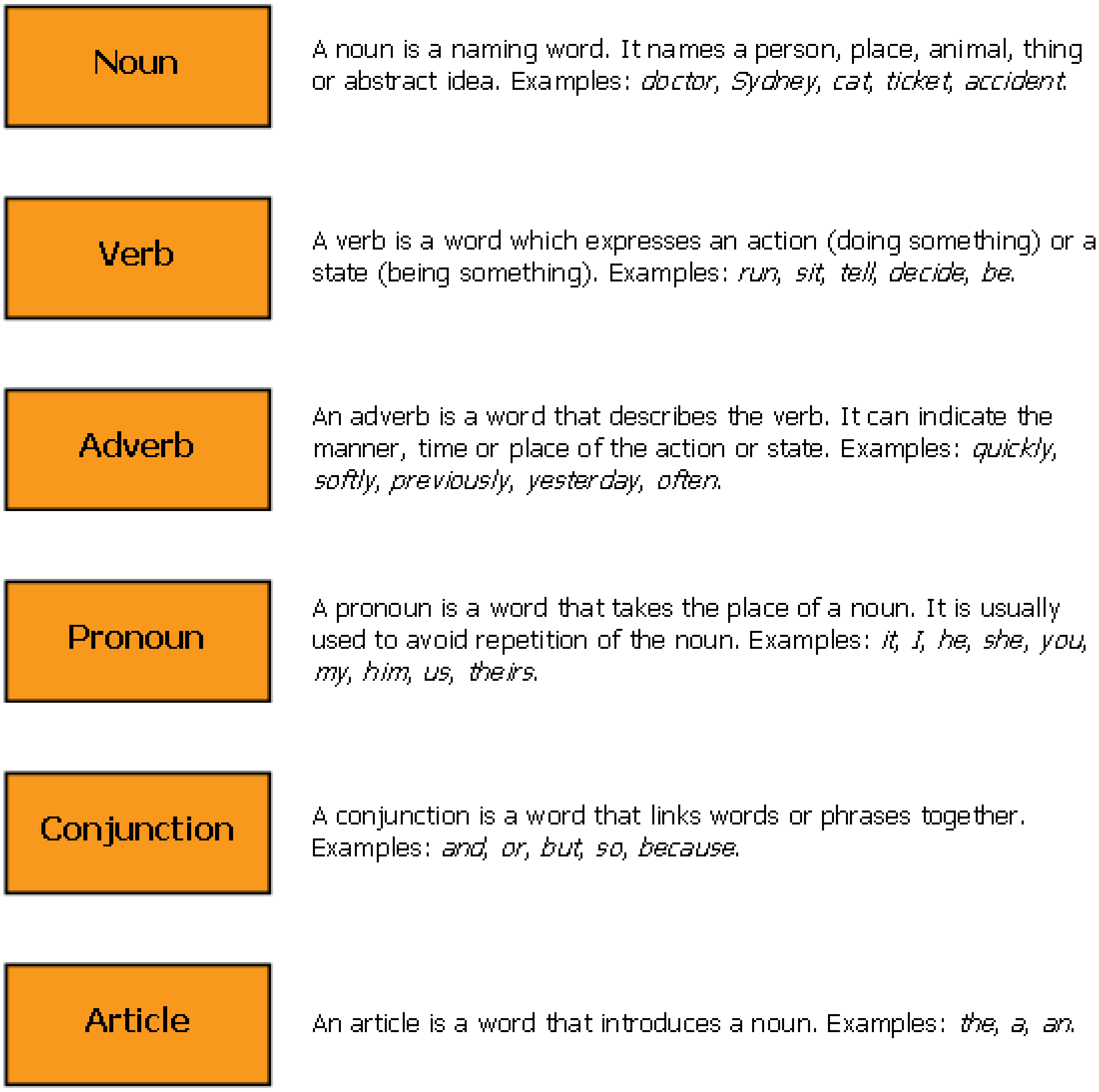
P	Q	P OR Q
UNCERTAIN	UNCERTAIN	UNCERTAIN

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

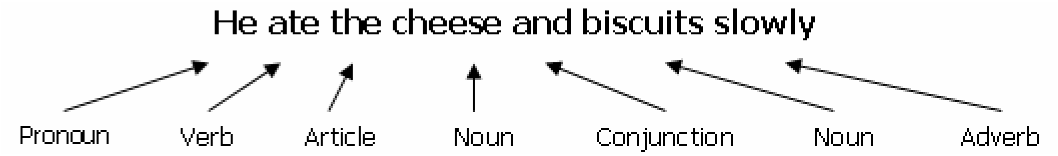
Parts of speech

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

To display	BI Publisher format	Example	Output	Notes
				modification.
Attribute value (unformatted)	<?attribute_id/-value?>	<?assessment_date/value?>	2011-06-03	BI Publisher provides a range of functions for working with unformatted 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 comparison operators can be used here.
Text when a	<?if:number(attribute_id/-	<?if:number(total_reim-	This format/example

To display	BI Publisher format	Example	Notes
currency attribute has a particular value	value)<unformatted attribute value?>Display text<?end if?>	bursement/value)<9?> You have been reimbursed less than the full amount.<?end if?>	uses the less than operator (<) but any of the comparison operators 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 comparison operators 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 which case, use the formatting described above.
Attribute value formatted a certain way when a formatted attribute has a particular value	<?choose:?><?when:attribute_id_value>'formatted attribute value'?><?attribute_id_value?><?end when?><?otherwise:?> <?attribute_id_value?> <?end otherwise?><?end choose?>	<?choose:?><?when:total_sweets_value>'4'?>**<?total_sweets_value?>** <?end when?><?otherwise:?><?total_sweets_value?><?end otherwise?><?end choose?>	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. This can also be achieved by implementing conditional formatting. See the BI Publisher Users Guide for more inform-

To display	BI Publisher format	Example	Notes
			ation.

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	<?for-each:entity_id?> <entity_level_attribute_element> <entity_level_attribute_element> <entity_level_attribute_element> ... <?end for-each?>	<?for-each:child?> <?child_name_text?> <?child_rating_overall_text?> <?child_rating_overall/@question?> <?child_rating_overall_value?> <?end for-each?>	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	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 <?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).
Entity-level attributes grouped by attribute	<?for-each:entity_id?><entity_level_attribute_element><?end for-each?> <?for-each:entity_id?><entity_level_attribute_element><?end for-each?> <?for-each:entity_id?><entity_level_attribute_element><?end for-each?> ...	<?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?>	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	
Entity-level attributes sorted	<?for-each:entity_id?> <?sort:entity_name_id_	<?for-	Courtney: The number of	This format can be used in

To display	BI Publisher format	Example	Output	Notes
alphabetically by entity name	value;'ascending';data-type='text'?> <entity_level_attribute_element> <entity_level_attribute_element> <entity_level_attribute_element> ... <?end for-each?>	each:child?><?sort:child_name_value;'ascending';data-type='text'?> <?child_name_value?>: <?child_servings_sweets_text?> <?child_servings_fruit_text?> <?child_servings_dairy_text?> <?end for-each?>	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.	native Microsoft Word tables too. That is, the first cell in the row needs to start with the <?for-each:entity_id?><?sort:entity_name_id_value;'ascending';data-type='text'?> fields, 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).

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	`<?template@inlines:decision-report?>`
	`<?if@inlines:"attribute-node"?>`
	`<fo:list-block start-indent="{count(ancestor::attribute-`
		n/a	This field only needs to appear in a template document once.

Field	BI Publisher code	Example	Notes

	`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?>`		

call decision report template	`<?for-each:/global-instance/attribute_id/decision-report/*?><?call-template:decision-report?><?end for-each?>`	`<?for-each:/global-instance/improvements/decision-report/*?><?call-template:decision-report?><?end for-each?>`	

Download BI Publisher

To download and install BI Publisher for use with Oracle Policy Automation (OPA):

- Go to the Oracle BI Publisher Downloads page on the Oracle Technology Network.
- Scroll until you can see the download links for the relevant version of BI Publisher. Note: The version of BI Publisher currently supported for use with OPA is 11.1.1.9. Ensure the BI Publisher Desktop package you select for download has a version number of 11.1.1.9 or later.
- Download the relevant **BI Publisher Desktop** package for your version of Microsoft Office (32 bit or 64 bit).

If you experience problems installing or using BI Publisher, try one of the following:

- The BI Publisher forum, or
- Troubleshooting guide for using BI Publisher with Policy Modeling

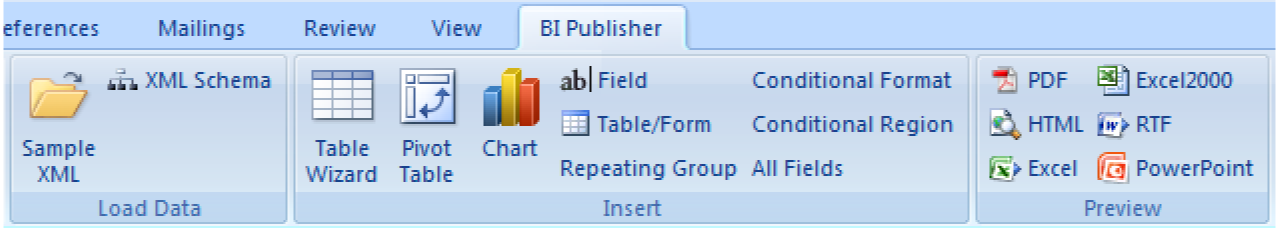
See also:

- Overview: The process of creating an interview document
- Create, update or delete an interview document
- Develop a template for an interview document
- Add a document link to the summary screen
- BI Publisher code for Policy Modeling

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.

BI Publisher ribbon not visible in Microsoft Word



If you cannot see the BI Publisher Ribbon in your Microsoft Word document after you open it from Policy Modeling, it means you do not yet have the BI Publisher Add-in for Microsoft Word installed. For information on how to do this, see Download BI Publisher.

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
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.

Shortcut Key	Function/Navigation
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
F4	Level 4 style
F5	Level 5 style

Shortcut key	Style/Function
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

Shortcut key	Style/Function
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

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 predefined 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
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

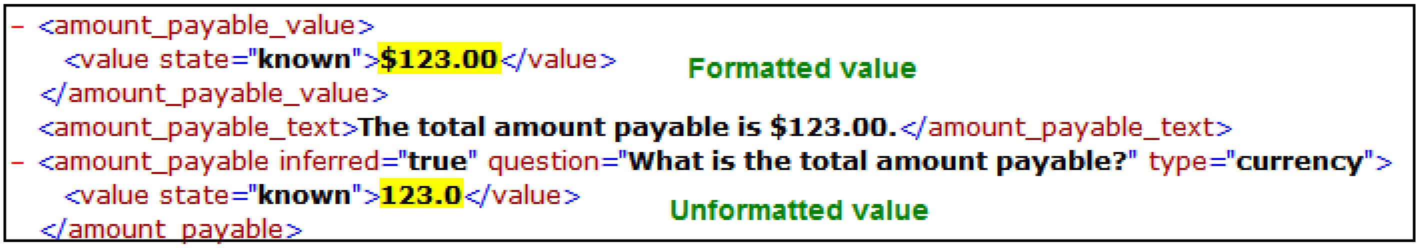
Variable type	Data format specified by rulebase region	Example
Date and time	Region set to Japan, which includes the date format yy/MM/dd and time format hh:mm:ss NOTES: If the regional format settings specify an output format for datetimes, all datetimes will be displayed with that format regardless of whether the 'Display seconds' option was ticked for that variable in the Attribute Editor. Also , the seconds component *may* 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>
  </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>
  </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
Build	Provides a means of building a rulebase from an Oracle Policy Modeling project using the command line
Regression tester	Provides a means of executing a rulebase project's text scripts using the command line
Batch processor	Provides a means of processing a large number of cases in batch using the command line