

# ORACLE POLICY AUTOMATION

## BEST PRACTICE GUIDE FOR RULE DEVELOPERS

*Author: Jasmine Lee*

*Last Updated: 19 March 2012*

## ABOUT THE AUTHOR

Jasmine Lee is an expert in advanced Oracle Policy Automation rule design and implementation topics. She has been working with the Oracle Policy Automation product line for 10 years, and has experience designing Oracle Policy Automation policy models for customers around the world. Jasmine currently works in Oracle's Industries Business Unit as a solution specialist. She has lived and worked in Australia, Europe and the United States and is currently based in Washington DC. When not designing rulebases, Jasmine can be found speedsolving the Rubik's Cube and playing with her cat.

## DOCUMENT PROPERTIES

<b>Author</b>	Jasmine Lee
---------------	-------------

## VERSION HISTORY

Date	Name	Comments
8 Nov 2011	Jasmine Lee	Released for internal review
15 Nov 2011	Jasmine Lee	Updates following feedback from internal review
19 Mar 2012	Jasmine Lee	Updated cover page, links to training courses and Useful Links section

## ACRONYMS AND ABBREVIATIONS

<b>OPA</b>	Oracle Policy Automation
<b>OPM</b>	Oracle Policy Modeling
<b>OPA Help</b>	Oracle Policy Automation Developer Help
<b>OPM Help</b>	Oracle Policy Modeling Help
<b>OU</b>	Oracle University

## DEFINITIONS

<b>Boolean attribute</b>	An attribute whose value will typically be True or False. In OPA, Booleans can also be Uncertain or Unknown.
<b>Currency variable</b>	A variable attribute with a data type of Currency.
<b>Date variable</b>	A variable attribute with a data type of Date.
<b>Design time</b>	Design time refers to when the rules are being authored. This is distinct from 'runtime' which is when the rules are being executed or 'run'.
<b>End user</b>	The term 'end user' is used in this document to describe the people who use the end result of the implementation. For example, customer service agents, call centre staff, or the general public.
<b>Grouping operator</b>	Used to make logic unambiguous in rules with premises connected by a mixture of 'and' and 'or'. The grouping operators are: all, both, any, either.
<b>Negation</b>	The negative form of a sentence parse.
<b>Number variable</b>	A variable attribute with a data type of Number.
<b>Procedural logic, rules and attributes</b>	Logic, rules and attributes whose purpose is to control interview flow and data collection.
<b>Rule author</b>	A person who implements the non-programming aspects of an OPA project such as the rules and screen definitions.
<b>Runtime</b>	A generic term to describe when the rules are being executed or 'run'. The term 'runtime' by itself does not necessary indicate how the rules are being run. For example, the rules could be run interactively in Web Determinations or a custom front-end application, or run as a batch process without a user interface.
<b>Source material</b>	The content from which the rules are being built. For public sector implementations, this will usually be a mixture of legislation, regulations and policy.
<b>Substantive logic, rules and attributes</b>	Logic, rules and attributes whose purpose is to determine the outcomes for the rulebase goals based on legislation, regulations or other source material.
<b>Text variable</b>	A variable attribute with a data type of Text.
<b>Variable attribute</b>	An attribute whose value is varied as opposed to whose value is Yes/No or True/False. A variable attribute will have a data type of Currency, Date, Number, Text, DateTime or TimeOfDay.
<b>Visibility logic, rules and attributes</b>	Logic, rules and attributes whose purpose is to control whether screen controls (attributes, labels, headings) are displayed or hidden from a summary screen or question screens.

**CONTENTS**

**About the Author..... 2**

**Document Properties..... 3**

    Version History ..... 3

    Acronyms and Abbreviations ..... 3

**Definitions ..... 4**

**Chapter 1. Introduction..... 8**

    1.1 Oracle Policy Modeling Help ..... 8

    1.2 Version of Oracle Policy Modeling..... 8

**Chapter 2. Rules..... 9**

    2.1 Different types of rules ..... 9

    2.2 Separation of different types of logic ..... 9

        2.2.1 Never mix procedural logic into substantive logic..... 10

        2.2.2 Never mix visibility logic into substantive logic..... 11

        2.2.3 Never use substantive logic rules as visibility attributes ..... 11

    2.3 Splitting logic out into separate rules ..... 12

    2.4 Nesting of logic..... 13

    2.5 Negations as conclusions ..... 13

    2.6 Variables as conclusions..... 13

    2.7 Explicit conjunctions ('and') and disjunctions ('or') ..... 14

    2.8 Rule functions..... 15

        2.8.1 Terse form vs natural language form..... 15

        2.8.2 Function names..... 15

    2.9 Tables..... 15

        2.9.1 Word Table vs Excel Table..... 15

        2.9.2 Do not conclude Booleans in Word tables..... 16

    2.10 Boolean attribute over the top of list items..... 16

        2.10.1 Variation to the Boolean attribute approach to list items..... 18

**Chapter 3. Entities..... 22**

    3.1 When to add entities and relationships..... 22

    3.2 Naming entities and entity relationships ..... 23

**Chapter 4. Attributes ..... 24**

    4.1 Attribute Text ..... 24

4.1.1	Boolean attributes.....	24
4.1.2	Variable attributes.....	25
4.1.3	Entity level attributes .....	26
4.1.4	Attribute purpose should be clear from attribute text.....	27
4.2	Attribute Parsing.....	28
4.2.1	Boolean attributes.....	28
4.2.2	Identifying and fixing parsing issues with Boolean attributes.....	29
4.2.3	Special exception: can not .....	31
4.2.4	Variable attributes.....	32
4.3	Substitution.....	32
4.3.1	Substitution in attribute text.....	33
4.3.2	Substitution in screen headings and labels.....	34
4.3.3	Substitution in attribute text with second person sentence generation .....	35
4.3.4	Gender pronoun substitution.....	35
<b>Chapter 5.</b>	<b>Screens .....</b>	<b>37</b>
5.1	Usability Considerations.....	37
5.2	Appropriate use of Free Form text .....	37
5.2.1	General details screen .....	38
5.2.2	Entity Collect screen .....	39
5.2.3	Collecting a series of similar things.....	41
5.3	Configuring the Add/Remove Instance button text on Entity Collect screens .....	43
<b>Chapter 6.</b>	<b>Interview Flow .....</b>	<b>44</b>
6.1	Screen Order .....	44
6.2	Screen Flow .....	44
6.2.1	Displaying screens with known data.....	45
6.2.2	Displaying information-only screens .....	45
6.2.3	Ability to re-run interview in same session.....	46
6.3	Screen attributes .....	46
<b>Chapter 7.</b>	<b>Organizing Content .....</b>	<b>47</b>
7.1	Project folders and files .....	47
7.1.1	Create logical subfolders in which to store different rule documents.....	47
7.1.2	Adding the Custom verbs file to the project .....	47
7.2	Rule Documents .....	48
7.2.1	Heading levels .....	48

7.2.2	Table of Contents .....	48
7.2.3	Spacing between rules.....	49
7.2.4	Adjusting rule table column width .....	49
7.2.5	Soft enters in calculation rules .....	51
7.2.6	Adding comments to rule documents .....	52
7.2.7	Pages per rule document .....	52
7.3	Attributes .....	53
7.3.1	Tidy up auto-generated attribute IDs .....	53
<b>Useful Links</b>	.....	<b>56</b>

## CHAPTER 1. INTRODUCTION

The purpose of this document is to provide some Oracle Policy Modeling (OPM) best practice guidelines for rule authors working on Oracle Policy Automation (OPA) implementations.

The guidelines in this document are just that – guidelines. They are not absolute rules (with a few exceptions). Every rulebase is a case-by-case situation where the rule author will need to use their experience and good judgment, whether it is in designing the overall rulebase structure, deciding how to model a particular piece of logic, or choosing appropriate attribute text.

This document is not intended for training someone who is new to Oracle Policy Automation. At a minimum, anyone who is working on an OPA implementation as a rule author should complete the following Oracle University courses:

- **Oracle Policy Modeling Essentials – Part 1**  
[http://education.oracle.com/pls/web\\_prod-plq-dad/db\\_pages.getCourseDesc?dc=D72146GC10](http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D72146GC10)
- **Oracle Policy Modeling Essentials – Part 2**  
[http://education.oracle.com/pls/web\\_prod-plq-dad/db\\_pages.getCourseDesc?dc=D72144GC10](http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D72144GC10)

This document assumes the rule author is familiar and confident with all the content of these courses. There are also some sections of this document which assume OPA knowledge beyond that covered in the Oracle University courses.

This document does not provide descriptions of OPM functionality, nor is it intended to provide ‘how to’ advice for using OPM features and functionality. For descriptions of OPM functionality and information about how to use the features and functionality of OPM, please refer to the OPM Help.

### 1.1 ORACLE POLICY MODELING HELP

The Oracle Policy Modeling Help is installed locally when Oracle Policy Modeling is installed. It is accessible via the Help menu from within OPM, and via the Program menu on the machine onto which it was installed (*Start | All Programs | Oracle Policy Modeling | Oracle Policy Modeling 10 Help*)

In addition, the OPM Help and OPA Help are available online on the OPA OTN site, in the Documentation section:  
<http://www.oracle.com/technetwork/apps-tech/policy-automation/documentation/index.html>

This document has many references to particular OPM Help articles. The URL links to specific Help articles are correct at the time of writing. However, as URLs may change over time, the name of the article is included so you can look it up in your local copy of the OPM Help.

### 1.2 VERSION OF ORACLE POLICY MODELING

The guidance in this document is not specific to a particular version of Oracle Policy Modeling. It is general guidance which is relevant to any version of OPM from recent years.

## CHAPTER 2. RULES

### 2.1 DIFFERENT TYPES OF RULES

Some of the guidance in this chapter refers to different types of rules: substantive rules vs procedural rules vs visibility rules. It is important to understand the difference between these different types. See examples below.

#### Substantive rules

Substantive rules are those whose purpose is to determine the outcome for the rulebase goals based on the source material, e.g.

**the person is eligible for Benefit ABC if**  
the person satisfies the residency requirements and  
the person satisfies the income requirements and  
the person satisfies the asset test

#### Procedural rules

Procedural rules are those whose purpose is to control interview flow and data collection, e.g.

**the assessment for Benefit ABC is complete if**  
the person's full name is known and  
the person's demographic details have been collected and  
it is known whether or not the person is eligible for Benefit ABC

#### Visibility rules

Visibility rules are those whose purpose is to control whether screen controls (attributes, labels, headings) are displayed or hidden from a summary screen or question screen, e.g.

**the eligibility goal for Benefit ABC should not be displayed on the summary screen if**  
the person is not eligible for Benefit ABC

### 2.2 SEPARATION OF DIFFERENT TYPES OF LOGIC

In Chapter 1 it was stated that this document contains guidelines rather than strict rules. This is certainly the case for the majority of the content. However, section 2.2 discusses some things you should never ever do:


- ✘ **Never mix procedural logic into substantive logic rules**
- ✘ **Never mix visibility logic into substantive logic rules**
- ✘ **Never use the conclusions of substantive logic rules as visibility attributes**

Each of these statements is discussed below with examples.


2.2.1 NEVER MIX PROCEDURAL LOGIC INTO SUBSTANTIVE LOGIC


Substantive logic rules should never be dependent on procedural attributes, i.e. there should never be procedural attributes used as premises in substantive logic rules.

Consider the following rule. There is an eligibility goal (“the person is eligible for Benefit ABC”) which is proven to be true or false based on collecting the person’s full name and determining if the person satisfies the income requirements.


**the person is eligible for Benefit ABC if**  
 → the person’s full name is known and  
 the person satisfies the income requirements 

Collecting the person’s full name should not be part of this rule. Determining a person’s eligibility for benefits is not dependent on the person’s actual name. There may be a procedural requirement to collect the person’s full name and other demographic data about the person, but unless the data is a determining factor in eligibility, it should not be part of the eligibility logic rule. Instead, it should be part of a higher level procedural rule. For example:


**the assessment for Benefit ABC is complete if**  
 the person’s full name is known and  
 it is known whether or not the person is eligible for Benefit ABC 


**the person is eligible for Benefit ABC if**  
 the person satisfies the income requirements 

Consider the situation where a piece of data is a determining factor. For example, imagine there is a benefit which is only available to U.S. citizens. In this case, it would be appropriate to include the citizenship requirement in the eligibility rule, e.g.

**the person is eligible for Benefit ABC if**  
 the person is a U.S. citizen and  
 the person satisfies the income requirements 

Whereas if the policy instructed to collect whether or not the person is a U.S. citizen purely for statistical purposes, then it should not be part of the eligibility logic, but rather should be part of a higher level procedural rule. For example:

**the assessment for Benefit ABC is complete if**  
 it is known whether or not the person is a U.S. citizen and  
 it is known whether or not the person is eligible for Benefit ABC 


**the person is eligible for Benefit ABC if**  
 the person satisfies the income requirements 

### 2.2.2 NEVER MIX VISIBILITY LOGIC INTO SUBSTANTIVE LOGIC

Substantive logic rules should never be dependent on visibility attributes, i.e. there should never be visibility attributes used as premises in substantive logic rules.

Consider the following rule. There is an eligibility goal (“the person is eligible for Benefit ABC”) which is proven to be true or false based on determining whether the person satisfies the income requirements and whether the income details should be displayed.

**the person is eligible for Benefit ABC if**  
 the person satisfies the residency requirements and  
 → the income details should be displayed



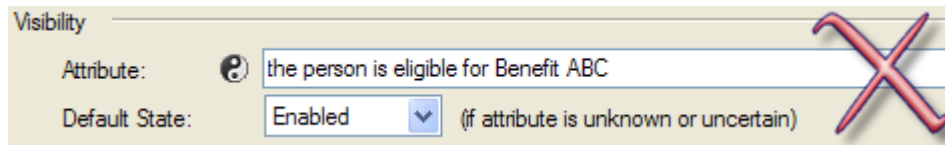
Substantive logic (i.e. “the person is eligible for Benefit ABC” in this example) should never be dependent on how the interview looks or flows at runtime. Whether or not a particular screen displays, or what it looks like when it displays, should have no bearing on whether a person is legally eligible for a benefit.

### 2.2.3 NEVER USE SUBSTANTIVE LOGIC RULES AS VISIBILITY ATTRIBUTES

Substantive logic rules should never be used *as* visibility rules, but rather should feed *into* visibility rules. For example, imagine you had the following substantive rule:

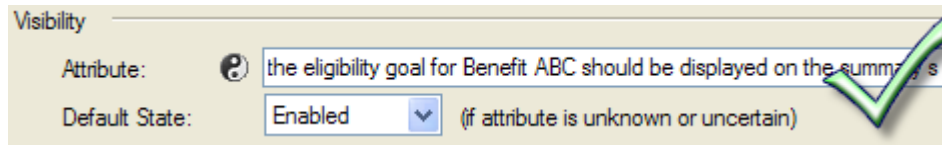
**the person is eligible for Benefit ABC if**  
 the person satisfies the residency requirements and  
 the person satisfies the income requirements

In Web Determinations, imagine that you want this eligibility goal to be displayed on the summary screen by default, but hidden if the conclusion is false. You should not use the eligibility goal attribute itself as its own visibility attribute. In fact, you should never use a substantive goal attribute as a visibility attribute, i.e. you should not do this on the summary screen:



Instead, create a visibility rule and link the eligibility goal to the visibility rule, e.g.

**the eligibility goal for Benefit ABC should not be displayed on the summary screen if**  
 the person is not eligible for Benefit ABC



Some reasons why it is good practice not to use substantive goal attributes as visibility attributes are:

- It means that modifying the substantive eligibility rules is less likely to break the visibility logic.
- Visibility logic may be dependent on multiple factors, e.g. if you needed the visibility of the goal to be dependent on whether the person is eligible and whether the person's age is greater than 18 years. It would not be possible to implement this visibility logic if the eligibility goal was used directly as the visibility attribute.

#### OPM Help article

##### ***Hide, display and disable an interview screen element***

Control the visibility of summary screen elements:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Hide\\_display\\_disable\\_interview\\_screen\\_element.htm#Control2](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Hide_display_disable_interview_screen_element.htm#Control2)

## 2.3 SPLITTING LOGIC OUT INTO SEPARATE RULES

This will always be a case-by-case assessment for the rule author, but here are some general guidelines to keep in mind:

- **Look to source material for guidance.** If the source material separates the logic into different sections, sub-sections and paragraphs, then consider doing the same in the rules.
- **Clarity.** Would separating the logic into more rules make it easier for another rule author to follow the logic? If yes, this is a good indication that the logic should be separated into multiple rules.
- **Maintenance.** Would separating the logic into more rules make it easier to maintain the rules in future? If yes, this is a good indication that the logic should be separated into multiple rules.

Do not be afraid to use a few extra lines in your rules. The shortest rule is not necessarily the easiest to read, understand, maintain and test.

It is very uncommon (but not impossible or unheard of) that it would be appropriate to have a single rule which is longer than one page.

Rule authors with a programming background (as opposed to rule authors with a legal/policy background) should be particularly mindful of splitting out logic into more rules. Those with purely programming backgrounds tend to cram more logic into a single OPA rule, whereas rule authors with a legal/policy background tend to separate out their logic into more rules. Both versions may be logically correct, but one is easier to read, understand, maintain and test, and is therefore preferable.

## 2.4 NESTING OF LOGIC

The majority of rules generally should not go deeper than Level 3. If you find yourself regularly getting to Level 4 and beyond, this is a good indication you should consider splitting out the logic into separate rules. However, there will be situations where it is quite appropriate to nest logic more deeply than this in a single rule, for example:

- Legislation and regulations. If trying to model the rules as isomorphically as possible (i.e. in the same shape and form as the original source material), then you might end up nesting more deeply.
- Procedural rules. Occasionally you may have deeply nested procedural logic which is not separable into sufficiently discrete components to make breaking out possible or useful.

## 2.5 NEGATIONS AS CONCLUSIONS

A misconception among newer rule authors is that they cannot or should not use negations as rule conclusions. While it is true that the majority of Boolean-conclusion rules will have the conclusion expressed in the positive form, it is okay to use negations as rule conclusions if required by the logic, e.g. if the source material has expressed the logic in the negative.

## 2.6 VARIABLES AS CONCLUSIONS

There are three situations where it is reasonable to use a variable as a conclusion in a Word rule:

- The rule is a conclusion-only rule, e.g.

**the threshold date = 2011-01-01**


- The rule is a calculation without conditional premises, e.g.


**the final amount = the first amount + (the second amount/the third amount)**

- The rule is a rule table, e.g.

<b>the standard monthly deduction</b>	
<b>142</b>	the number of people in the household >= 1 and the number of people in the household <= 3
<b>153</b>	the number of people in the household = 4
<b>179</b>	the number of people in the household >= 5
<b>uncertain</b>	<b>otherwise</b>

If there is conditional logic for setting the conclusion variable, you should use a rule table. You should not write rules like these:

**the applicable rate = 200 if**  
 the person owns a house 

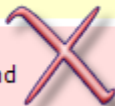
**the person's preferred location = "Hawaii" if**  
 the person loves sunny weather 

The rules above do not make sense. Regular rules have implicit alternate conclusions, so what is the opposite of 200? Is it negative 200? Is it zero? The second example is even more nonsensical. What is the logical opposite of "Hawaii"? Alternate conclusions of 'the opposite of 200' and 'the opposite of Hawaii' do not make sense, and thus you should not have rules like these.


## 2.7 EXPLICIT CONJUNCTIONS ('AND') AND DISJUNCTIONS ('OR')

Conjunctions and disjunctions should be explicit, not implicit, i.e. where there is and/or logic being used, write 'and' and 'or' into the rule. Also, avoid adding extra lines to your rules purely for the purpose of unnecessary grouping operators.


The rule below has an unnecessary grouping operator ('all').

**the person is eligible if**  
 all  
 the person satisfies the age requirement and  
 the person satisfies the income requirements and  
 the person satisfies the residency requirements 

The rule below does not have explicit conjunctions. Instead, it is relying on the presence of the extra line with the grouping operator.

**the person is eligible if**  
 all  
 the person satisfies the age requirement  
 the person satisfies the income requirements  
 the person satisfies the residency requirements 

The rule below has explicit conjunctions ('and') and it does not have an unnecessary grouping operator.

**the person is eligible if**  
 the person satisfies the age requirement and  
 the person satisfies the income requirements and  
 the person satisfies the residency requirements 

Use grouping operators when required by the logic, such as when you have conjunctions ('and') and disjunctions ('or') in the same rule. For example, in the rule below a grouping operator is required to make the logic unambiguous.

**the person is eligible if**

the person satisfies the age requirement and  
any

the person satisfies the income requirements or  
the person satisfies the residency requirements or  
the person satisfies the asset requirements



## 2.8 RULE FUNCTIONS

### 2.8.1 TERSE FORM VS NATURAL LANGUAGE FORM

Commonly used OPM rule functions have a terse form of the syntax (e.g. AddDays(<attribute>, number of days)) and a natural language form (e.g. the date X days after <attribute>):

**the expiration date = AddDays(the start date, 30)**

**the expiration date = the date 30 days after the start date**

Very complex functions, and the less commonly used functions, do not have natural language forms, e.g. IntervalConsecutiveDays, TemporalFromStartDate and ConcatenateDateTime.

There is no general best practice about using one form over the other. It is perfectly acceptable to use whichever form you prefer. It is also okay to use the terse form in some rules and the natural language form in other rules. Many rule authors find the terse form of the syntax easier to remember and allows them to more quickly see the logic in the rule, and thus use the terse form by preference.

### 2.8.2 FUNCTION NAMES

When using the terse form of functions, capitalize the first letter of each word in the function name, e.g. NextDayOfTheWeek rather than nextdayoftheweek, or InstanceMinimumIf instead of instanceminimumif. This is not technically required for the function to work, but it does make the function name, and therefore the rule, slightly easier to read.

**the age of the youngest female household member =  
instanceminimumif(the household members, the household  
member's age in years, the household member is female)**



**the age of the youngest female household member =  
InstanceMinimumIf(the household members, the household  
member's age in years, the household member is female)**



## 2.9 TABLES

### 2.9.1 WORD TABLE VS EXCEL TABLE

Here are some things to keep in mind when deciding whether to use a Word rule table or Excel rule table:

- **Simple logic or complex?** If the logic being modeled is relatively simple and can neatly be expressed in a Word table, consider Word. Many people find Word easier to work with, and so it makes sense to use it unless there is a good reason otherwise.
- **Multiple conclusions from same conditions?** One major difference between Word tables and Excel tables is that you can have multiple conclusions in a single Excel table (by having multiple conclusion columns in the one table), whereas with Word tables you can only have one conclusion per table. If the logic is such that you want to have multiple conclusions based on the same conditions then consider Excel.
- **Concise expression of logic.** If the logic can be expressed more concisely in an Excel table, then consider Excel. For example, if it would take 2 pages of Word rules to express the logic, as opposed to a concise 1/2 screen Excel table, then Excel may be more appropriate.

### 2.9.2 DO NOT CONCLUDE BOOLEANS IN WORD TABLES

Boolean attributes should not be concluded in Word tables.

<b>the person is eligible for the benefit</b>	
<b>True</b>	the person is aged 18 years or older and the person satisfies the residency requirements and either the person satisfies the income test or the person satisfies the asset test
<b>False</b>	<b>otherwise</b>

Booleans should be concluded in regular rules, e.g.

<b>the person is eligible for the benefit if</b>
the person is aged 18 years or older and the person satisfies the residency requirements and either the person satisfies the income test or the person satisfies the asset test

### 2.10 BOOLEAN ATTRIBUTE OVER THE TOP OF LIST ITEMS

For some variable attributes, most commonly text variables, the value for the attribute at runtime will be set from a pre-defined set of values, e.g. a drop-down list, radio buttons or list box. The screenshot below shows an example of a drop-down list on a Web Determinations question screen.

What is the person's immigration status? \*

- Citizen
- Citizen
- Lawful Permanent Resident
- Refugee
- Other

In this situation, a Boolean attribute should be created to sit over each list item (e.g. see below), and the Boolean should be used throughout the rulebase, rather than using premises like this: the person's immigration status = "Citizen"

**the person is a citizen if**

the person's immigration status = "Citizen"

**the person is a permanent resident if**

the person's immigration status = "Lawful Permanent Resident"

**the person is a refugee if**

the person's immigration status = "Refugee"

If the "Other" option in the drop-down list is just a catch-all for when none of the other options apply, then you may not need to create a Boolean attribute to sit over it as you may never be specifically referring to that option in the rules. In the example above, selecting "Other" will result in the Booleans for citizen, permanent resident and refugee being evaluated to false, and this may be sufficient for your logic.

The reason for using the Booleans throughout the rules is ease of future maintenance. Imagine you have used the following premise in dozens of places throughout the rulebase:

the person's immigration status = "Lawful Permanent Resident"

Now imagine that the name of this status category in the policy changed from "Lawful Permanent Resident" to "Legal Permanent Resident". If you have used the premise above in dozens of places, then you would need to make this change in dozens of places. Whereas if you have used the Boolean ("the person is a permanent resident") then you only need to change the text string in one rule:

**the person is a permanent resident if**

the person's immigration status = "Legal Permanent Resident"

Or imagine that the variable "the person's immigration status" needed to be changed to "the person's citizenship status". If you have used the variable "the person's immigration status" throughout the rules, then you need to modify all those rules. If you have used the Boolean approach, then you only have a couple of rules to modify.

A couple of additional points to keep in mind here:

- If the change to the text string or the variable is a change in the actual meaning, then you may need to edit all the rules in which the attribute is used.
- There is a 'Change Text Globally' feature which can help if the change is to the attribute text itself, however, this will not help if the change is to the text string associated with the variable.
- The advantage of the approach described here is that it minimizes the number of rules which are touched. Even if the 'Change Text Globally' functionality is used to do the actual rule change work, it is still the case that the rule has been *changed*, and anytime a rule has been changed, it should be re-tested. If you minimize the rules which are touched, then you reduce what needs to be re-tested.

The best way to approach these situations will be highly dependent on the exact logic of your source material. This section just offers some ideas and you will need to use good judgment in deciding what to do in your rulebase.

---

### 2.10.1 VARIATION TO THE BOOLEAN ATTRIBUTE APPROACH TO LIST ITEMS

There are limited circumstances where this Boolean approach is not necessarily the most sensible option, such as when the pre-defined list of values is very long and the individual list items are never (or rarely) used in Word rules. The list of U.S. States will be used for the purpose of this example, but the general principle is the same for any geography, or indeed any variable with a long list.

#### Scenario

As part of an income test in an eligibility determination, your source material requires the rules to compare the household's income against the U.S. Federal Poverty Levels. The Federal Poverty Levels include 3 sets of values: one set for Alaska, one set for Hawaii, and one set for the remaining 48 contiguous States and DC. For the income test, it does not make any difference if the person lives in Virginia or California or Florida or any of the other 48 contiguous States. Purely for income test purposes, it would be sufficient to ask the end user to select from the following items at runtime:

- What is the person's State of residence?
  - Alaska
  - Hawaii
  - 48 contiguous States or DC

However, one of the business requirements is that the interview must collect the person's State of residence. Here are a couple of ways you could approach this situation.

#### Approach 1

- Create a drop-down list which includes all the States and DC
- Create 3 Boolean attributes, one for each 'group' required by the income test logic. Prove each of those Booleans from the variable, e.g.

**the person lives in Alaska if**

the person's State of residence = "Alaska"

**the person lives in Hawaii if**

the person's State of residence = "Hawaii"

**the person lives in the 48 contiguous States or DC if**

the person's State of residence = "Alabama" or  
 the person's State of residence = "Arizona" or  
 the person's State of residence = "Arkansas" or  
 the person's State of residence = "California" or  
 etc.

- When you need to refer to the group in Word rules, use the Boolean, e.g. use "the person lives in the 48 contiguous States or DC" rather than listing out 49 individual premises.

**Approach 2**

If the 'residence' criteria was primarily going to be used in Excel tables, then here is another approach to consider.

- Set up the 'grouping' logic in an Excel table, e.g.

State	Geographic Region
Alaska	Alaska
Hawaii	Hawaii
Alabama	48 contiguous States and DC
Arizona	
Arkansas	
California	
Colorado	
Connecticut	
Delaware	
District of Columbia	
Florida	
Georgia	
Idaho	
Illinois	
Indiana	
Iowa	
Kansas	
Kentucky	
Louisiana	

- Use the grouping in other rule tables which are conditional on that logical grouping, e.g.

Geographic Region	Household Number	Threshold
48 contiguous States and DC	1	907.50
	2	1225.83
	3	1544.17
	4	1862.50
	5	2180.83
	6	2499.17
	7	2817.50
	8	3135.83
	>= 9	$3135.83 + (318 * (\text{Household Number} - 8))$
Alaska	1	1133.33
	2	1531.67
	3	1930.00
	4	2328.33
	5	2726.67
	6	3125.00
	7	3523.33
	8	3921.67
	>= 9	$3921.67 + (398 * (\text{Household Number} - 8))$
Hawaii	1	1045.00
	2	1410.83
	3	1776.67
	4	2142.50
	5	2508.33
	6	2874.17
	7	3240.00
	8	3605.83
	>= 9	$3605.83 + (365 * (\text{Household Number} - 8))$
	<i>else</i>	

- Note that you can still use the individual States if you need to in other rules, e.g.

State	Utility Allowance
Alabama	285
Alaska	760
Arizona	328
Arkansas	271
California	287
Colorado	507
Connecticut	720
Delaware	444
District of Columbia	300
Florida	317
Georgia	323
Hawaii	384
Idaho	400
Illinois	324
Indiana	433
Iowa	393
Kansas	350
Kentucky	326
Louisiana	322

## CHAPTER 3. ENTITIES

There are both pros and cons to using entities and relationships in your rulebase. The significant pro is that it allows much more complex logic to be authored in the rulebase. The con is that, compared to Global-only rulebases, rulebases with entities and relationships are more difficult to write, test and maintain. This is due to the unavoidable fact that the more complex the logic, the more complex it is to write, test and maintain.

If the logic you need to implement requires entity reasoning, then by all means add the appropriate entities and relationships. The benefit you will gain, in terms of the logic that can be written into the rulebase, will far outweigh any cons. However, if your logic does not require entity reasoning, then do not add entities just because you can. You will be making more work for yourself and others, with no benefit as a result.

When designing your entity structure, it is wise to think about future phases of the project as well, and whether future extensions to the rulebase will require entity reasoning. It may be the case that the current preliminary phase does not require entity reasoning, but the later phases will. You will save yourself some effort and re-work later on if you set up the entity structure now to accommodate the future phases.

### 3.1 WHEN TO ADD ENTITIES AND RELATIONSHIPS

An entity should be used when you have a set of 'things', and you need to collect data about each member of the set, and you need to do logic over the set. The most common example in Public Sector is a set of household members where you need to collect data about each household member, and do logic over the set of household members. For example, imagine the source material said:

- The household is eligible for the Family Benefit if the household income for the purpose of the benefit calculation is less than \$20,000.
- For adult household members (where adult = aged 18 years or older), include their entire income in the household income. For non-adult household members, only 20% of their income counts towards the household income.

To implement this logic, you would need to use entities. Specifically, you would need to create an entity called 'the household member' and a one-to-many containment relationship from 'Global' to 'the household member'.

Note that entities do not always represent people. Depending on your source material, entities may be a variety of other things, e.g. the subsidiary company, the product, the vehicle, the asset, etc. Also, OPA logic allows for entities to be nested below other entities; entities do not all have to be immediately contained by Global.

#### OPM Help article

##### *Define an entity*

[http://download.oracle.com/docs/html/E24270\\_01/Content/Data%20model/Define\\_an\\_entity.htm](http://download.oracle.com/docs/html/E24270_01/Content/Data%20model/Define_an_entity.htm)

## 3.2 NAMING ENTITIES AND ENTITY RELATIONSHIPS

The name of the entity should start with the definite article ("the") and refer to what the entity is, e.g. the household member.

The name of the entity relationship should also start with the definite article ("the") and it should be a meaningful name which describes the relationship. This requires taking into consideration what type of relationship is being described, e.g. one-to-many, many-to-one, many-to-many, inferred relationship.

It is important to choose appropriate names for the entity relationships as the entity relationship names appear throughout the project, e.g. entity quantification rules, conclusions in inferred relationship membership rules, membership statements in conditional premises in rules, and decision reports.

The OPM Help has an excellent article on choosing names for entity relationships, and includes examples for all relationship types, so refer to the Help for further advice on this topic.

### **OPM Help article**

#### ***Choose a name for an entity, relationship or attribute***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Data%20model/Choose\\_name\\_for\\_entity%20relationship\\_or\\_attribute.htm](http://download.oracle.com/docs/html/E24270_01/Content/Data%20model/Choose_name_for_entity%20relationship_or_attribute.htm)

## CHAPTER 4. ATTRIBUTES

### 4.1 ATTRIBUTE TEXT

#### 4.1.1 BOOLEAN ATTRIBUTES

The OPM Help has excellent guidance on choosing appropriate attribute text, including further detail on most of the principles below, so the OPM Help should be referred to in combination with the guidance here.

General principles for choosing attribute text:

- Statements should start with the definite article ("the"), not the indefinite article ("a", "an")
- Statements should use correct spelling, grammar and punctuation
  - The rule documents you are writing are not casual emails to friends; they are part of a customer implementation which represents the government agency or corporation who is your customer. Therefore, it is important to use correct spelling, grammar and punctuation.
- Statements should be complete grammatical sentences
- Statements should be written in the third person
  - Even if you want the question text to display in second person in Web Determinations, attributes should still be written in third person in the rules. There is a separate mechanism to switch on second person sentence generation for display in Web Determinations.
- Statements must be able to be negated
- Statements should represent a single concept
- Statements should not use contractions
- Statements should make sense without reference to another statement
- Statement should be kept simple but explicit
- Statements should indicate entity membership
- Statements should not use personal pronouns
- Statements which refer to amounts should indicate the unit of measurement
- Statements should be worded consistently where possible
  - For example, consider the following two statement structures: 'the person satisfies the XX requirements', 'the XX requirements are satisfied for the person'. Either sentence is fine, but

pick one structure and use it consistently throughout the rulebase. If you picked the first one, your rulebase might have Booleans such as "the person satisfies the income requirements" and "the person satisfies the residency requirements", whereas if you picked the second one the equivalent Booleans would be "the income requirements are satisfied for the person" and "the residency requirements are satisfied for the person".

- Statements should not use the ampersand symbol ("&")

---

#### 4.1.2 VARIABLE ATTRIBUTES

The principles above for Boolean attributes generally apply to variables as well, with the exception of principles which logically cannot apply to variables, e.g. it is not possible to negate a variable.

A good way to test in your head if your variable is worded properly is to say "what is.." at the start and see if that sounds like a properly constructed question, e.g. *What is the person's date of birth?* *What is the person's annual income?*

#### Example of a badly worded variable: "person income"

This is a very badly worded variable. The question form of this variable would be "what is person income?" This question is grammatically incorrect, and does not even make sense. There are at least three mistakes in the wording of this variable:

- The definite article ("the") has not been used. This variable should start with "the...".
- The "income" referred to is the income of the person, therefore there should be a possessive "s" attached to the person to indicate ownership of the income, e.g. the person's income.
- There is no indication as to timeframe. Does the attribute refer to an annual income, monthly income, weekly income, some other time period?

A better alternative would be something like "the person's annual income".

Depending on your logic, you may need to be more specific such as "the person's annual earned income" if you need to distinguish earned income from unearned income.

**OPM Help articles**

**Choose a name for an entity, relationship or attribute**

Choose attribute text:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Data%20model/Choose\\_name\\_for\\_entity%20relationship\\_or\\_attribute.htm#Choose3](http://download.oracle.com/docs/html/E24270_01/Content/Data%20model/Choose_name_for_entity%20relationship_or_attribute.htm#Choose3)

Basic English grammar:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Reference/Basic\\_English\\_Grammar.htm](http://download.oracle.com/docs/html/E24270_01/Content/Reference/Basic_English_Grammar.htm)

**4.1.3 ENTITY LEVEL ATTRIBUTES**

Entity level attributes (Booleans and variables) should contain the name of the entity in the text of the attribute. For example, if you have an entity "the household member" and an attribute within that entity representing the date of birth, that attribute must contain the exact text string "the household member" somewhere in the attribute text.

The entity name is often placed at the start of the entity attribute text, but it does not have to be the start, it just needs to appear somewhere in the attribute text. Note also that the full entity name has to appear in its exact form without interruption by another word. See below for examples of acceptable entity attribute text and incorrect entity attribute text.

**Entity: the household member**

Example attribute text	Comment
✓ the household member's date of birth	<b>Acceptable.</b> Attribute text contains the entity name.
✓ the household member's birthdate	<b>Acceptable.</b> Attribute text contains the entity name.
✓ the date of birth of the household member	<b>Acceptable.</b> Attribute text contains the entity name.
✓ the birthdate of the household member	<b>Acceptable.</b> Attribute text contains the entity name.
✗ the date of birth	<b>Incorrect.</b> Attribute text does not contain the entity name.
✗ the birthdate	<b>Incorrect.</b> Attribute text does not contain the entity name.
✗ the person's date of birth	<b>Incorrect.</b> Attribute text does not contain the entity name.
✗ the date of birth of the member of the household	<b>Incorrect.</b> Attribute text contains all the words from the entity name ("the", "member" and "household"), but not in the exact form (i.e. not "the household member")

### Why should entity level attributes contain the name of the entity?

There are many reasons why entity level attributes should contain the name of the entity in the text of the attribute. For example:

- If the entity attribute contains the name of the entity then the attribute will *automatically* exist within the correct entity in the Build Model, without the rule author having to artificially force the attribute into the entity via the Properties file.
- If the entity attribute contains the name of the entity then when declaring variables directly from the Word document into the Properties file, the correct entity will be selected by default in the 'Add Entity' dialog box.
- If the entity attribute contains the name of the entity then when creating a Public Name for an attribute via the Build Model, the attribute will automatically be put into the correct entity in the Properties file.
- If the entity attribute contains the name of the entity then it is possible to identify the entity level of the attribute by looking at the attribute text. This is useful during rule development. If you get 'scope' errors when writing entity rules, it will be easier to debug the rules if entity attributes contain the name of the entity.
- If the entity attribute contains the name of the entity then Name Substitution can be implemented. See section 4.3 for discussion of why Name Substitution can be particularly important in rulebases with entities.

#### OPM Help article

##### ***Define an attribute***

Check attribute entity levels:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Data%20model/Define\\_an\\_attribute.htm#Check](http://download.oracle.com/docs/html/E24270_01/Content/Data%20model/Define_an_attribute.htm#Check)

##### ***Set public identifiers for entities and attributes***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Integrating/Set\\_public\\_identifiers\\_for\\_entities\\_and\\_attributes.htm](http://download.oracle.com/docs/html/E24270_01/Content/Integrating/Set_public_identifiers_for_entities_and_attributes.htm)

##### ***Define a variable to use in a rule***

Create a new variable from within a Word document:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Variables%20and%20constant%20values/Define\\_a\\_variable\\_to\\_use\\_in\\_a\\_rule.htm?#Create](http://download.oracle.com/docs/html/E24270_01/Content/Variables%20and%20constant%20values/Define_a_variable_to_use_in_a_rule.htm?#Create)

#### 4.1.4 ATTRIBUTE PURPOSE SHOULD BE CLEAR FROM ATTRIBUTE TEXT

It should be clear from the attribute text whether the attribute's main purpose is for substantive logic, procedural logic, visibility logic or screen display logic.

Here are some example attributes which illustrate this:

- **Substantive:** “the person is eligible for income assistance” (conclusion attribute for a substantive rule)
- **Procedural:** “the person’s general details have been collected” (conclusion attribute for a procedural rule)
- **Visibility:** “the benefit eligibility goal should be displayed on the summary screen” (conclusion attribute for a summary screen visibility rule)
- **Screen attribute:** “the screen ‘Medical Details’ has been displayed” (screen attributes can be auto-generated from the question screen in the screens file, they are never rule conclusions but rather they feed into other rules, usually procedural rules)

Screen attributes are special purpose auto-generated attributes. Their attribute text is always in the form "the screen '<screen title>' has been displayed" (Global screen) or "the screen '<screen title>' has been displayed for <the entity>" (Entity level screen), therefore you should never use this sentence form for any other type of attribute.

## 4.2 ATTRIBUTE PARSING

One of the major benefits of using Oracle Policy Automation is the natural language rule authoring capability. If the rule author does not use appropriate attribute text and correct parsing, then they are ignoring one of the distinctive features of OPA, and also reducing the benefits that can be gained by using OPA in the first place.

Many people underestimate the importance of correct parsing. The attribute parse is used throughout OPA, OPM, and OWD, so if you do it badly, there are repercussions throughout the whole project and runtime. Here are some example places where attribute parse may be used:

- Rule documents
- Question screens
- Summary screen
- Decision report
- Data Review screen
- Generated documents

### 4.2.1 BOOLEAN ATTRIBUTES

A Boolean attribute is something to which you are looking for a yes/no answer, e.g. Does the person have health insurance?

A full parse for a Boolean attribute consists of:

- the positive form, e.g. the person **has** health insurance
- the negative form, e.g. the person **does not have** health insurance

- the question form, e.g. **does** the person **have** health insurance
- the uncertain form, e.g. the person **might have** health insurance

When creating and parsing Boolean attributes, it is important to make sure that *all* forms of the parse make sense, not just the positive form.

Correct parsing on an attribute requires correctly identifying the **operative verb**. The operative verb is the verb around which the various statement forms are created, i.e. positive form, negative form, question form and uncertain form. In the example parse above, the operative verb is "have".

#### OPM Help articles

##### **Compile rules and correct errors**

Understanding what parsing means:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Understa](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Understa)

Review the statement parses:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Review](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Review)

Identify the operative verb:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Identify](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Identify)

Select an alternative parse:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Select](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Select)

## 4.2.2 IDENTIFYING AND FIXING PARSING ISSUES WITH BOOLEAN ATTRIBUTES

Rule authors must learn how to identify and fix parsing issues. For the majority of sentences, the auto-generated parse will be exactly what you want. There are two situations when this may not be the case:

- There is *more than one* word in the sentence which matches a verb from the verbs list.
- There is *no* word in the sentence which matches a verb from the verbs list.

### 4.2.2.1 MULTIPLE KNOWN VERBS IN THE BOOLEAN ATTRIBUTE TEXT

If there are multiple words in the sentence which match a verb from the verbs list, the parser will use the first one as the operative verb, unless the rule author chooses otherwise. Often the first verb is the operative verb. Here is an example Boolean attribute ("the person submitted the tax return") with multiple verbs and what it looks like when parsed incorrectly vs correctly.

#### **Incorrect parsing using "return" as the operative verb:**

- the positive form, e.g. the person submitted the tax **return**

- the negative form, e.g. the person submitted the tax **do not return**
- the question form, e.g. **do** the person submitted the tax **return**
- the uncertain form, e.g. the person submitted the tax **might return**

✓ **Correct parsing using "submit" as the operative verb:**

- the positive form, e.g. the person **submitted** the tax return
- the negative form, e.g. the person **did not submit** the tax return
- the question form, e.g. **did** the person **submit** the tax return
- the uncertain form, e.g. the person **might have submitted** the tax return

As you can see, the positive form the sentence appears the same, i.e. "the person submitted the tax return". However, when you look at the rest of the parse, it is quite clear which verb is the correct operative verb for that sentence, i.e. "submit". In this example, the attribute would have been automatically parsed correctly since the operative verb *is* the first known verb in the attribute text. However, this is not always the case, so you need to pay attention to the parsing.

The OPM Help has excellent information about parsing, so please refer to the articles for further advice on this topic.

#### OPM Help articles

##### **Compile rules and correct errors**

Understanding what parsing means:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Understa](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Understa)

Review the statement parses:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Review](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Review)

Identify the operative verb:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Identify](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Identify)

Select an alternative parse:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Select](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Select)

#### 4.2.2.2 NO KNOWN VERBS IN THE BOOLEAN ATTRIBUTE TEXT

The out-of-the-box verb list contains hundreds of verbs, including the majority of verbs you will ever need in a rulebase. However, occasionally you may come across a new verb you want to use.

If it is a simple thing to re-word the sentence to use a verb which is already in the default verb list, then the pragmatic option is to do so. The other option is to add a new verb to the default verb list. It is very simple to add a new verb, however, keep in mind that it means you now have a customized verb list for the project.

#### OPM Help articles

##### **Configure the list of recognized verbs**

[http://download.oracle.com/docs/html/E24270\\_01/Content/Languages/Configure\\_list\\_of\\_recognized\\_verbs.htm](http://download.oracle.com/docs/html/E24270_01/Content/Languages/Configure_list_of_recognized_verbs.htm)

##### **Compile rules and correct errors**

Identify the operative verb:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Compiling%20and%20building/Compile\\_rules\\_and\\_correct\\_errors.htm#Identify](http://download.oracle.com/docs/html/E24270_01/Content/Compiling%20and%20building/Compile_rules_and_correct_errors.htm#Identify)

### 4.2.3 SPECIAL EXCEPTION: CAN NOT

If using the negative form of “can” as part of the operative verb in a Boolean attribute, it is necessary to put a space in the word, e.g. “the person can not read”. While "can not" is not the correct English spelling, it is necessary for this one exception because "can not" is the only example where the "not" is joined to the previous word. With all others, the "not" is a separate word, e.g. is not, has not, will not, does not, etc.

If "cannot" is part of the operative verb, and you use it without a space, the parser will either not parse the attribute, or it will not parse correctly. See the example below.



#### **Parsing "the person can not read" with a space in "can not":**

- the positive form, e.g. the person **can read**
- the negative form, e.g. the person **can not read**
- the question form, e.g. **can** the person **read**
- the uncertain form, e.g. the person **might read**



#### **Parsing "the person cannot read" without a space in "cannot":**

- the positive form, e.g. the person cannot **read**
- the negative form, e.g. the person cannot **did not read**
- the question form, e.g. **did** the person cannot **read**
- the uncertain form, e.g. the person cannot **might not read**

In real life, this issue very rarely arises because it is rare to need "cannot" as part of a compound verb in a rule attribute. Reasons for this are:

- The operative verb in the majority of attributes will be a single verb rather than a compound verb
- Rule attributes with compound verbs are most commonly written in the positive form.
- Rule attributes with compound verb negations are likely to use other auxiliaries, e.g. "the donation **has not been** collected"

#### 4.2.4 VARIABLE ATTRIBUTES

A variable attribute is something for which you are looking for a variable value such as a currency value, a date value, a number value or a text value, e.g. What is the person's rent? What is the person's date of birth? What is the person's number of cars? What is the person's job title?

Parsing variables is much simpler than parsing Booleans. Variables do not have operative verbs because variables are not full sentences; variables are phrases. This means that variables do not have negations. However, variables do have question forms.

The vast majority of variables you will ever use will have this question form: **What is <attribute text>?** For example:

- Attribute text: the person's date of birth
- Question: **what is** the person's date of birth

There are two exceptions to the "what is...?" rule for question generation for variables:

- **When the variable is the name substitution variable.** When name substitution is set up correctly, the auto-generated question form will be "who is...?" rather than "what is...?"
- **When the variable is a plural.** This very rarely occurs in real rulebases, but if it does you can mark an attribute as 'plural' in the Attribute Editor in the Properties file so that the default question form is "what are...?" rather than "what is...?"

### 4.3 SUBSTITUTION

Substitution is a great way to personalize the Web Determinations interview experience. For Global attributes, substitution is a 'nice to have'. For entity level attributes, substitution is required so it is clear to the end user which entity instance the interview is asking about on any particular entity level question screen. In other words, if you have multiple instances of "the child" and the question screen asks "What is the name of the child's school?", the end user does not know to which child the question is referring.

There are several ways to implement substitution:

- Substitution in attribute text
- Substitution in screen headings and labels
- Substitution in attribute text with second person sentence generation

- Gender pronoun substitution

The appropriate method will depend on the project element in which you are implementing it, e.g. attribute text, screen headings, screen labels, etc.

As a general rule, the type of substitution you should implement first is **substitution in attribute text**. This is also sometimes referred to as **name substitution** as it is most commonly used to substitute people's names into attribute text. The substitution does not have to be for names though, it can be used for other variable values as well.

There are several reasons why you should usually start with name substitution before considering the other types of substitution:

- Name substitution is easier to implement
- Name substitution is easier to maintain
- The small effort to implement name substitution has widespread benefit because name substitution will flow through to many areas of the rulebase, e.g. the question screens, the Summary Screen, the Decision Report, the Data Review Screen, the Debugger, and Regression Test Scripts.

---

#### 4.3.1 SUBSTITUTION IN ATTRIBUTE TEXT

Substitution for attribute text, including all parse forms for the attribute, should be set up in the Properties file. By far the most common use of attribute text substitution is name substitution, which is the example used in this section.

When done properly, all attributes at runtime, whether in Web Determinations or in the Debugger, will display with the person's name substituted directly into the attribute text. For example:

- "the person is eligible for the benefit" becomes "Jane is eligible for the benefit"
- "what is the person's date of birth?" becomes "what is Jane's date of birth?"

No hard-coding of substitution in screens is required for this. This is implemented purely through the Properties file and the substitution automatically flows through to any screen controls which use the auto-generated question form of the attribute.

### OPM Help articles

#### ***Change the text of an interview question or sentence:***

Set up substitution:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Set](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Set)

Substitute for actual value of a variable for its text:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Substitu4](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Substitu4)

#### ***Text Substitution Principles:***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Reference/Text\\_substitution\\_principles.htm](http://download.oracle.com/docs/html/E24270_01/Content/Reference/Text_substitution_principles.htm)

### 4.3.2 SUBSTITUTION IN SCREEN HEADINGS AND LABELS

Where there is text hardcoded on a screen, e.g. a screen heading or screen label, it will not be impacted by the regular attribute text substitution described in section 4.3.1. To substitute names or other values into screen headings and labels there is another method. See the OPM Help article below for instructions.

A few tips with regard to this form of substitution:

- Use this type of substitution sparingly. If regular substitution in the attribute text (as per section 4.3.1 above) makes the interview and questioning sufficiently clear and unambiguous, then there is no need to add further substitution in screen headings and labels as well.
- Do not use this form of substitution to manually hardcode name substitution into free form question text. If you want name substitution in attribute and question text, use regular attribute substitution as per section 4.3.1 above.
- Always create a public name for the substitution variable. For example, if you want to substitute the value of "the person's age in years" into a screen label, make sure to create a public name for the attribute "the person's age in years", and use the public name in the substitution. If you do not do this, your substitution will most likely get broken at a later point as the auto-generated attribute IDs (e.g. p1@Rules\_Rules\_doc) are not static.

**OPM Help article*****Change the text of an interview question or sentence:***

Substitute an attribute value into the text on screens:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Substitu](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Substitu)

**4.3.3 SUBSTITUTION IN ATTRIBUTE TEXT WITH SECOND PERSON SENTENCE GENERATION**

Something which new rule authors often get wrong is thinking that if they want the questions to display in second person on the screens in Web Determinations, then they have to write the attributes in second person. This is not so. Even if the attributes are written in third person (as they should be), there is an option to switch the display to second person in Web Determinations.

Here are example question forms in third person and in second person:

- Third person: Is the person happy? (default)
- Second person: Are you happy? (when second person sentence generation is turned on)

Refer to the OPM Help for instructions on second person sentence generation.

**OPM Help article*****Change the text of an interview question or sentence***

Display interview questions in second person form:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Display](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Display)

**4.3.4 GENDER PRONOUN SUBSTITUTION**

When attributes have been worded properly and name substitution has been set up correctly, any attributes where the subject of the name substitution appears multiple times in the sentence will appear like this:

- **the person** has submitted **the person's** tax return (attribute)
- **Jane** has submitted **Jane's** tax return (attribute with name substitution)

Having Jane's name appear twice in the one sentence is not very natural language. A more natural way to express the sentence would be:

- **Jane** has submitted **her** tax return (attribute with name substitution and gender pronoun substitution)

The second time "the person" appears in the sentence, the gender pronoun is substituted instead of the person's name.

Gender pronoun substitution is easily implemented with a couple of additional steps after regular name substitution has been set up.

#### OPM Help articles

##### ***Change the text of an interview question or sentence:***

Substitute a gender pronoun for a text variable:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Substitu3](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Substitu3)

Collect the gender of a person:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Collect](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Collect)

## CHAPTER 5. SCREENS

### 5.1 USABILITY CONSIDERATIONS

As a general guide, anything you can do to reduce clicks, keystrokes and mouse use, for the end user is a good thing from a usability perspective. There are many simple things you can do in the rulebase with out-of-the-box functionality which addresses this. Here are some examples.

- **Scrolling is bad.** Avoid putting so many questions on a single question screen so that the user is forced to scroll around to complete the screen. For entity collect screens, consider setting the Display Style to Landscape rather than Portrait. This usually results in less scrolling around for the user.
- **Reduce the number of clicks.** Aim to reduce the number of clicks and keystrokes required by the user to complete the screen. For example:
  - **Radio buttons for Booleans.** For Booleans, use radio buttons, not drop-down lists. With radio buttons, the user has one click to select, i.e. click on Yes or No. With a drop-down list, the user has two clicks to select, i.e. one click to open the drop-down list and a second click to select Yes or No. Note that radio buttons are the default input type for Booleans on question screens.
  - **Drop-down lists for variables.** For variables, where appropriate create drop-down lists for the user to select from, rather than forcing the user to type out an answer. This has the added benefit of reducing user input errors. An appropriate use of a drop-down list might be giving the user a list of States when asking them for their State of residence. An inappropriate use of a drop-down list is anytime where the response to the question is not something from a pre-defined set of options, e.g. asking the person for their name.
  - **Default values for Booleans and variables.** Use default values for fields which have a high likelihood of a common answer across users. This will reduce the number of clicks, keystrokes, and mouse use for most users.
    - *Note:* Some customers do not like default values. Their concern is that end users will get lazy and not think about each question properly before submitting the question screen. Not having default values requires the user to actively provide answers for all mandatory questions before submitting the screen. Consult with your customer before implementing default values in the screens file.

### 5.2 APPROPRIATE USE OF FREE FORM TEXT

As general rule, avoid or minimize use of 'free form' text for question text in a screen control. If the attributes have been parsed correctly, the auto-generated sentence from the parser should be sufficient. See section 4.2 for information about attribute parsing.

However, there are some situations where it is appropriate to use free form text:

- On a general details type screen where you are collecting basic details like name, date of birth, gender, etc.
- On entity collect screens
- On a question screen which collects a series of similar items

### 5.2.1 GENERAL DETAILS SCREEN

A 'general details' screen is unlikely to be collecting complex questions where the full question text is required to understand the meaning of the question. It will usually be collecting simple basic details, e.g. see screen below.

**ORACLE** Web Determinations

[Summary](#) | [Data Review](#) | [Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

### Applicant's Details

What is the applicant's first name? \*

What is the applicant's last name? \*

What is the applicant's date of birth? \*

What is the applicant's gender? \*

What is the applicant's social security number? \*

Using the full auto-generated question text is fine, but using free form text on this screen can make it simpler, and can do so without losing context, e.g. see screen below.

The other consideration is that a 'general details' screen like this in any other application (i.e. not OPA) is likely to have short labels rather than long question text, so using free form text here will make the screen feel more familiar to the end user.

If you do happen to have a couple of complex questions on the screen (and if it is not appropriate to put those complex questions on a separate screen), then perhaps use the full question text for those questions while using the shortened free form text for the simple questions.

**OPM Help article**

***Change the text of an interview question or sentence***

Customize question text:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Customiz2](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Customiz2)

**5.2.2 ENTITY COLLECT SCREEN**

As entity collect screens are the screens on which the user creates instances of the entity, they are generally going to be more busy and cluttered than a regular question screen. When creating entity instances on the entity collect screen, especially if the layout is set to Landscape, each additional instance adds to the width of the screen. Therefore, it is preferable to minimize the question text on entity collect screens by using the Free Form text option.

The entity collect screen below is using the full auto-generated question text. With only one entity instance, it looks okay.

**ORACLE** Web Determinations

Summary | Data Review | Save | Save As | Load | Restart | Close

### Household Members

What is the household member's first name? \*

What is the household member's last name? \*

What is the household member's date of birth? \*

What is the household member's gender? \*

What is the household member's social security number? \*

Remove

Add Person

Remove Person

Submit

However, once multiple entity instances are created, the screen looks more cluttered, e.g. see screen below.

**ORACLE** Web Determinations

Summary | Data Review | Save | Save As | Load | Restart | Close

### Household Members

What is the household member's first name? \*  \*  \*

What is the household member's last name? \*  \*  \*

What is the household member's date of birth? \*  \*  \*

What is the household member's gender? \*  \*  \*

What is the household member's social security number? \*  \*  \*

Remove  Remove  Remove

Add Person

Remove Person

Submit

By using free form text, the entity collect screen can be made to look less cluttered without losing context. Also, as the screen will not get as wide/long when instances are added (compared to when the default question text is used), it is less likely the user will need to scroll around the screen.

Using Free Form text adds a maintenance overhead. However, it should be minor as there are typically only a couple of entity collect screens per rulebase, and the benefit gained from a usability perspective by using Free Form text in this situation outweighs the minor additional maintenance. Also, the entity collect screens are less likely to change frequently over time anyway.

**OPM Help articles**

***Change the text of an interview question or sentence***

Customize question text:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Customiz2](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Customiz2)

***Collect information about entity instances***

Define a screen for collecting entity instances:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Collect\\_information\\_about\\_entity\\_instances.htm#Define](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Collect_information_about_entity_instances.htm#Define)

Collect attributes for the entity:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Collect\\_information\\_about\\_entity\\_instances.htm#Specify](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Collect_information_about_entity_instances.htm#Specify)

**5.2.3 COLLECTING A SERIES OF SIMILAR THINGS**

Sometimes you will have question screens where you are collecting a large number of items of a similar type, e.g. income values from many different income sources. In these situations, the auto-generated natural language questions can make the screen appear rather busy, e.g. see the example screen below.

**ORACLE** Web Determinations

[Summary](#) | [Data Review](#) [Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: ExampleRulebase Locale: en-US User ID: guest

## Income

What is the person's monthly income from wages and salary? \*

What is the person's monthly income from tips? \*

What is the person's monthly income from self-employment? \*

What is the person's monthly income from investments? \*

What is the person's monthly income from boarders/lodgers? \*

What is the person's monthly income from child support? \*

The questions are quite repetitive, and repeating the full natural language question text does not really add value. To make the question screen more user-friendly, you can use free-form text and include a label at the top to add context, e.g. see the example screen below.

**ORACLE** Web Determinations

[Summary](#) | [Data Review](#) [Save](#) | [Save As](#) | [Load](#) | [Restart](#) | [Close](#)

Rulebase: ExampleRulebase Locale: en-US User ID: guest

## Income

*Please provide monthly incomes values for the following sources:*

Wages and salary: \*

Tips: \*

Self-employment: \*

Investments: \*

Boarders/lodgers: \*

Child support: \*

The free-form text does not detract from the meaning of the questions, but rather makes the screen easier to read. The minor additional work and maintenance on this screen is justified by the improvement in usability.

**OPM Help articles**

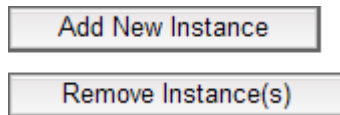
***Change the text of an interview question or sentence***

Customize question text:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Change\\_text\\_of\\_interview\\_question\\_or\\_sentence.htm#Customiz2](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Change_text_of_interview_question_or_sentence.htm#Customiz2)

**5.3 CONFIGURING THE ADD/REMOVE INSTANCE BUTTON TEXT ON ENTITY COLLECT SCREENS**

On an entity collect screen, the default button text for adding and removing instances is "Add New Instance" and "Remove Instance(s)".



The button text can easily be configured in the screens file for each different entity collect screen. While not necessary, it is nicer from the user perspective to have specific button text rather than the generic default text.

For example, if you had the entities 'the household member' and 'the household member's income', consider using button text such as the following on the respective entity collect screens:



**OPM Help article**

***Collect information about entity instances***

Define a screen for collecting entity instances (step 6):

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Collect\\_information\\_about\\_entity\\_instances.htm#Define](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Collect_information_about_entity_instances.htm#Define)

## CHAPTER 6. INTERVIEW FLOW

There are several ways to control how the interview flows in Web Determinations:

- Screen Order
- Screen Flow
- Forward/backward chaining – sometimes referred to as ‘question search’, ‘interview cycle’, or ‘get next question’

In older rulebases, the interview flow was usually controlled entirely by the engine’s forward/backward chaining working over the procedural rules and substantive rules of the rulebase. From OPA 10.x onwards, the preferred approach is to use a combination of the Screen Order functionality, the Screen Flow functionality and forward/backward chaining.

### OPM Help articles

#### ***Oracle Determinations Engine and the Inference Cycle***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Introducing%20Oracle%20Policy%20Modeling/Deter\\_Engine\\_and\\_infer\\_cycle.htm](http://download.oracle.com/docs/html/E24270_01/Content/Introducing%20Oracle%20Policy%20Modeling/Deter_Engine_and_infer_cycle.htm)

### 6.1 SCREEN ORDER

For the vast majority of rulebases, Screen Order in combination with forward/backward chaining provides ample functionality for controlling the interview flow.

As a general rule, use the Screen Order functionality unless there is a good reason otherwise. Compared to Screen Flow, Screen Order is easier to implement and takes advantage of the intelligent forward/backward chaining of the engine, while at the same time taking into account the ordering of screens in the screens file.

See section 6.2 for some example situations where Screen Flow should be considered instead Screen Order.

### OPM Help article

#### ***Define interview screen order***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Define\\_interview\\_screen\\_order.htm](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Define_interview_screen_order.htm)

### 6.2 SCREEN FLOW

For most situations, using a Screen Flow to control the interview flow will be more effort and less useful than using a Screen Order. However, there are some situations where Screen Flow should be considered. Here are some examples.

- Example 1. Need to display question screens for which all the data is already known
- Example 2. Need to display many information-only screens
- Example 3. Need to be able to re-run the interview repeatedly in the same session

Each of these examples is discussed in more detail below.

#### OPM Help article

##### *Define interview screen flow*

[http://download.oracle.com/docs/html/E24270\\_01/Content/Interviews%20and%20flows/Define\\_interview\\_screenflow.htm](http://download.oracle.com/docs/html/E24270_01/Content/Interviews%20and%20flows/Define_interview_screenflow.htm)

### 6.2.1 DISPLAYING SCREENS WITH KNOWN DATA

**Example 1:** Need to display screens for which the data is already known

Imagine you have the following requirements for your rulebase:

- Need to model a claim form which has a fairly linear progression and minimal branching of logic.
- Some of the base data will be fed to the rules from an external system such as a case management system. The particular base data being sent will vary from case to case, and may include all base data, or sometimes none of the base data.
- All relevant screens of the claim form assessment should be displayed to the end user, regardless of whether all the required data on the question screen is known or not. The screens should be shown so the end user can review and amend if necessary.

#### ➔ Consider using a Screen Flow

The primary reason for considering a Screen Flow in this example is the requirement to display all relevant screens, regardless of whether the data on that screen is known or not. A Screen Order will skip past relevant screens for which all the required data is known. Since the claim form has minimal branching of logic, the Screen Flow itself should be relatively simple, thus the extra effort in developing and maintaining the Screen Flow should be reasonable.

Screen attributes may also be used to display a screen with known attributes. However, in this example where it is possible that a lot of data is being sent, it makes more sense to use a Screen Flow rather than creating screen attributes for every question screen.

### 6.2.2 DISPLAYING INFORMATION-ONLY SCREENS

**Example 2:** Need to display many information-only screens

Imagine you have the following requirement for your rulebase:

- Need to develop a rulebase interview where there are a large number of information-only screens compared to regular interview screens. (Information-only screens contain information, but no base attributes. A regular interview screen contains one or more base attributes.)

➔ **Consider using a Screen Flow**

When a question screen contains information, but no base attributes, it will never be called in a Screen Order. A Screen Order displays screens containing questions for which data is required. If there are no questions, there is no data to be collected. With a Screen Flow, you can specify that a particular screen should be displayed, regardless of what is on the screen.

If there are only a couple of information-only screens compared to regular interview screens, consider using a Screen Order rather than a Screen Flow, especially if the interview has a lot of branching of logic. You would also need to create screen attributes for the information-only screens, and use them in procedural rules which are working in combination with a Screen Order. See section 6.3 for information about screen attributes.

---

### 6.2.3 ABILITY TO RE-RUN INTERVIEW IN SAME SESSION

**Example 3:** Need to be able to re-run the interview repeatedly in the same session

Imagine you have the following requirements for your rulebase:

- The end user needs to be able to re-run the interview over and over again in the same session.
- When re-running the interview flow, the data just entered needs to be remembered (note: the data will be remembered if re-running in the same session)

➔ **Consider using a Screen Flow**

With a Screen Order, once the required base level data has been collected and the rulebase goals have been evaluated, the end user cannot re-run the interview flow unless they re-start the session. Re-starting the session will cause all the collected data to be forgotten. If the session is not re-started, the end user may navigate back to particular question screens one and a time via the Data Review Screen, but they cannot go through the whole interview flow again in the same session. However, a Screen Flow allows for the interview to be re-run over and over again in the same session.

---

## 6.3 SCREEN ATTRIBUTES

Screen attributes should be used sparingly. The flow of the interview should usually be controlled by a Screen Order (working in combination with the forward/backward chaining of the engine) or a Screen Flow.

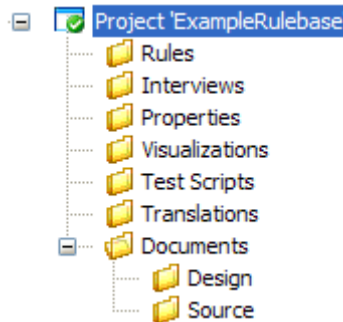
If the interview is fairly dynamic (i.e. a lot of branching of logic) and there are just a couple of information-only screens, consider using a Screen Order in combination with procedural rules, with screen attributes for the few information-only screens.

## CHAPTER 7. ORGANIZING CONTENT

### 7.1 PROJECT FOLDERS AND FILES

#### 7.1.1 CREATE LOGICAL SUBFOLDERS IN WHICH TO STORE DIFFERENT RULE DOCUMENTS

When a new project is created in Oracle Policy Modeling, the default folder structure is created:



To help organize the project content, you can create subfolders within the default folder structure to help organize content, e.g. add sub-folders, sub-subfolders, etc. inside the Rules folder to organize the rule documents.

While the rule author is free to modify the top level default folders, generally they should not. Some thought has gone into the default folder structure, and the vast majority of OPA projects around the world use it. It is beneficial to have a structure which is familiar generally to rule authors because:

- When people move teams, move projects, move countries, they can open up any rule project and the high level folder structure will look familiar.
- If you need advice from someone outside the team, and need to send them the project, they do not need to hunt around the project to work out where your rule documents are, where you screens file is, etc.

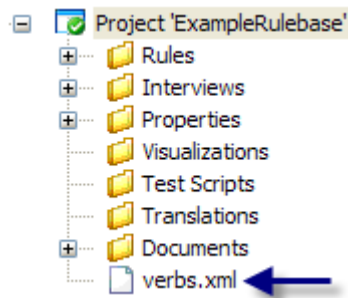
#### OPM Help article

##### ***Organize project files***

[http://download.oracle.com/docs/html/E24270\\_01/Content/Projects%20and%20files/Organize\\_project\\_files.htm](http://download.oracle.com/docs/html/E24270_01/Content/Projects%20and%20files/Organize_project_files.htm)

#### 7.1.2 ADDING THE CUSTOM VERBS FILE TO THE PROJECT

If you have added custom verbs to the project, and thus have a custom verbs file, consider adding the custom verbs file to the Project Folders so it is clear to others it is there. The verbs file could be added to the end of the Folder Structure or it could go into an existing or new folder.



### OPM Help article

#### **Create, modify or delete a project**

Add existing files to a project:

[http://download.oracle.com/docs/html/E24270\\_01/Content/Projects%20and%20files/Create\\_modify\\_or\\_delete\\_a\\_project.htm#Add3](http://download.oracle.com/docs/html/E24270_01/Content/Projects%20and%20files/Create_modify_or_delete_a_project.htm#Add3)

## 7.2 RULE DOCUMENTS

### 7.2.1 HEADING LEVELS

When writing technical documentation, training documentation, whitepapers or any other type of document, you probably use heading levels to help organize the contents of your document. This makes the document easier to read, easier to understand and easier to maintain.

The same principle applies to rule documents. Rule authors should use heading levels to organize the rules in their rule documents. The OPM toolbar has some built-in heading levels, but you are not restricted to these. You can use the regular default MSWord heading levels instead. However, unless there is a particular reason otherwise, use the OPM heading levels.

### 7.2.2 TABLE OF CONTENTS

As the rules are authored in Microsoft Office documents, there are features of the Microsoft Office applications that you can take advantage of when writing rules. A great example of this is adding a table of contents. If you have used heading levels to organize your rule documents (which you should, see section 7.2.1), then you can generate a table of contents from those heading levels.

There is nothing OPM-specific about generating a table of contents, it is done by merely using existing functionality within MSWord. MSWord allows you to choose which Styles to incorporate in the table of contents, and how many levels deep the table of contents should go. Consult the MSWord Help if you need assistance generating a table of contents.

Note that if a rule document has a table of contents, it will take a little longer to compile if the table of contents is in view when compiling. You can easily avoid this issue by scrolling the table of contents out of view before compiling the document.

### 7.2.3 SPACING BETWEEN RULES

Just as you would usually have spacing between paragraphs in any other type of document to make it easier to read, it is helpful to put a blank line in between each rule for the same reason. The OPM style 'Blank Line' may be used for this, or any other non-OPA style.

### 7.2.4 ADJUSTING RULE TABLE COLUMN WIDTH

When creating a rule table in Word or Excel, the columns will initially be the default widths. You do not need to keep this default column width. If adjusting the column width makes the rule more readable and generally more aesthetic, then feel free to do so.

With regard to the default table width, you should keep the default table width for Word, but adjust as required for Excel.

#### Word rule tables

The Word rule table below is using the default column widths. The contents of the left column is crammed into a narrow column while there is plenty of empty unused space in the width column on the right. It looks untidy and is not as readable as it could be.

<b>the final amount</b>	
<b>the first amount + the second amount</b>	the application date < 2010-01-01
<b>the third amount * the fourth amount</b>	the application date >= 2010-01-01 and the application date < 2011-01-01
<b>the fifth amount - the sixth amount</b>	<b>otherwise</b>

In the example rule table below, the column widths have been adjusted to balance the contents on either side, which makes the logic is easier to read. Also, the table takes up less space on the page, and looks neater and tidier in the document.

<b>the final amount</b>	
<b>the first amount + the second amount</b>	the application date < 2010-01-01
<b>the third amount * the fourth amount</b>	the application date >= 2010-01-01 and the application date < 2011-01-01
<b>the fifth amount - the sixth amount</b>	<b>otherwise</b>

**Excel rule tables**

The Excel rule table below is badly set out. The columns are too narrow for the contents, and some of the contents is not visible.

State	Income Threshold
Alabama	Poverty Level (175%
Alaska	Poverty Level (150%
Arizona	Poverty Level (200%
Arkansas	Poverty Level (150%
California	State Median Incom
Colorado	Poverty Level (185%
Connecticut	Poverty Level (150%
Delaware	Poverty Level (200%
District of Colum	State Median Incom



Here is the same rule table set out more nicely and with all the contents visible.

State	Income Threshold
Alabama	Poverty Level (175%)
Alaska	Poverty Level (150%)
Arizona	Poverty Level (200%)
Arkansas	Poverty Level (150%)
California	State Median Income (60%)
Colorado	Poverty Level (185%)
Connecticut	Poverty Level (150%)
Delaware	Poverty Level (200%)
District of Columbia	State Median Income (60%)



The rule below is okay because all the text is visible. However, the version of this rule further below is better as it is set out more concisely.

Area	Household Number	Threshold
48 contiguous States and DC	1	907.50
	2	1225.83
	3	1544.17
	4	1862.50
	5	2180.83
	6	2499.17
	7	2817.50
	8	3135.83
	>= 9	3135.83 + 318.33 * (Household Number - 8)

You can use centering and word wrap to lay out the rules more neatly in Excel, e.g.

Area	Household Number	Threshold
48 contiguous States and DC	1	907.50
	2	1225.83
	3	1544.17
	4	1862.50
	5	2180.83
	6	2499.17
	7	2817.50
	8	3135.83
	>= 9	3135.83 + 318.33 * (Household Number - 8)



### 7.2.5 SOFT ENTERS IN CALCULATION RULES

As calculation rules get longer they become more difficult to read. Consider the two example rules below.

The rule below is short, has few inputs, and thus is very easy to read:

**the total cost of the meal = the cost of the food + the cost of the drink**

The rule below is longer, has far more inputs, and is a little more difficult to read:

**the person's total cost for the trip to the zoo = the cost of the bus fare + the entry fee to the zoo + the cost of lunch at the zoo cafe + the cost of peanuts to feed the elephants + the cost of the ticket for the seal show + the cost of souvenirs at the gift shop**

You can make a long calculation rule easier to read by using a 'soft enter' (Shift+Enter). A 'regular enter', i.e. just hitting the enter key, inserts a hard enter which moves the remaining content to a new line and also indicates a new paragraph. This means that all formatting associated with a new paragraph will apply. From an OPM rule perspective it means the remaining content would become detached from the start of the rule, and the rule would be broken.

A soft enter moves the remaining content to a new line, but it does not indicate a new paragraph. This means you can make the rule appear like the example below, without the rule being broken.

**the person's total cost for the trip to the zoo =  
the cost of the bus fare  
+ the entry fee to the zoo  
+ the cost of lunch at the zoo cafe  
+ the cost of peanuts to feed the elephants  
+ the cost of the ticket for the seal show  
+ the cost of souvenirs at the gift shop**

In the example above, a soft enter was inserted between each item in the calculation, making it much easier to read. Below is a screenshot of the same rule with the formatting marks displayed. Notice that the only paragraph mark (¶) appears at the very end of the rule, whereas the other lines end in a soft enter (↵).

```

the-person's-total-cost-for-the-trip-to-the-zoo.=↵
the-cost-of-the-bus-fare.↵
+the-entry-fee-to-the-zoo.↵
+the-cost-of-lunch-at-the-zoo-cafe.↵
+the-cost-of-peanuts-to-feed-the-elephants.↵
+the-cost-of-the-ticket-for-the-seal-show.↵
+the-cost-of-souvenirs-at-the-gift-shop¶

```

### 7.2.6 ADDING COMMENTS TO RULE DOCUMENTS

Just as programmers may be encouraged to put comments in their code, rule authors should put comments in their rule documents when it is helpful. For many rules it may be obvious from the rule itself what is going on. However, for very complex logic, such as rules involving inferred relationships or temporal reasoning, it may not be clear what is going on. A brief explanation can be extremely useful for other rule authors, as well as for the original rule author if they have not worked on that part of the rulebase in a long time.

The OPM style 'Blank Line' may be used for this as that style is ignored by the compile macro. Regular MSWord styles such as 'Normal' may also be used for comments and notes in the rule document as the compile macro ignores all non-OPA styles.

### 7.2.7 PAGES PER RULE DOCUMENT

There is no absolute answer to 'how many pages of rules per rule document?'. As general guidance, rule documents should generally be under 10 pages long. More than 10 pages is long, 20 pages is very long. This is not to say that you should never ever have a 20 page rule document, or that it will not compile, but just that it would be quite rare for it to be appropriate to have a single rule document that long.

Here are some reasons for avoiding excessively long rule documents:

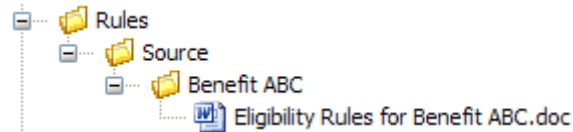
- Multi-developer projects. Only one person can work on a particular rule document at a time.
- Shorter documents compile faster.
- Easier to find the rules you are working on (although appropriate use of heading levels and a table of contents helps alleviate this)

You do not need to decide upfront exactly how you are going to break up the rules into different documents. It is okay to plan out upfront the high level division of documents, and then later as the rules are developed, split out some individual rule documents into multiple rule documents.

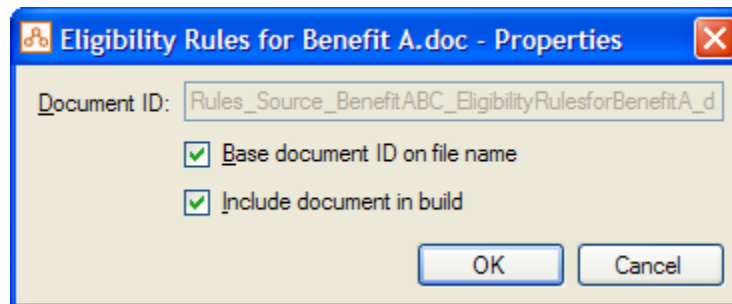
## 7.3 ATTRIBUTES

### 7.3.1 TIDY UP AUTO-GENERATED ATTRIBUTE IDS

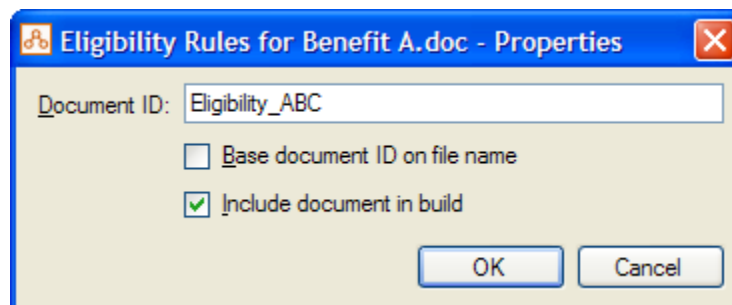
Imagine you have created the following sub-folders and rule document:



While this structure may be very sensible for organizing the rule documents, it will result in rather verbose auto-generated attribute IDs. To make the auto-generated IDs more concise, right-click on the rule document and select Properties. This will show you the auto-generated ID extension:



Un-check the option “Base document ID on file name” and supply an alternative. Make sure to choose something unique for each document.

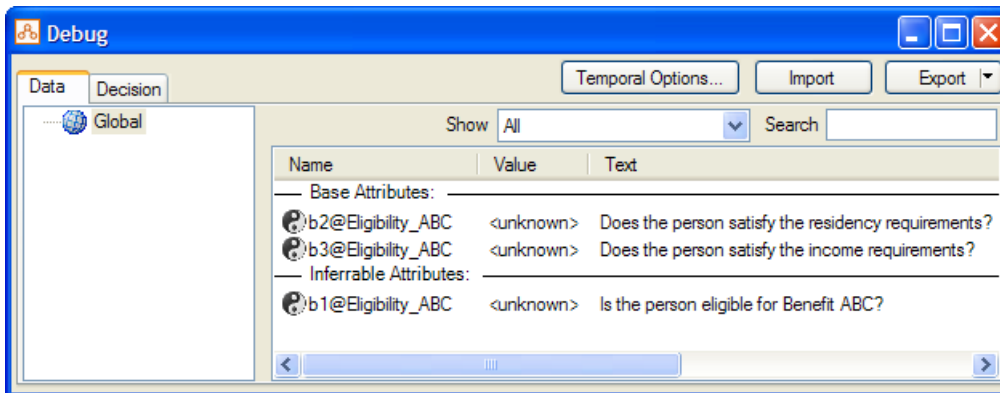
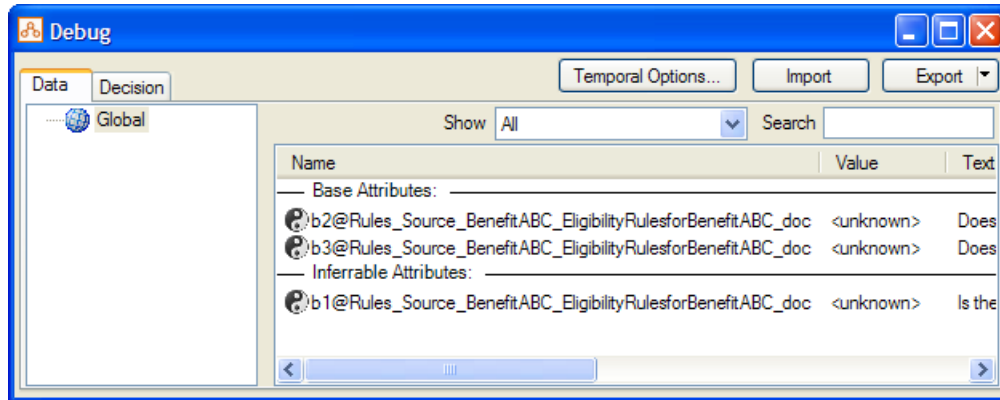


The auto-generated IDs will then use the shortened form, e.g.

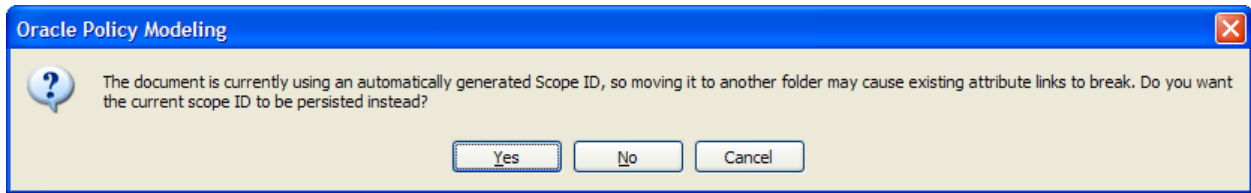
Default auto-generated ID	Modified auto-generated ID
b1@Rules_Source_BenefitA_EligibilityRulesforBenefitABC_doc	b1@Eligibility_ABC
p1@Rules_Source_BenefitA_EligibilityRulesforBenefitABC_doc	p1@Eligibility_ABC

The benefits of doing this are:

- The attribute ID column does not take over the whole screen when viewing the Build Model or the Debugger (compare the two screenshots below)



- You will not get the 'persist current scope ID' message every time a document is moved to a different folder



## USEFUL LINKS

### Online Help Guides

- Oracle Policy Modeling 10.3 Help: [http://download.oracle.com/docs/html/E24270\\_01/toc.htm](http://download.oracle.com/docs/html/E24270_01/toc.htm)
- Oracle Policy Automation 10.3 Developer Help:  
[http://download.oracle.com/docs/html/E24274\\_01/toc.htm](http://download.oracle.com/docs/html/E24274_01/toc.htm)

### General Information

- Oracle Policy Automation on the Oracle Technology Network: <http://www.oracle.com/technetwork/apps-tech/policy-automation/overview/index.html>
- Oracle Policy Automation 10.3 Features and Benefits: <http://www.oracle.com/technetwork/apps-tech/policy-automation/documentation/opa10-3-features-benefits-451968.pdf>
- Oracle Policy Automation 10.3 System Requirements: <http://www.oracle.com/technetwork/apps-tech/policy-automation/documentation/opa10-3-systemreqs-451965.pdf>
- Guide to Designing a Policy Model: <http://www.oracle.com/technetwork/apps-tech/policy-automation/learnmore/opaguidetodesigningapolicymodel-1531936.pdf>

### Forums

- Oracle Policy Automation external discussion forum:  
<https://forums.oracle.com/forums/forum.jspa?forumID=828>
- Oracle Policy Automation internal product forum:  
<http://myforums.oracle.com/jive3/forum.jspa?forumID=3918> (accessible by Oracle staff only)