ORACLE INTELLIGENT ADVISOR

BEST PRACTICE GUIDE FOR POLICY MODELERS

Author: Jasmine Lee

Last Updated: May 2020

ABOUT THE AUTHOR

Jasmine Lee is an expert in advanced Oracle Intelligent Advisor rule design and implementation topics. She has been working with the Oracle Intelligent Advisor (formerly known as Oracle Policy Automation) product line for more than 18 years, and has experience designing Oracle Intelligent Advisor policy models for customers around the world. She has lived and worked in Australia, Europe and the United States and is currently based in Washington DC. When not designing policy models, Jasmine can be found speedsolving the Rubik's Cube and playing with her cat.

DOCUMENT PROPERTIES

Author	Jasmine Lee

VERSION HISTORY

Date	**Name**	**Comments**
November 2011	Jasmine Lee	Released for internal review.
November 2011	Jasmine Lee	Updates following feedback from internal review.
March 2012	Jasmine Lee	Updated cover page, links to training courses and Useful Links section.
March 2013	Jasmine Lee	Added sections on entity functions, current date, combining functions, event rules, interview screens and general tips. Updated definitions and links. Other minor updates throughout the document.
March 2017	Jasmine Lee	Removed OPA 10-specific content. General revisions and updates throughout the document to align with OPA 12.
April 2017	Jasmine Lee	Minor updates following internal review.
February 2018	Jasmine Lee	Updated links to OPA Documentation Library.
December 2019	Fiona Guy, Jasmine Lee	Updated to reflect product name change from Oracle Policy Automation to Intelligent Advisor.
May 2020	Jasmine Lee	Updated links to Intelligent Advisor Documentation Library. Other minor edits throughout the document.

ACRONYMS AND ABBREVIATIONS

OPA	Oracle Policy Automation – now known as Oracle Intelligent Advisor
OPM	Oracle Policy Modeling
OU	Oracle University

DEFINITIONS

Automatic screen	An automatic screen is a screen which is generated automatically in the Interview using default settings only. This is different to an 'interview screen' which is configured by the policy modeler in the Interview tab in Oracle Policy Modeling.
Boolean attribute	An attribute whose value will typically be True or False. In Intelligent Advisor, Booleans can also be Uncertain or Unknown.
Currency attribute	An attribute with the data type 'Currency'.
Date attribute	An attribute with the data type 'Date'.
DateTime attribute	An attribute with the data type 'Date and Time'.
Design time	Design time refers to when the rules are being created and the interview is being designed. This is distinct from ‘runtime’ which is when the rules are being executed or ‘run’.
End user	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.
Grouping operator	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.
Interview screen	The term 'interview screen' is used in this document to mean a configured screen created in the Interview tab in Oracle Policy Modeling. This is different to an 'automatic screen'.
Negation	The negative form of a sentence parse.
Non-Boolean attribute	An attribute whose value is varied as opposed to whose value is Yes/No or True/False. A non-Boolean attribute will have a data type of Currency, Date, Number, Text, DateTime or TimeOfDay.
Number attribute	An attribute with the data type 'Number'.
Parsing	Parsing refers to generating the various sentence forms of an attribute. A full parse for a Boolean attribute consists of the positive form, the negative form, the question form and the uncertain form. A full parse for a Non-Boolean attributes includes the phrase form and the question form.
Procedural logic, rules and attributes	Logic, rules and attributes whose purpose is to control interview flow and data collection.
Policy Modeler	A person who implements the non-programming aspects of an OPM project such writing rules and configuring interview screens.

Runtime	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 an Intelligent Advisor interview or a custom front-end application, or run as a batch process without a user interface.
Source material	The content from which the rules are being built. This may include legislation, regulations, policy, design documents, use cases and other material.
Substantive logic, rules and attributes	Logic, rules and attributes whose purpose is to determine the outcomes for the policy model goals based on legislation, regulations or other source material.
Text attribute	An attribute with the data type 'Text'.
TimeOfDay attribute	An attribute with the data type 'Time of Day'.
Visibility logic, rules and attributes	Logic, rules and attributes whose purpose is to control whether screen controls (attributes, labels, headings) are displayed or hidden on an Interview screen.

CONTENTS

CHAPTER 1. INTRODUCTION

The purpose of this document is to provide some Oracle Policy Modeling (OPM) best practice guidelines for policy modelers working on Oracle Intelligent Advisor implementations.

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

This document is not intended for training someone who is new to Intelligent Advisor. At a minimum, anyone working on an Intelligent Advisor implementation as a policy modeler should complete the following Oracle University courses (or equivalent training):

- **Introduction to Intelligent Advisor**
  https://education.oracle.com/introduction-to-oracle-policy-automation/courP_6524

- **Oracle Policy Modeling for Policy Experts**
  https://education.oracle.com/oracle-policy-modeling-for-policy-experts/courP_8376

This document assumes the policy modeler is familiar and confident with all the content of these courses. There are also are some sections of this document which assume Intelligent Advisor 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 information about how to use the features and functionality of OPM, please refer to the Policy Modeling User Guide that is available in the Intelligent Advisor Documentation Library.

1.1 INTELLIGENT ADVISOR DOCUMENTATION LIBRARY

The Intelligent Advisor Documentation Library is available via the question mark (?) icon in the top-right corner of Oracle Policy Modeling. The latest version of the documentation is also available here:
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Default.htm

This document has many references to Policy Modeling User Guide articles. The URL links to specific 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 the latest version the Policy Modeling User Guide.

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. However, the terminology and descriptions align with Intelligent Advisor 12 (formerly known as OPA 12). If you are using OPA 10, please refer to the OPA 10 version of this document.

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.

Substantive rules

Substantive rules determine the outcome for the policy model 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 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 control whether screen controls (attributes, labels, headings) are displayed or hidden from an interview 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 most of the content. However, section 2.2 discusses some things you should never ever do:

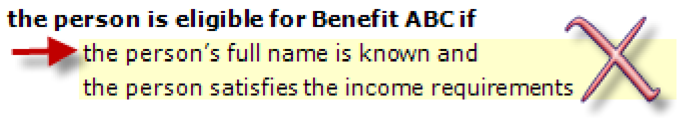
- ✘ **Never mix procedural logic into substantive logic rules**
- ✘ **Never mix visibility logic into substantive logic rules**
- ✘ **Never use the conclusions of substantive logic rules as visibility attributes**

Each of these statements is discussed below with examples.

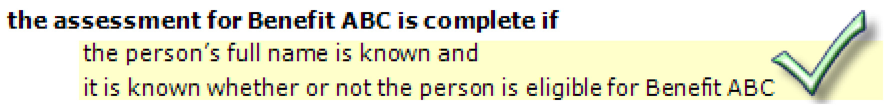
2.2.1 NEVER MIX PROCEDURAL LOGIC INTO SUBSTANTIVE LOGIC

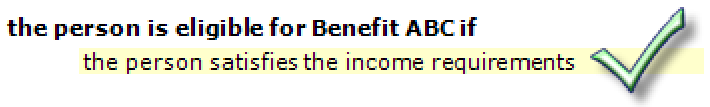
Substantive logic rules should never be dependent on procedural attributes, i.e. there should never be procedural attributes used as premises in substantive logic rules.

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.

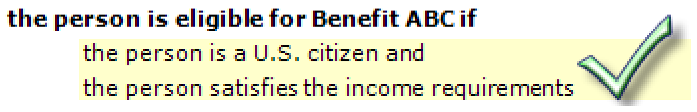

the person is eligible for Benefit ABC if
the person’s full name is known and
the person satisfies the income requirements

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:

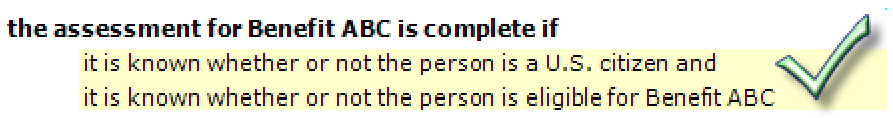

the assessment for Benefit ABC is complete if
the person’s full name is known and
it is known whether or not the person is eligible for Benefit ABC

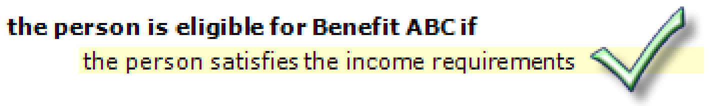

the person is eligible for Benefit ABC if
the person satisfies the income requirements

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:

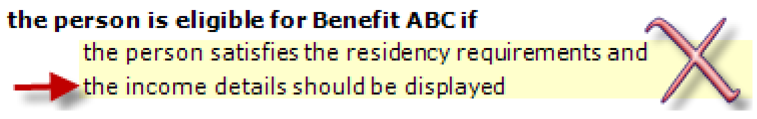

the assessment for Benefit ABC is complete if
it is known whether or not the person is a U.S. citizen and
it is known whether or not the person is eligible for Benefit ABC


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 residency requirements and whether the income details should be displayed.

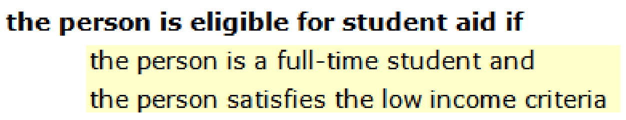


the person is eligible for Benefit ABC if
  the person satisfies the residency requirements and
  the income details should be displayed

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 eligible for a benefit.

2.2.3 NEVER USE SUBSTANTIVE LOGIC RULES AS VISIBILITY ATTRIBUTES

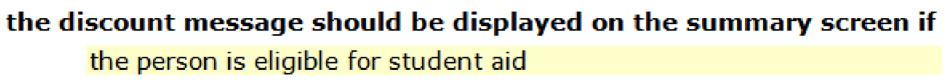
Substantive logic rules should not be used *as* visibility rules, but rather should feed *into* visibility rules. For example, imagine you had the following substantive rule:



the person is eligible for student aid if
  the person is a full-time student and
  the person satisfies the low income criteria

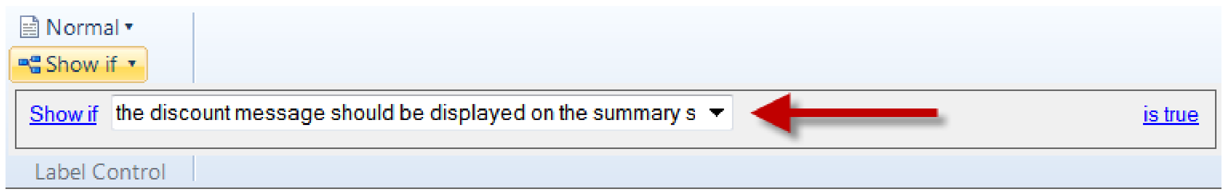
In the interview, imagine you want to display a message at the end if the person is eligible for student aid. You should create a visibility rule and link it to the message on the interview screen.

Visibility rule:

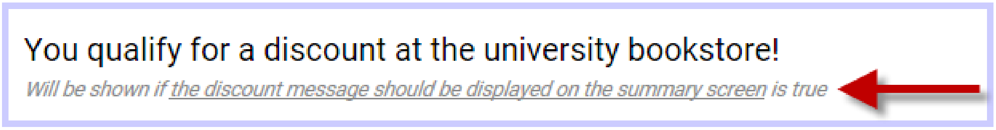


the discount message should be displayed on the summary screen if
  the person is eligible for student aid

Linking the visibility rule to a label on the interview screen:



Show if the discount message should be displayed on the summary s is true



You qualify for a discount at the university bookstore!

Will be shown if the discount message should be displayed on the summary screen is true

Some reasons why it is good practice to create separate visibility rules:

- 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.

Policy Modeling User Guide

Hide and show screen controls
Configure a control to display conditionally:
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Configure_control_display_conditionally.htm

2.3 SPLITTING LOGIC OUT INTO SEPARATE RULES

This will always be a case-by-case assessment for the policy modeler, 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 policy modeler 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) to need to have a single rule which is longer than one page.

Policy modelers with a programming background (as opposed to policy modelers 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 Intelligent Advisor rule, whereas policy modelers 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.
- Cross entity reasoning. Complex cross entity reasoning rules which need to traverse multiple entities within the one rule will likely need to go deeper than Level 3.

2.5 NEGATIONS AS CONCLUSIONS

A misconception among newer policy modelers 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 NON-BOOLEAN ATTRIBUTES AS CONCLUSIONS

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

- Conclusion-only rules, e.g.

  the threshold date = 2020-01-01

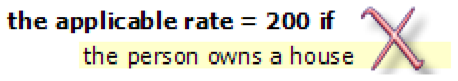
- Calculation rules without conditional premises, e.g.

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

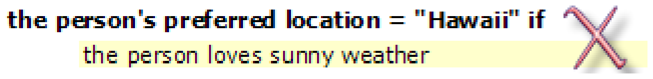
- Rule tables, e.g.

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

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



the applicable rate = 200 if
the person owns a house



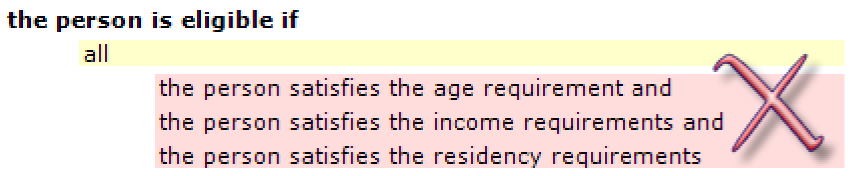
the person's preferred location = "Hawaii" if
the person loves sunny weather

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')

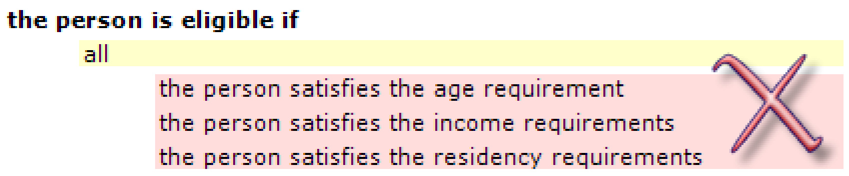
Conjunctions ('and') and disjunctions ('or') should be explicit, not implicit, i.e. where there is 'and'/'or' logic being used, write 'and' and 'or' into the rule. Also, avoid adding extra lines to your rules purely for the purpose of unnecessary grouping operators.

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



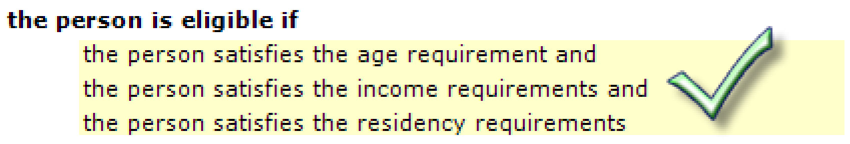
the person is eligible if
all
the person satisfies the age requirement and
the person satisfies the income requirements and
the person satisfies the residency requirements

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



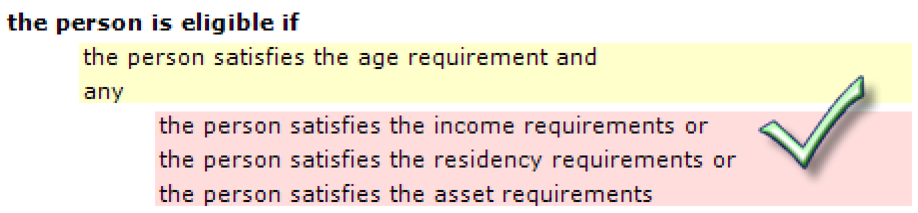
the person is eligible if
all
the person satisfies the age requirement
the person satisfies the income requirements
the person satisfies the residency requirements

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, the rule below requires a grouping operator 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)

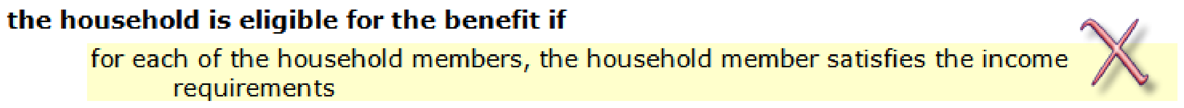
the expiration date = the date 30 days after the start date

More complex functions, and 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 policy modelers 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.

2.8.1.1 TERSE FORM FOR ENTITY FUNCTIONS

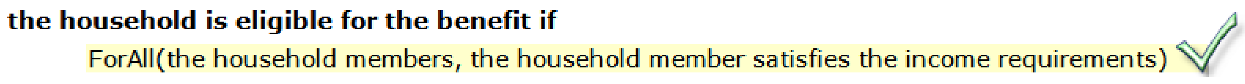
For entity functions it is better to use the terse form. If you use the natural language form and have not defined the entity/relationship correctly, OPM may simply treat the entity function line of your rule as a single Boolean attribute (e.g. "for each of the household members, the household member satisfies the income requirements" in the example below), effectively masking the fact that there is an entity definition error.



the household is eligible for the benefit if

for each of the household members, the household member satisfies the income requirements

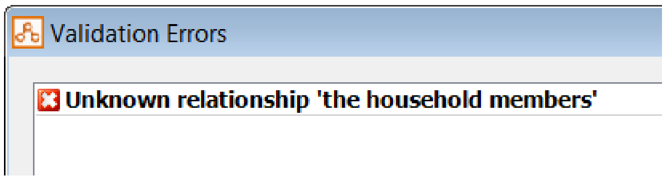
Whereas if you use the terse form, either the rule will be interpreted correctly by OPM (if the entity/relationship has been defined):



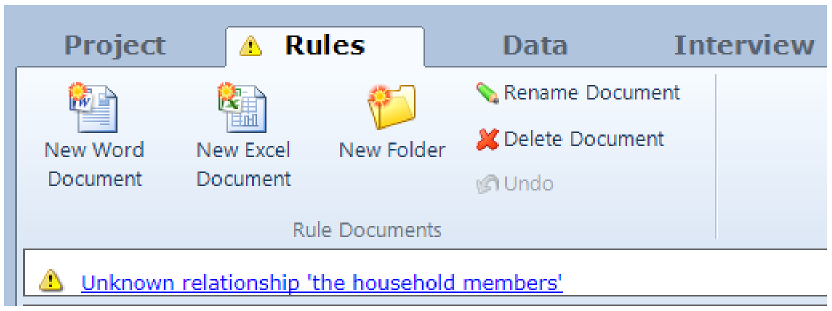
the household is eligible for the benefit if

ForAll(the household members, the household member satisfies the income requirements)

Or you will get a Validation error in the rule document and in the Rules tab of OPM (if the entity/relationship has not been defined properly):



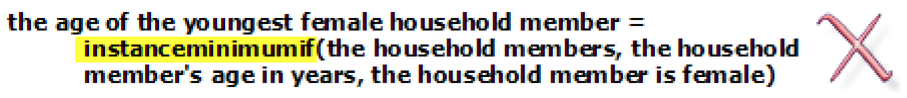
Unknown relationship 'the household members'



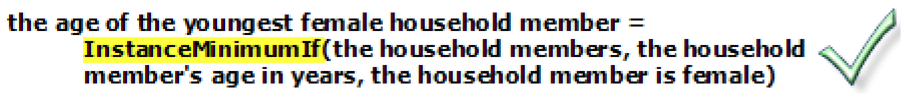
Unknown relationship 'the household members'

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 makes 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)

2.8.3 CURRENT DATE

The CurrentDate function (natural language form: the current date) can be used to get today's date, i.e. the current system date. If you have multiple rules which depend on 'the current date', create a regular date attribute (e.g. 'the date of assessment' or something else appropriate), set that date attribute to the current date using either the natural language or terse form (see below), then use the regular date attribute throughout the rules.

the date of assessment = the current date

the date of assessment = CurrentDate()

Benefits of this approach include:

- **Easier for maintenance.** If you later want the rules to run based on a different assumed interview date (e.g. the start of the current month, a week from today, or some other date) you only have to edit one rule instead of many rules.
- **Easier for testing.** If you want to test or experiment with hypothetical interview dates, you do not need to adjust the system date of your machine or your test environment. You can simply comment out the rule above and make it an input attribute while testing and experimenting.
- **Easier for searching usage.** You can use the Search in OPM to find everywhere 'the date of assessment' is used throughout the policy model. You cannot use the Search to find usage of the CurrentDate function, or indeed usage of any function.

Policy Modeling User Guide

Get the current
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Write_rules_using_dates_and_times/Get_current_date.htm

2.8.4 COMBINING FUNCTIONS

There are many functions available out-of-the-box for your rules. Sometimes the logic you need can be addressed by a single function, sometimes it cannot. If it cannot, consider whether the logic can be addressed by a *combination* of functions.

A mistake that newer policy modelers sometimes make is to assume that if they cannot find a function which directly addresses their specific issue, then the logic cannot be written. This is usually not the case. For example, imagine you need to find the second Monday after the application date. There is no single function for determining the second Monday after a given date. However, there is a function to find the first Monday on or after a given date (NextDayOfTheWeek) and a function to add X number of weeks to a given date (AddWeeks).

the relevant date = AddWeeks(NextDayOfTheWeek(the application date, "Monday"), 1)

Note that the example above assumes that if the application date falls on a Monday, then that Monday counts as the first Monday, i.e. if the application date is the 1st day of the month and that happens to be a Monday, then the relevant date is the 8th of the month. If the source material indicates the application date itself should not be included (i.e. the relevant date is the 15th of the month in this example), then add 2 weeks instead of 1, i.e.

the relevant date = AddWeeks(NextDayOfTheWeek(the application date, "Monday"), 2)

When combining functions, it is okay to use the terse form for some functions and the natural language form for others, especially if it makes the overall rule easier to read, e.g.

the relevant date = AddWeeks(the next Monday on or after the application date, 2)

If combining more than a couple of functions, consider whether you should break out into separate rules to make the logic easier to follow.

Policy Modeling User Guide

Function References
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Function_references/Function_references.htm

2.9 TABLES

2.9.1 WORD TABLE VS EXCEL TABLE

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

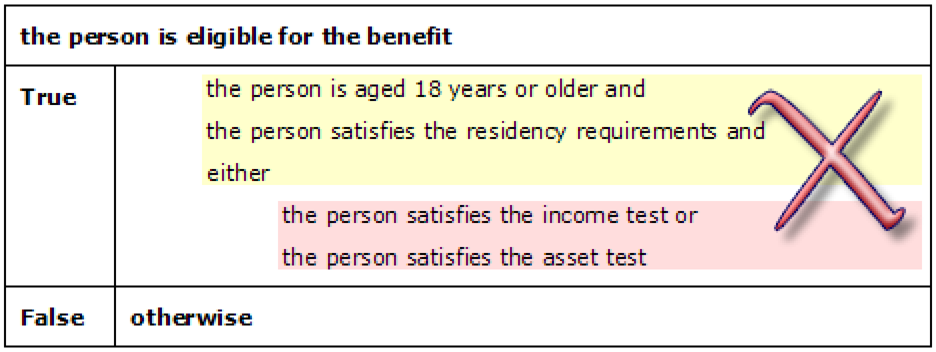
- **Simple logic or complex?** If the logic being modeled is relatively simple and can neatly be expressed in a Word table, consider Word. Many people find Word easier to work with, and so it makes sense to use it unless there is a good reason otherwise.
- **Multiple conclusions from same conditions?** 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 Word tables only allow one conclusion per table. If the logic has 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.

Policy Modeling User Guide

Decide whether to write rules in Word or Excel
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Create_rules/Decide_whether_to_write_rules_in_Word_or_Excel.htm

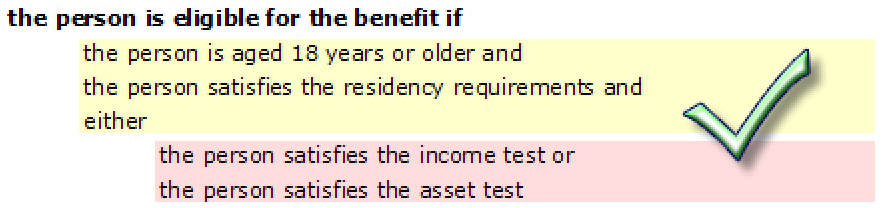
2.9.2 DO NOT CONCLUDE BOOLEANS IN WORD TABLES

Boolean attributes should not be concluded in Word tables.



the person is eligible for the benefit	
True	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
False	**otherwise**

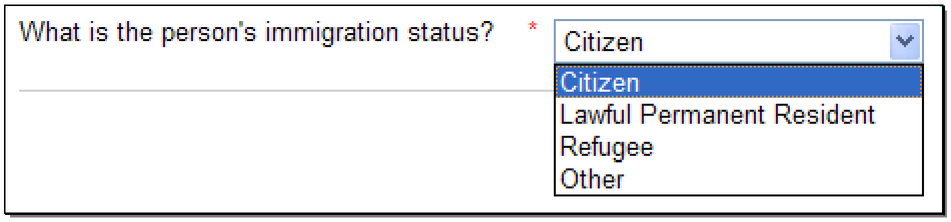
Booleans should be concluded in regular rules, e.g.



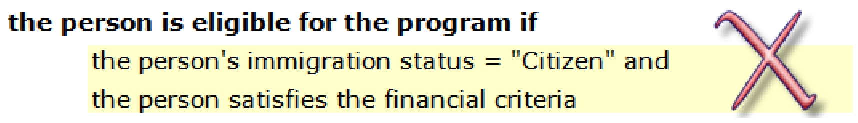
the person is eligible for the benefit if
the person is aged 18 years or older and
the person satisfies the residency requirements and
either
the person satisfies the income test or
the person satisfies the asset test

2.10 BOOLEAN ATTRIBUTE OVER THE TOP OF LIST ITEMS

For some non-Boolean attributes, most commonly text attributes, 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 an interview screen.

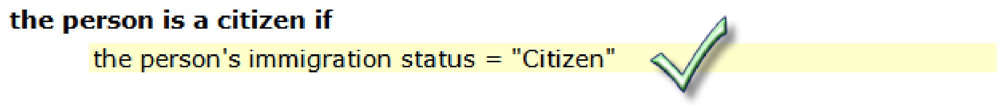

What is the person's immigration status? *
Citizen
Citizen
Lawful Permanent Resident
Refugee
Other

In this situation, generally the non-Boolean attribute should not be used directly throughout the policy model. For example, in general you should not have rules using the non-Boolean in this way:

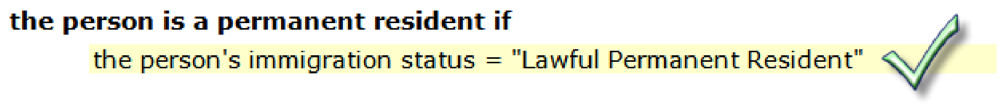


the person is eligible for the program if
the person's immigration status = "Citizen" and
the person satisfies the financial criteria

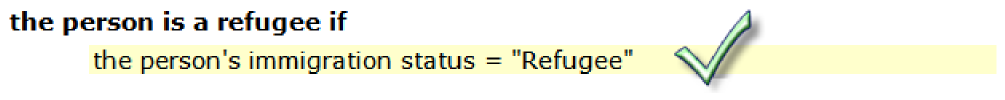
Instead, create a Boolean attribute to sit over each list item, e.g.



the person is a citizen if
  the person's immigration status = "Citizen"

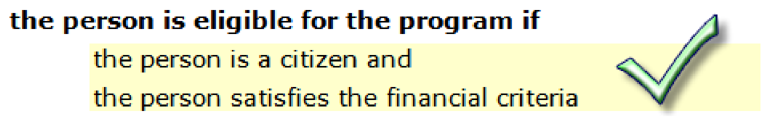


the person is a permanent resident if
  the person's immigration status = "Lawful Permanent Resident"



the person is a refugee if
  the person's immigration status = "Refugee"

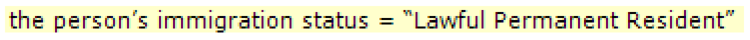
And use the Boolean attribute throughout the policy model, e.g.



the person is eligible for the program if
  the person is a citizen and
  the person satisfies the financial criteria

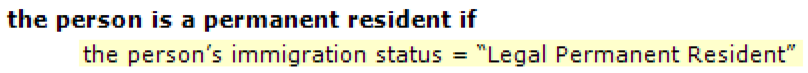
If the "Other" option in the drop-down list is just a catch-all for when none of the other options apply, then you may not need to create a Boolean attribute to sit over it as you may never be specifically referring to that option in the rules. In the example above, selecting "Other" will result in the 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 policy model:



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 the attribute "the person’s immigration status" needed to be changed to “the person’s citizenship status”. If you have used the attribute "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 attribute is a change in the actual meaning, then you may need to edit all the rules in which the attribute is used.

- The advantage of this approach is that it minimizes the number of rules which are touched. 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. You will need to use good judgment in deciding what to do in your policy model.

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 non-Boolean attribute 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 48 contiguous States. Purely for income test purposes, it would be enough 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

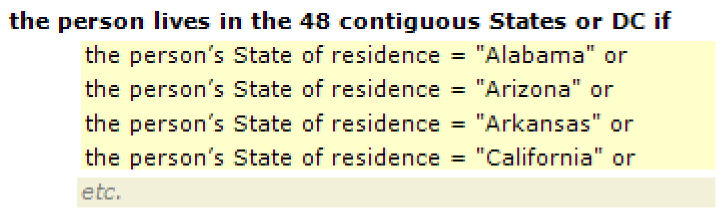
However, one of the business requirements is that the interview must collect the person’s State of residence. Here are a couple of ways you could approach this situation.

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 non-Boolean, 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

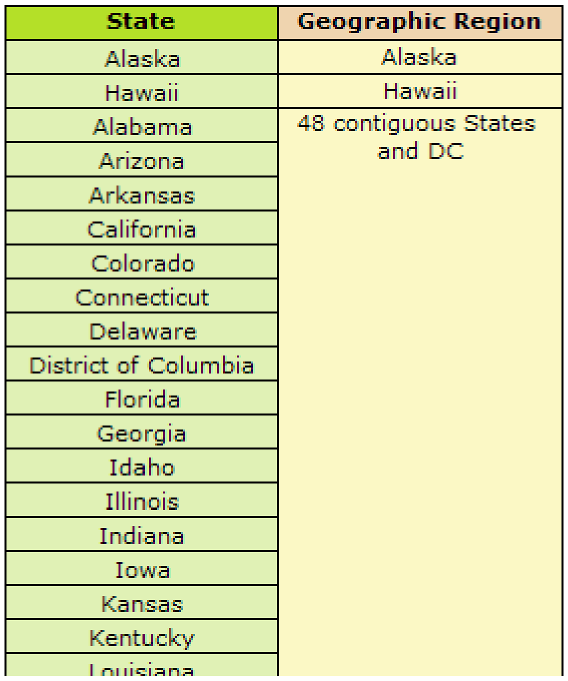
the person's State of residence = "Alabama" or
the person's State of residence = "Arizona" or
the person's State of residence = "Arkansas" or
the person's State of residence = "California" or
etc.

- 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

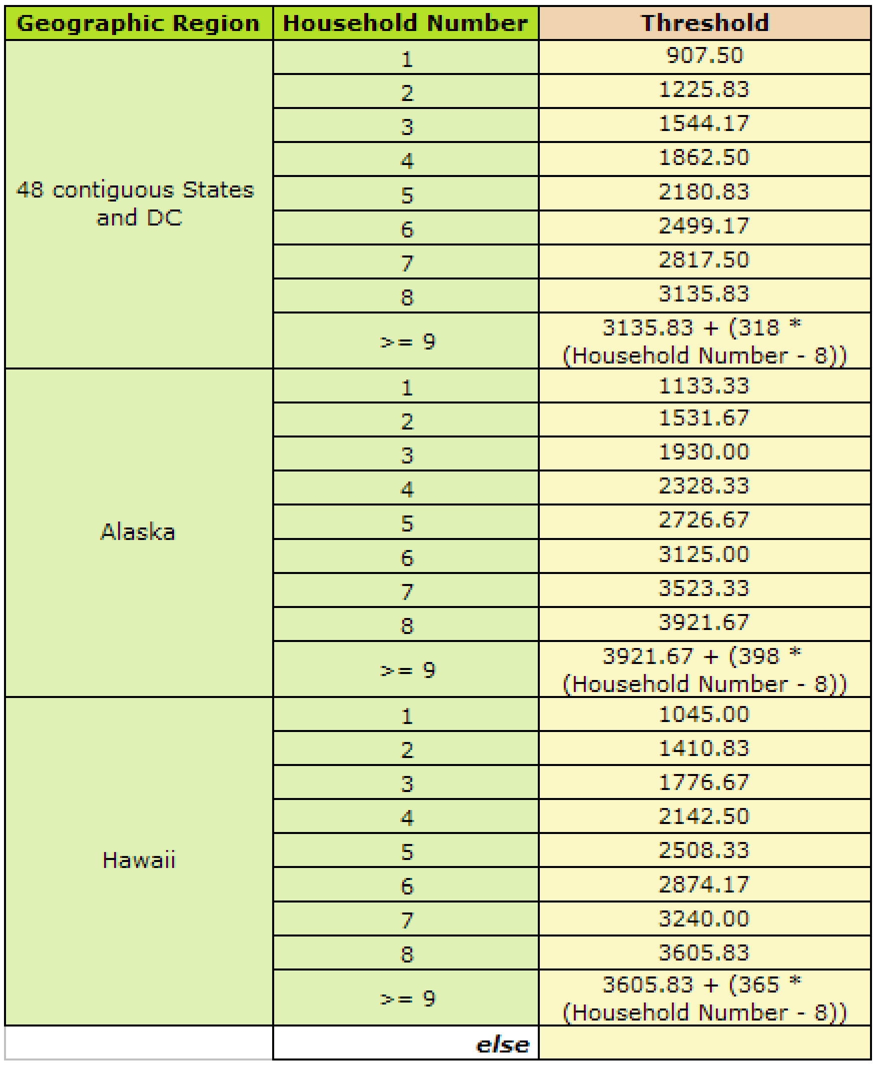
If the 'residence' criteria were primarily going to be used in Excel tables, then here is another approach to consider.

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



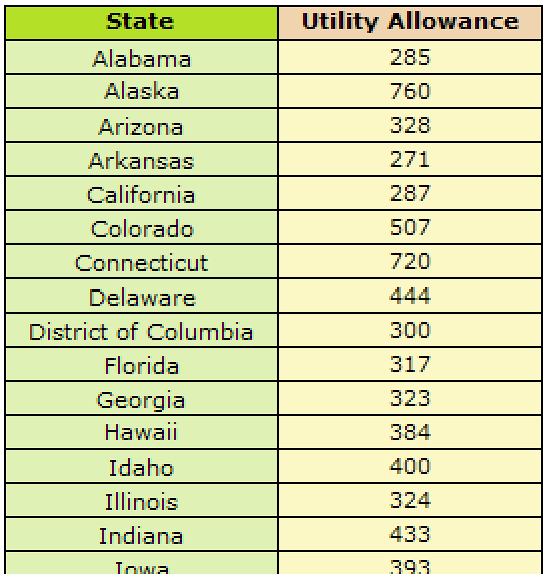
State	Geographic Region
Alaska	Alaska
Hawaii	Hawaii
Alabama	48 contiguous States and DC
Arizona	
Arkansas	
California	
Colorado	
Connecticut	
Delaware	
District of Columbia	
Florida	
Georgia	
Idaho	
Illinois	
Indiana	
Iowa	
Kansas	
Kentucky	
Louisiana	

- 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 * (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 * (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 * (Household Number - 8))
	else	

- 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

Policy Modeling User Guide

Make your Excel rules easier to understand
Simplify rule table layout by merging cells
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Write_rules_in_Excel/Simplify_rule_table_layout_by_merging_cells.htm

CHAPTER 3. ENTITIES

There are pros and cons to using entities and relationships in your policy model. The significant pro is that it allows much more complex logic to be authored in the policy model. The con is that, compared to Global-only policy models, policy models with entities and relationships are more complicated 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 requires entity reasoning, then you should add the appropriate entities and relationships. The benefit you will gain, in terms of the logic that can be written into the policy model, 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, and whether future extensions to the policy model 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. A typical example is a household where you need to collect data about each household member and do a calculation over the set of household members. For example, imagine the source material said:

- The household is eligible for the Family Benefit if the annual 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 need entities. Specifically, you need an entity called 'the household member' with 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 many other things, e.g. the subsidiary company, the product, the vehicle, the asset, etc. Also, Intelligent Advisor logic allows for entities to be nested below other entities; entities do not all have to be immediately contained by Global.

Policy Modeling User Guide

Create an entity
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Entities/Create_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 product, the asset, etc.

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 explanations.

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

For tips on naming entity level attributes, see section 4.1.3.

Policy Modeling User Guide

Give an entity a name
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Entities/Give_an_entity_a_name.htm

Choose relationship text
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Relationships/Choose_relationship_text.htm

CHAPTER 4. ATTRIBUTES

4.1 ATTRIBUTE TEXT

4.1.1 BOOLEAN ATTRIBUTES

The Policy Modeling User Guide has excellent advice on choosing appropriate attribute text, including further detail on most of the principles below, so the Policy Modeling User Guide 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 the interview, 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 interviews.
- 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 policy model. If you picked the first one, your policy model 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 ("&")
- Do not use the natural language syntax of operators as part of regular attribute text
    - For example, you should not have a Boolean attribute such as "the person's phone number is currently known". 'Is currently known" is the natural language syntax for the 'currently known' operator. A premise such as "the person's phone number is currently known" should only appear in a rule where it is referring to the 'currently known' operator acting on the non-Boolean attribute "the person's phone number". If you need a Boolean, consider using something such as "the person's phone number has been collected".

4.1.2 NON-BOOLEAN ATTRIBUTES

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

A good way to test in your head if your non-Boolean attribute 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?

Note that if the non-Boolean attribute has personal substitution enabled, the default question form is "who is…", e.g. *Who is* the person? *Who is* the household member? *Who is* the applicant?

✘ **Example of a badly worded non-Boolean attribute: "person income"**

This is a very badly worded non-Boolean attribute. The question form of this attribute 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 attribute:

- The definite article ("the") has not been used. This attribute 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.

Policy Modeling User Guide

Choose attribute text
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Choose_attribute_text.htm

Basic English grammar
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Basic_English_Grammar.htm

4.1.3 ENTITY LEVEL ATTRIBUTES

Entity level attributes (Booleans and non-Booleans) 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	**Acceptable.** Attribute text contains the entity name.
✓ the household member's birthdate	**Acceptable.** Attribute text contains the entity name.
✓ the date of birth of the household member	**Acceptable.** Attribute text contains the entity name.
✓ the birthdate of the household member	**Acceptable.** Attribute text contains the entity name.
✗ the date of birth	**Incorrect.** Attribute text does not contain the entity name.
✗ the birthdate	**Incorrect.** Attribute text does not contain the entity name.
✗ the person's date of birth	**Incorrect.** Attribute text does not contain the entity name.
✗ the date of birth of the member of the household	**Incorrect.** 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?

Here are some reasons why entity level attributes should contain the name of the entity:

- If the entity attribute contains the name of the entity, the attribute will *automatically* exist within the correct entity in the data model.
- If the entity attribute contains the name of the entity, 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 policy models with entities.
- If you are using cross entity reasoning and inferred relationships, you *must* have the name of the entity in entity attributes to use aliasing. Aliasing is required for some types of cross entity logic, so if you do not include the name of the entity in entity attributes then you are effectively limiting the extent to which you can use a very powerful branch of logic.

Policy Modeling User Guide

Assign an attribute to an entity
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Entities/Assign_attribute_to_entity.htm

Work with complex entity logic
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Write_rules_using_ent_and_rel/Work_with_complex_entity_logic.htm

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, or visibility 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 child care locations should be displayed on the interview screen" (conclusion attribute for a screen visibility rule)

4.2 ATTRIBUTE PARSING

One of the major benefits of using Intelligent Advisor is the natural language rule authoring capability. If you do not use appropriate attribute text and correct parsing, then you are ignoring one of the distinctive features of Intelligent Advisor and reducing the benefits that can be gained by using Intelligent Advisor in the first place.

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

- Rule documents
- Interview screens
- Explanations
- Forms

4.2.1 BOOLEAN ATTRIBUTES

A Boolean attribute is something for which you are looking for a yes/no answer, e.g. Did the person apply for a loan?

A full parse for a Boolean attribute consists of:

- the positive form, e.g. the person **applied** for a loan
- the negative form, e.g. the person **did not apply** for a loan
- the question form, e.g. **did** the person **apply** for a loan
- the uncertain form, e.g. it is uncertain whether the person **applied** for a loan

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 "apply".

Policy Modeling User Guide

Change the sentence forms for an attribute
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Change_sentence_forms_for_attribute.htm

4.2.2 IDENTIFYING AND FIXING PARSING ISSUES WITH BOOLEAN ATTRIBUTES

Policy modelers must learn how to identify and fix parsing issues. For most 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 that matches a verb from the verbs list.
- There is *no* word in the sentence that 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 that match a verb from the verbs list, the parser will use the first one as the operative verb, unless the policy modeler 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. it is uncertain whether the person submitted the tax **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. it is uncertain whether the person **submitted** the tax return

As you can see, the positive and uncertain forms are the same in both parses. However, the negative and question forms make it is quite clear which verb is the correct operative verb for the 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 Policy Modeling User Guide has excellent information about parsing, so please refer to the articles for further advice.

Policy Modeling User Guide

Change the sentence forms for an attribute
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Change_sentence_forms_for_attribute.htm

4.2.2.2 NO KNOWN VERBS IN THE BOOLEAN ATTRIBUTE TEXT

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

If it is simple to re-word the sentence to use an existing verb, then it is pragmatic 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, but keep in mind it means you now have a customized verb list for the project. If you copy-and-paste rules from the existing project into another project, you will need to check if those rules contain a custom verb, and if so, add that verb to the new project.

Policy Modeling User Guide

Configure the list of recognized verbs
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Configure_list_of_recognized_verbs.htm

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. it is uncertain whether the person **can 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. it is uncertain whether the person cannot **cannot 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 NON-BOOLEAN ATTRIBUTES

A non-Boolean attribute is something for which you are looking for a value such as a currency value, a date value, a number value or a text value, e.g. What is the person's monthly 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 non-Boolean attributes is much simpler than parsing Booleans. Non-Boolean attributes do not have operative verbs because non-Booleans are not full sentences; non-Boolean are phrases. This means non-Boolean attributes do not have negations. However, they do have question forms.

The vast majority of non-Boolean attributes have the 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 non-Boolean attributes:

- **When the non-Boolean attribute is the name substitution attribute.** When name substitution is set up correctly, the auto-generated question form will be "who is...?" rather than "what is...?"
- **When the non-Boolean attribute is a plural.** This rarely occurs in real policy models, but if it does you can use free form text to change the question form on the interview screen to "what are …?". You can also change the Sentence, Uncertain and Unknown forms in the Attribute Editor in the Data tab.

4.3 SUBSTITUTION

Substitution is a great way to personalize the interview experience. For Global attributes, substitution is a 'nice to have'. For entity level attributes, substitution is required to make it clear to the end user which entity instance the interview is asking about. 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.

Generally, the type of substitution you should implement first is **substitution in attribute text**. This is also 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 non-Boolean attributes 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 policy model, e.g. interview screens, explanations, the Debugger, and test cases.

4.3.1 SUBSTITUTION IN ATTRIBUTE TEXT

Substitution for attribute text should be set up in the Data tab. By far the most common use of attribute text substitution is name substitution, which is the example used here.

When done properly, all attributes at runtime, whether in an interview 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 hardcoding of substitution in screens is required for this. This is implemented purely through the Data tab and the substitution automatically flows through to any screen controls which use the auto-generated question form of the attribute.

Note that substitution for entity identifiers is set up automatically when the entity is created. However, it is still important to understand how substitution works and how to set it up manually. You may need to set up substitution on other attributes, or debug substitution that is not working correctly.

Policy Modeling User Guide

Personalize attribute text
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Personalize_attribute_text.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 Policy Modeling User Guide article below for instructions.

A few things to note about 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 sufficiently clear and unambiguous, there is no need to add further substitution in screen headings and labels.
- 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.
- The substitution attribute must have a name. So if you want to substitute the value of "the person's age in years" into a screen label, you must add a name, e.g. PersonAgeYears or person_age_years, and use the name in the substitution.

Policy Modeling User Guide

Show attribute values on screens
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Show_attribute_values_on_screens.htm

4.3.3 SUBSTITUTION IN ATTRIBUTE TEXT WITH SECOND PERSON SENTENCE GENERATION

Generally, attributes should be written in third person (‘the person is eligible’) rather than second person (‘you are eligible’). However, sometimes you may want to display attributes in second person at runtime, e.g. ‘Are you a student?’ (second person) rather than ‘Is the person a student?’ (third person). You can do this automatically for all attributes by enabling a few simple settings in OPM, and without having to re-write attribute text or use text-override on interview screens.

Benefits of using second person sentence generation in OPM, rather than writing the attributes themselves in second person are:

- When attributes are written in third person, you just need to enable/disable settings in OPM to switch between second person, name substitution, and third person. This is relatively quick and simple to implement, and to modify if you change your mind about how the attributes should be displayed at runtime.
- The source material (legislation, regulations, policy, etc.) is often in third person, so the attribute text more closely matches the source material.

If the interview will only ever be needed in second person, then you may consider writing the attributes in second person. However, just be aware that the attributes will all need editing if the interview later needs to be displayed in third person or with name substitution.

Policy Modeling User Guide

Display interview text in second person form
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Display_interview_text_in_second_person_form.htm

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.

Policy Modeling User Guide

Personalize attribute text

Substitute a pronoun for a text attribute
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Attributes/Substitute_a_pronoun_for_text_attribute.htm

Collect the gender of a person

https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Collect_gender_of_person.htm

CHAPTER 5. SCREENS

5.1 CREATE INTERVIEW SCREENS FOR INPUT ATTRIBUTES

If a policy model is intended to be run interactively by an end user (e.g. a person interacting with an online benefit screening tool), then generally all input attributes in the policy model should be attached to interview screens.

It is true that a policy model can be run interactively without creating any screens. Automatic screens will be generated, with one question per screen. All screens will have the default screen title (usually the attribute text), and all attributes will have the default input type. While technically this will work, configuring interview screens rather than relying on auto-generated screens provides much greater control over the content of the screens and the flow of the interview. This allows for a much nicer and user-friendly interface.

Policy Modeling User Guide

Create a screen
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Create_a_screen.htm

5.1.1 GREATER CONTROL OVER SCREEN CONTENT

When interview screens are created, the policy modeler configure many different options on each screen. For example:

- Group specific questions together on a particular interview screen
- Control whether each question is optional or mandatory
- Set the input type of each question, e.g. drop-down list, free-text field, radio buttons, list box, etc.
- Specify default values for each question if desired
- Specify headings, subheadings and notes for each screen
- Specify alternate question text (where appropriate, see section 5.3)
- Add images to screens
- Decide what happens when the end user exists the screen, e.g. continue the interview, save data, attach files, redirect to a different URL

Policy Modeling User Guide

Screens
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Screens.htm

Overview of designing an interview
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Design_interviews/Overview_of_designing_an_interview.htm

5.1.2 GREATER CONTROL OVER INTERVIEW FLOW

Creating your own screens, rather than relying on the auto-generated screens, means you can define a default screen order for the interview flow. When doing so, all input attributes should be attached to interview screens. Any input attributes not attached to interview screens will be displayed on automatic screens at the end of the interview (which may not be a sensible place from the end user's perspective).

Policy Modeling User Guide

Control screen navigation
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Design_interviews/Control_screen_navigation.htm

Set the goals for an interview
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Design_interviews/Set_the_goals_for_an_interview.htm

Allow screens to be visited in any order
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Design_interviews/Allow_screens_in_any_order.htm

5.1.3 ACCEPTABLE EXCEPTIONS

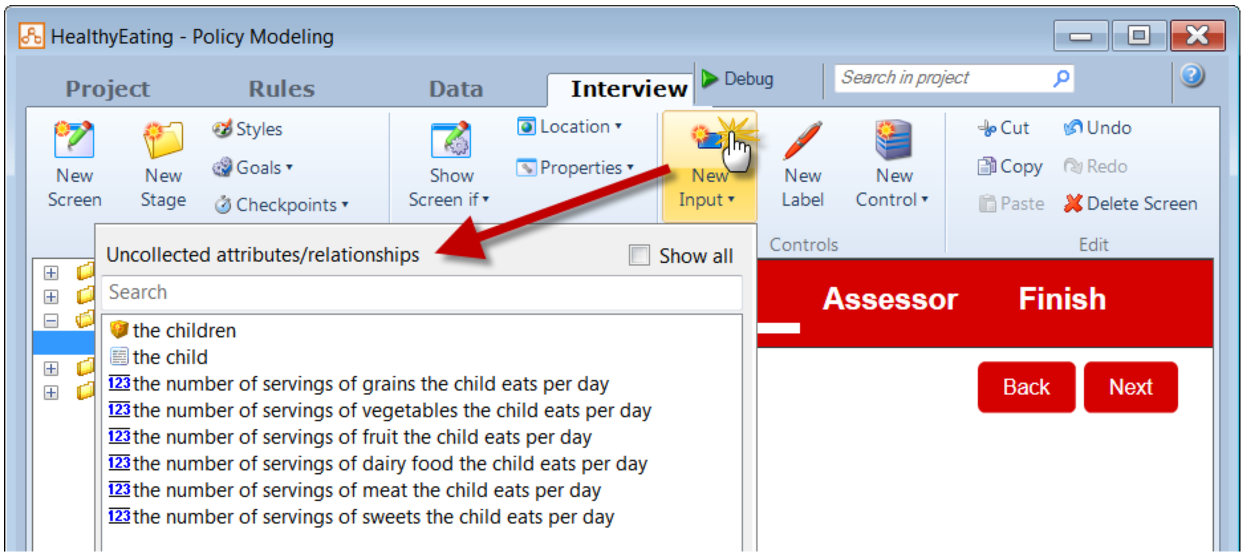
There are a couple of exceptions to the rule that all input attributes should be attached to interview screens:

- The policy model will not be used interactively, i.e. there will not be any end users interacting with the policy model directly. For example, if the policy model is integrated with another system and the purpose of the policy model is to receive data from that system, run the data through the rules, and send the result back to the system.
- There is external data which will be pre-loaded into the interview session at runtime. Even if the policy model will be used interactively, input attributes which will always be set by external reference data do not need to be attached to interview screens. If the input attribute will sometimes be set by external data and sometimes be asked of the end user, then it should be on an interview screen.

- The policy model is just for demonstration purposes. A polished demonstration policy model *will* have all input attributes attached to screens if it is a demonstration of a policy model intended to run interactively, e.g. an online benefit screening tool. However, a demonstration should not need to be developed to the same standard as a production implementation, therefore attaching all input attributes to interview screens is good guidance for demos rather than a strict rule.

5.1.4 IDENTIFYING INPUT ATTRIBUTES NOT ATTACHED TO SCREENS

By default, the New Input drop-down window shows all input attributes not collected on screens.



the children
the child
the number of servings of grains the child eats per day
the number of servings of vegetables the child eats per day
the number of servings of fruit the child eats per day
the number of servings of dairy food the child eats per day
the number of servings of meat the child eats per day
the number of servings of sweets the child eats per day

Policy Modeling User Guide

Add questions to screens
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Add_questions_to_screens.htm

5.2 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 with out-of-the-box functionality to address this. Here are some examples.

- **Minimize scrolling.** Generally, you should avoid putting so many questions on a single interview screen that the end user needs to scroll around to complete the screen.
  - When collecting entities, consider using the Tabular setting rather than Portrait. This usually results in less scrolling for the user.

  - Where appropriate, use containers for more efficient screen layout. Containers allow for side-by-side screen controls. This can reduce the length of the screen, and thus reduce scrolling. See section 5.4 for an example.

  - Sometimes it may be appropriate to put a larger number of questions on a single interview screen. For example, if it is a very short interview and it makes more sense for the end user to complete it all on one screen. Or if you have dynamic visibility on some attributes meaning that the average end user will not be presented with too many questions at once anyway.

- **Reduce the number of clicks.** Aim to reduce the number of clicks and keystrokes required by the user to complete the screen. For example:

  - **Single-click selection for Booleans.** For Booleans, use an input option which only requires a single click to select, e.g. radio buttons. With radio buttons, the user has one click to select, i.e. click on Yes or No. Whereas with a traditional 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 interview screens.

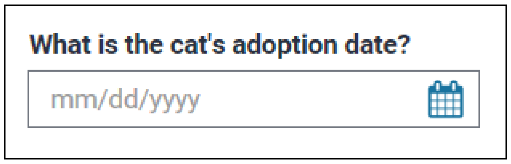
  - **Drop-down lists or radio buttons for non-Boolean attributes.** For non-Boolean attributes, where appropriate create lists for the end user to select from. These lists may be displayed in a variety of ways, such as drop-down lists, radio buttons, or images. The key point is not to require the end 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 list is anytime where the response to the question is not something from a pre-defined set of options, e.g. asking for the person's name, date of birth or income.

  - **Default values for attributes.** Use default values for fields which have a high likelihood of a common answer across end 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 on interview screens.

- **Use hint text.** Date attributes should have hint text so the end user knows the expected date format. Without hint text, the end user may just guess the date format and then either get an error message or enter an incorrect date, e.g. does 12/01/2016 mean 12 January 2016 or December 1, 2016?

  - While using the calendar input type reduces the likelihood of an incorrect date, it is still recommended that you add hint text, e.g.


What is the cat's adoption date?

mm/dd/yyyy

Policy Modeling User Guide

Provide answer values
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Provide_answer_values.htm

Change control layout for a screen
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Change_control_layout_for_a_screen.htm

Dynamically display controls on an interview screen
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Dynamically_display_controls_on_screen.htm

5.3 APPROPRIATE USE OF FREE FORM TEXT

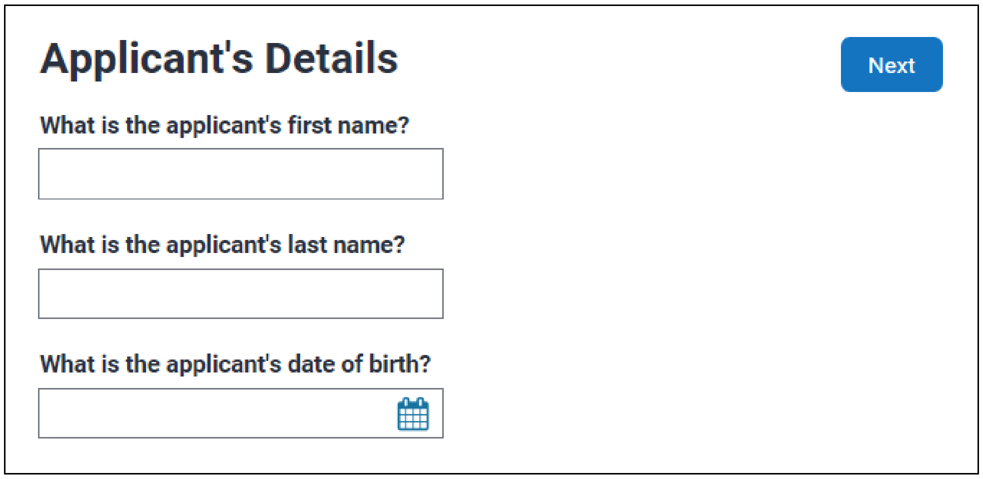
Generally, minimize use of ‘free form’ text for question text in a screen control. If the attributes have been parsed correctly, the auto-generated sentence from the parser should be sufficient. See section 4.2 for information about attribute parsing.

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 an interview screen that collects entity instances
- On an interview screen that collects a series of similar items

5.3.1 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.



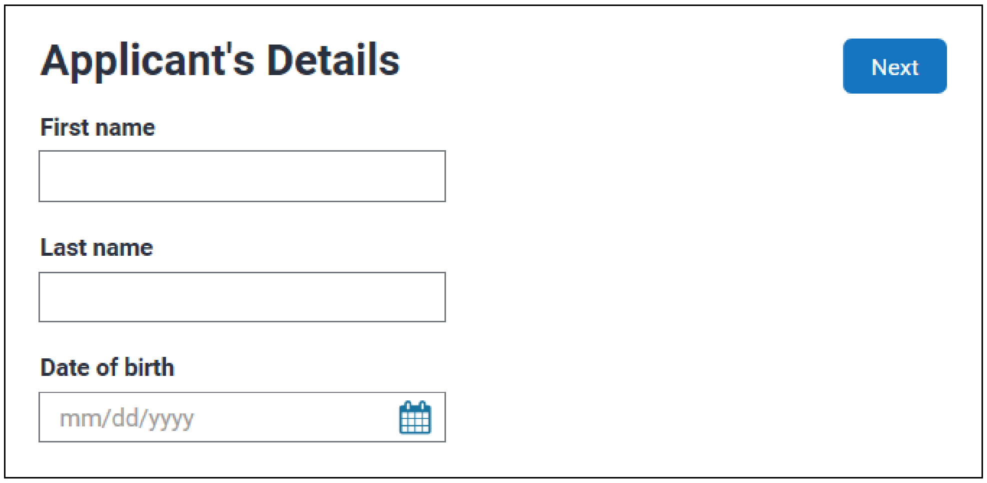
Applicant's Details

What is the applicant's first name?

What is the applicant's last name?

What is the applicant's date of birth?

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.



Applicant's Details

First name

Last name

Date of birth

mm/dd/yyyy

The other consideration is that a 'general details' screen like this in any other application (i.e. not Intelligent Advisor) 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.

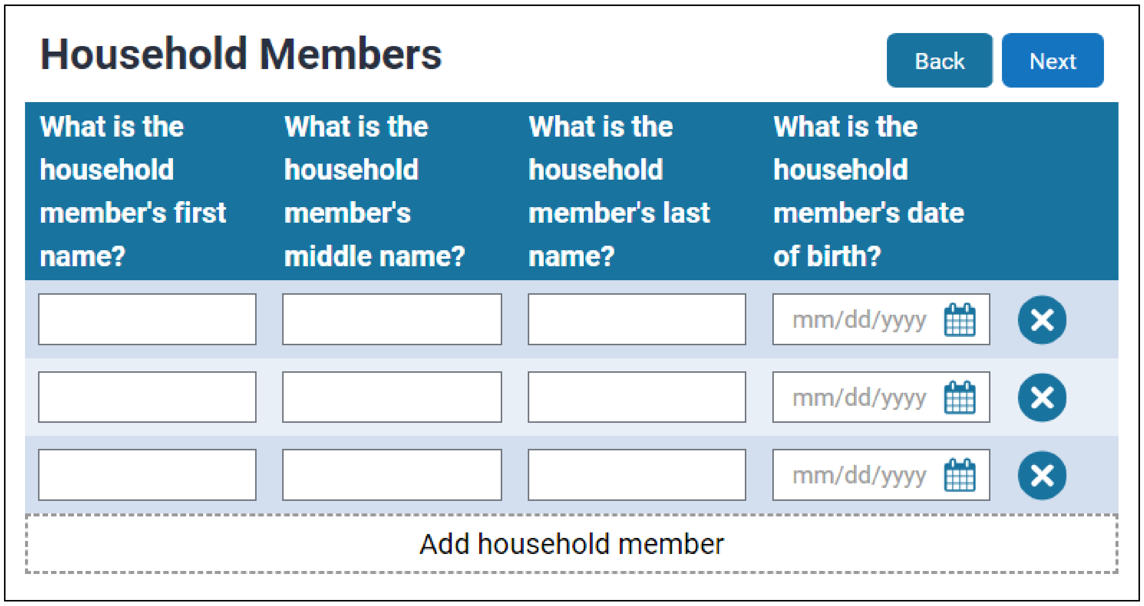
Policy Modeling User Guide

Change the text of an interview question
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Change_text_of_interview_question.htm

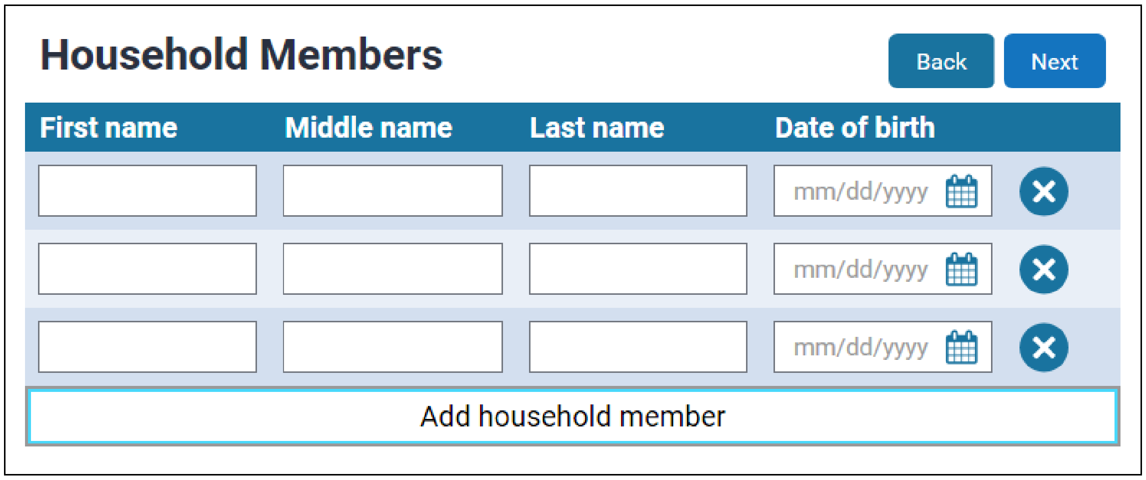
5.3.2 ENTITY COLLECT SCREEN

The most concise way to collect entity instances is with the Tabular layout. When doing so, it is more user-friendly to use shortened free form text rather than the full question text. See examples below.

Entity collection with the full question text:


Household Members
Back
Next
What is the household member's first name?
What is the household member's middle name?
What is the household member's last name?
What is the household member's date of birth?
mm/dd/yyyy
mm/dd/yyyy
mm/dd/yyyy
Add household member

Entity collection with shortened free form text:



First name	Middle name	Last name	Date of birth	
			mm/dd/yyyy	
			mm/dd/yyyy	
			mm/dd/yyyy	

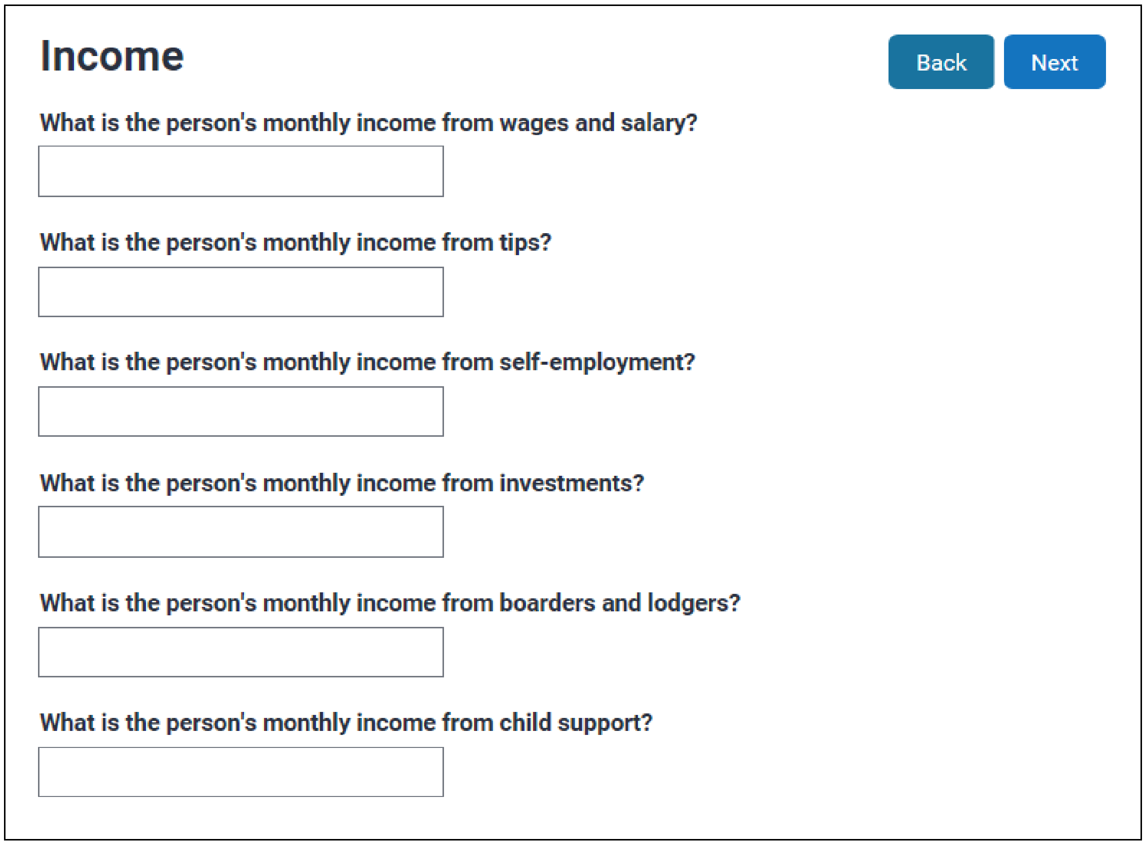
Policy Modeling User Guide

Collect entity instances
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Collect_entity_instances.htm

Change the text of an interview question
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Change_text_of_interview_question.htm

5.3.3 COLLECTING A SERIES OF SIMILAR THINGS

Sometimes you will have question screens that collect a series of similar items, 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.



Income

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

What is the person's monthly income from tips?

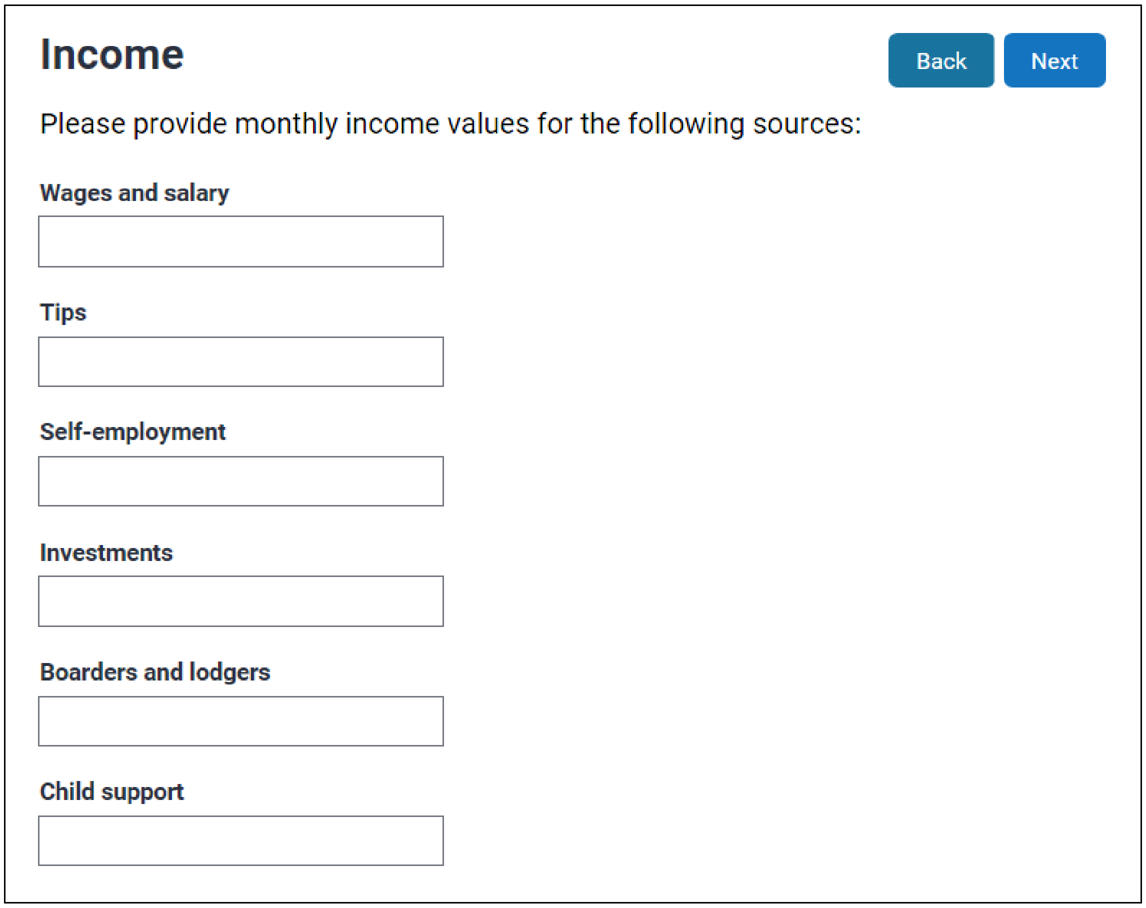
What is the person's monthly income from self-employment?

What is the person's monthly income from investments?

What is the person's monthly income from boarders and 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, use free-form text and include a label at the top to add context, e.g.



Income

Please provide monthly income values for the following sources:

Wages and salary

Tips

Self-employment

Investments

Boarders and lodgers

Child support

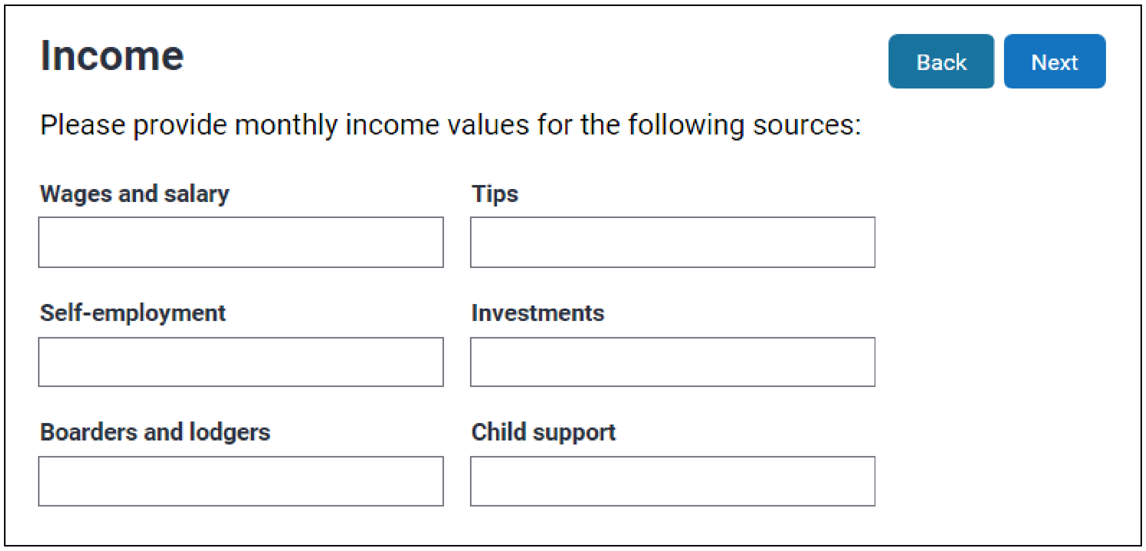
The free-form text does not detract from the meaning of the questions, but rather makes the screen easier to read. The minor additional work and maintenance on this screen is justified by the improvement in usability.

Policy Modeling User Guide

Change the text of an interview question
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Change_text_of_interview_question.htm

5.4 USE CONTAINERS FOR MORE EFFICIENT SCREEN LAYOUTS

By default, each screen control is on a new line as per the screenshots in section 5.3.3 above. Where appropriate, use screen containers to lay out screen controls side by side, e.g.

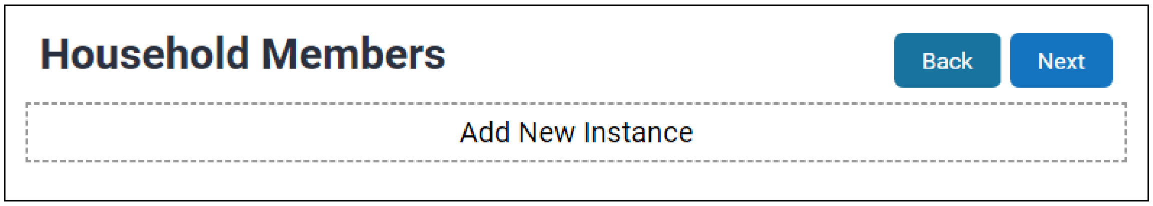


Policy Modeling User Guide

Change control layout for a screen
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Screens/Change_control_layout_for_a_screen.htm

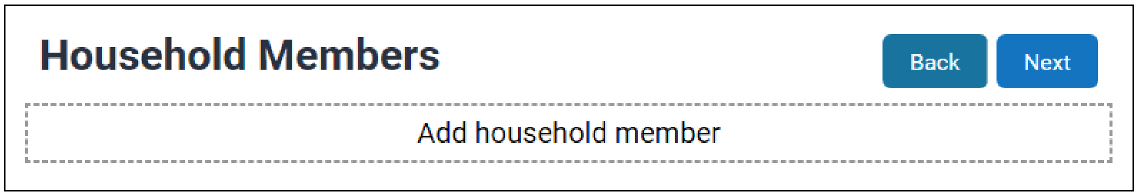
5.5 CONFIGURING THE ADD INSTANCE BUTTON TEXT ON ENTITY COLLECT SCREENS

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



The button text can easily be configured in the Interview tab for each 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 entity 'the household member', consider using 'Add household member' as the button text, e.g.

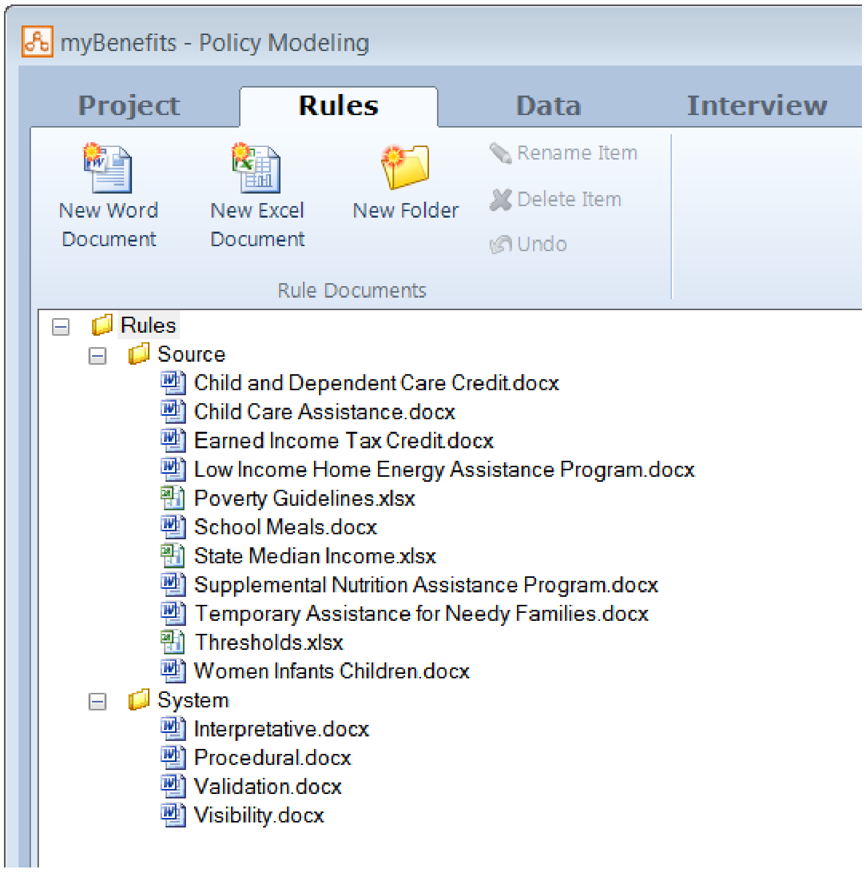


CHAPTER 6. ORGANIZING CONTENT

6.1 RULE FOLDERS AND FILES

6.1.1 CREATE LOGICAL SUBFOLDERS IN WHICH TO STORE DIFFERENT RULE DOCUMENTS

Folders and subfolders can be used in Policy Modeling to organize rule files.



```
Rules
  Source
    Child and Dependent Care Credit.docx
    Child Care Assistance.docx
    Earned Income Tax Credit.docx
    Low Income Home Energy Assistance Program.docx
    Poverty Guidelines.xlsx
    School Meals.docx
    State Median Income.xlsx
    Supplemental Nutrition Assistance Program.docx
    Temporary Assistance for Needy Families.docx
    Thresholds.xlsx
    Women Infants Children.docx
  System
    Interpretative.docx
    Procedural.docx
    Validation.docx
    Visibility.docx
```

Create subfolders to help organize content, e.g. add sub-folders, sub-subfolders, etc. in the Rules tab to organize the rule documents.

Policy Modeling User Guide

Organize rule files into folders
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Organize_rules/Organize_rule_files_into_folders.htm

6.2 RULE DOCUMENTS

6.2.1 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. Policy modelers should use heading levels to organize the rules in their rule documents. The Policy Modeling toolbar has some built-in heading levels, but you are not restricted to these. You can also use the regular Microsoft Word heading levels.

6.2.2 TABLE OF CONTENTS

As the rules are authored in Microsoft Office documents, there are features of the Microsoft Office applications 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 6.2.1), 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 Microsoft Word. Microsoft Word 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 Microsoft Word Help if you need assistance generating a Table of Contents.

Note that if a rule document has a Table of Contents, it may take a little longer to save and validate if the Table of Contents is in view when validating. You can easily avoid this issue by scrolling the Table of Contents out of view before validating the document.

6.2.3 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-Intelligent Advisor style.

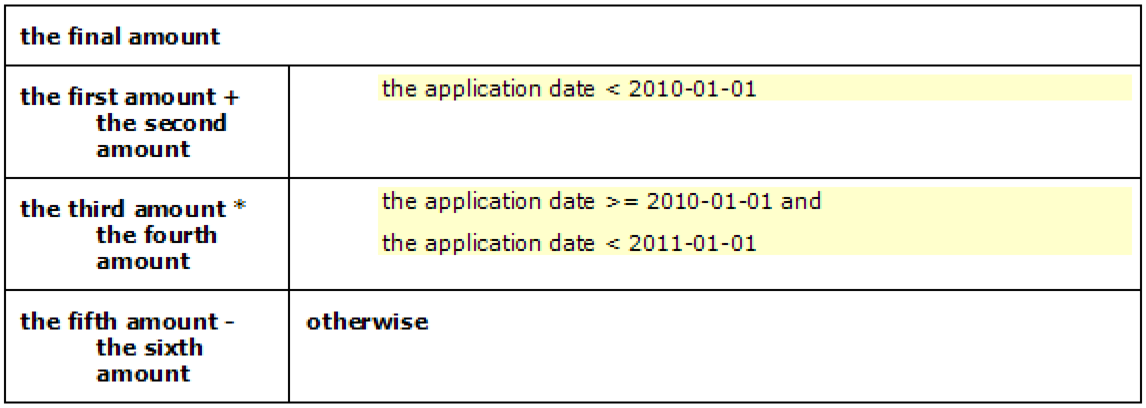
6.2.4 ADJUSTING RULE TABLE COLUMN WIDTH

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.

Generally, you should keep the default table width for Word rules, but adjust as required for Excel.

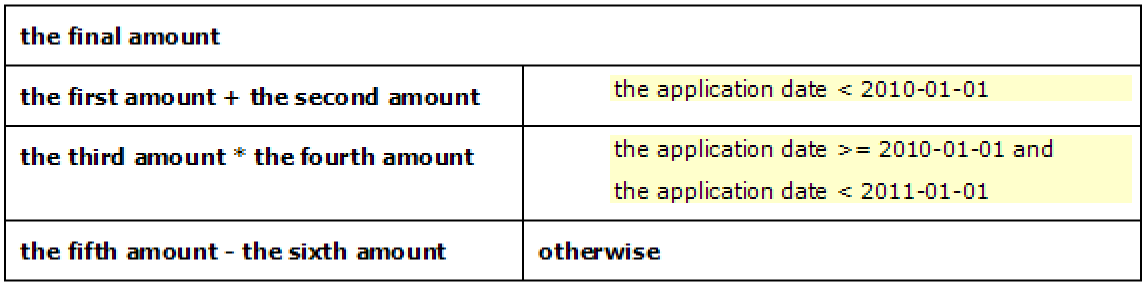
Word rule tables

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.



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

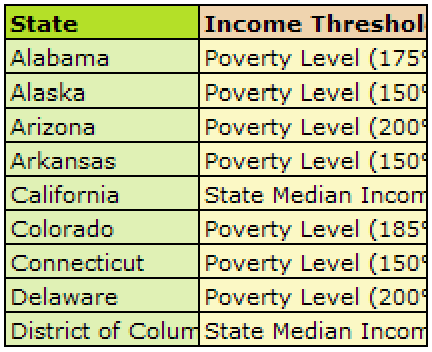
In the example rule table below, the column widths have been adjusted to balance the contents on either side, which makes the logic easier to read. Also, the table takes up less space on the page, and looks neater and tidier in the document.



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

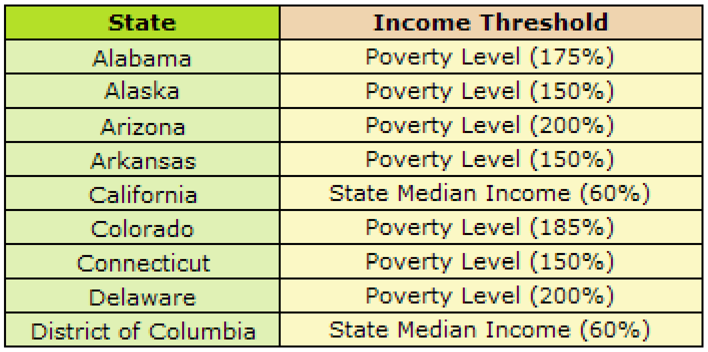
Excel rule tables

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



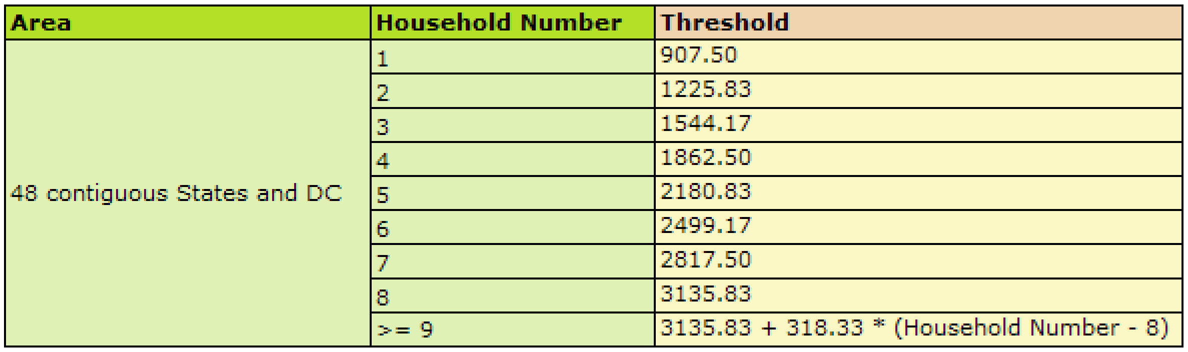
State	Income Threshol
Alabama	Poverty Level (175
Alaska	Poverty Level (150
Arizona	Poverty Level (200
Arkansas	Poverty Level (150
California	State Median Incon
Colorado	Poverty Level (185
Connecticut	Poverty Level (150
Delaware	Poverty Level (200
District of Colum	State Median Incon

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



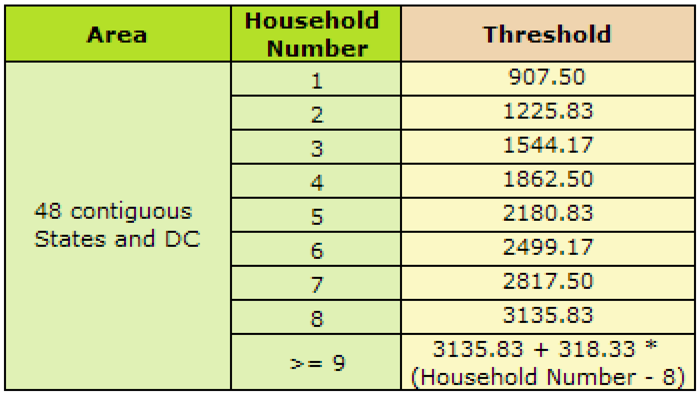
State	Income Threshold
Alabama	Poverty Level (175%)
Alaska	Poverty Level (150%)
Arizona	Poverty Level (200%)
Arkansas	Poverty Level (150%)
California	State Median Income (60%)
Colorado	Poverty Level (185%)
Connecticut	Poverty Level (150%)
Delaware	Poverty Level (200%)
District of Columbia	State Median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

6.2.5 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:

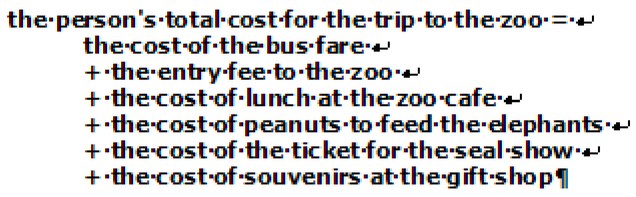
the person's total cost for the trip to the zoo = the cost of the bus fare + the entry fee to the zoo + the cost of lunch at the zoo cafe + the cost of peanuts to feed the elephants + the cost of the ticket for the seal show + the cost of souvenirs at the gift shop

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 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¶
```

6.2.6 ADDING COMMENTS TO RULE DOCUMENTS

Just as programmers may be encouraged to put comments in their code, policy modelers 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 policy modelers, as well as for the original policy modeler if they have not worked on that part of the policy model in a long time.

The OPM style 'Blank Line' may be used for adding notes in rule documents. Regular MSWord styles such as 'Normal' may also be used as OPM ignores all non-OPM styles.

6.2.7 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 validate, but just that it would be usual for it to be appropriate to have a single rule document that long.

Here are some reasons for avoiding excessively long rule documents:

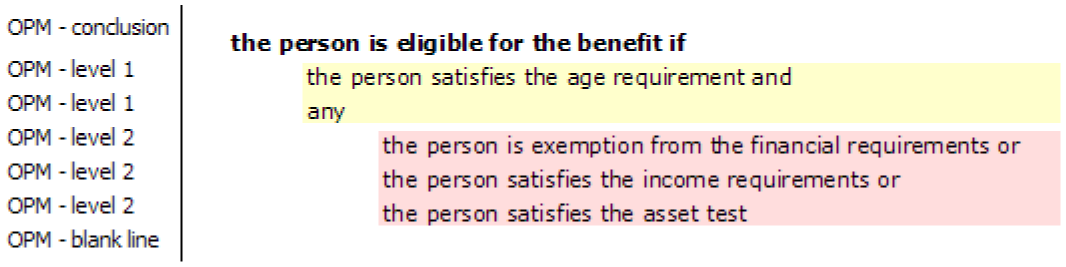
- Multi-developer projects. Only one person can work on a particular rule document at a time.
- Shorter documents validate faster.
- Easier to find the rules you are working on (although appropriate use of heading levels and a table of contents helps alleviate this)

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.

CHAPTER 7. GENERAL TIPS

7.1 TURN ON THE STYLE AREA IN WORD

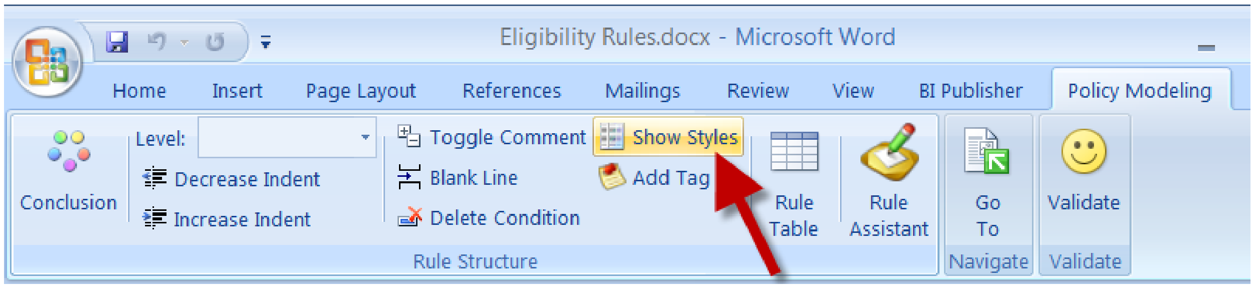
When writing rules, you may find it useful to turn on the Style Area in Word so there is another visual cue (in addition to the colours) as to the OPM styles applied to each line.



```
OPM - conclusion    the person is eligible for the benefit if
OPM - level 1           the person satisfies the age requirement and
OPM - level 1           any
OPM - level 2               the person is exemption from the financial requirements or
OPM - level 2               the person satisfies the income requirements or
OPM - level 2               the person satisfies the asset test
OPM - blank line
```

Having this option turned on in Word is a good idea generally when writing rules and is particularly recommended for new policy modelers. Note that this feature is not part of OPM, but rather just a regular feature of Word that you can make use of for any Word document, not just rule documents.

To turn on the Style Area, click the 'Show Styles' button on the Policy Modeling toolbar in MSWord.



Policy Modeling User Guide

Rule document formatting and structure
https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Policy_Modeling_User_Guide/Work_with_rules/Write_rules_in_Word/Rule_document_format_and_structure.htm

USEFUL LINKS

Online User Guide

- Intelligent Advisor Documentation Library: https://documentation.custhelp.com/euf/assets/devdocs/unversioned/IntelligentAdvisor/en/Content/Guides/Default.htm

General Information

- Oracle Intelligent Advisor on oracle.com: https://www.oracle.com/au/applications/customer-experience/service/intelligent-advisor.html
- Guide to Designing a Policy Model: http://www.oracle.com/technetwork/apps-tech/policy-automation/learnmore/opaguidetodesigningapolicymodel-1531936.pdf
- Tips for Polishing a Policy Modeling: http://www.oracle.com/technetwork/apps-tech/policy-automation/learnmore/polishing-opa-policy-model-2372829.pdf

Forum

- Oracle Intelligent Advisor external discussion forum: https://community.oracle.com/community/groundbreakers/oracle-applications/industries/oracle_policy_automation/oracle_policy_automation_-_general_discussion