ORACLE

Oracle Policy Automation Rapid Language Support (RLS) User Guide

ORACLE

Contents

ORACLE

Introduction

Rapid Language Support is intended to allow Oracle Policy Automation (hereafter referred to as OPA) to expand to new and unsupported regions quickly and efficiently and in a cost effective manner.

The Solution – Rapid Language Support (RLS)

The RLS tools have the following advantages:

- RLS is a simple tool for creating parsers and as such can be used by non-technical personnel; that is, sales consultants and language experts. As no assistance from R&D is required, the solution becomes scalable and efficient.
- RLS is very quick to use - a new language parser can be created within a couple of days
- A demo created with an RLS parser and other Multi-Language Support tools to localize the interface (for example, localizing demo interfaces such as the Rule Authoring interface) can be finished within a week, instead of taking months

Terminology

OPA	Oracle Policy Automation
OPM	Oracle Policy Modeling (previously known as Office Rules Studio)
RLS	Rapid Language Support
UI/GUI	User Interface/Graphical User Interface
RLS Tool	A UI tool used to create and configure RLS parsers
Target Language	The (new) language the user intends to create a parser for using the RLS
RLS parser	A parser created for RLS usage

Requirements

To be able to use the RLS functionality, the following is needed:

- Installation of OPM
- Installed character set for the target language
- Install/Enable keyboard input for the target language
- At least Foundation-level understanding of Oracle Policy Modeling
- The RLS Tool executable (which comes with the OPM Install)

ORACLE

Fundamental RLS Concept – Sentence Templates

The key concept of the RLS is that it uses "sentence templates" to generate the different states of Boolean and Variable attributes when compiling a Rule document (Word or Excel). This is different when compared to full-parsers, which uses verbs to generate the states.

How the RLS Works

In OPA Rule Authoring, there are two types of attributes: Boolean and Variable. Each one also has states, which is generated by OPM when 'compiling' a Rule document.

Boolean has 3 states generated after a compile:

1. Negative
2. Uncertain
3. Question

Variable has 4 states generated after a compile:
Sentence
Question
Uncertain
Unknown

When creating a parser for a new language using RLS, all the user has to do is to specify a 'sentence template' for each state of Boolean and Variable, and that's it! A new parser is ready to be used in compiling Rule documents.

A 'sentence template' is simply a sentence with 'tokens' in it. When compiling and using an RLS parser, OPM inserts the normal (positive) state of the attribute into the token. Therefore, these templates serve as a 'generic' way of generating the various states. Full parsers work differently in that they break sentences down into Subject-Verb-Object, then rearrange them to form the various states.

The following is a sample configuration of an English RLS parser:

Boolean:

- negative state: "it is not the case that {*attribute-text*}"
- question state: "{ *attribute-text* }?"
- uncertain state: "It is not certain whether or not {*attribute-text* }"

Variable:

- sentence state: "{*variable*} is {*value*}"
- question state: "What is {*variable*}?"

{variable} is unknown
{variable} is uncertain

Example – RLS parser vs. Full parser

This section examines how the RLS parser works compared to a full parser.

RLS Boolean Sentence Templates:

Boolean attribute:		**the person is eligible**
	Full Parser (English)	**RLS parser (English)**
Negative state	the person is not eligible	it is not the case that *the person is eligible*
Uncertain state	the person might be eligible	it is not certain whether or not *the person is eligible*
Question state	is the person eligible?	*the person is eligible*?
Variable Text:		the person's age
Variable Value (during runtime):		31
Sentence state	the person's age is 31	*the person's age* is *31*
Question state	What is the person's age?	What is *the person's age?*
Unknown state	The person's age is unknown	*The person's age* is unknown
Uncertain state	The person's age is uncertain	*The person's age* is uncertain

As you can see, from the generated Boolean states the full parser is very different, as it uses the 'is' verb to create new sentences for the states. The RLS parser simply places the Boolean attribute text where the tokens are in the corresponding sentence template.

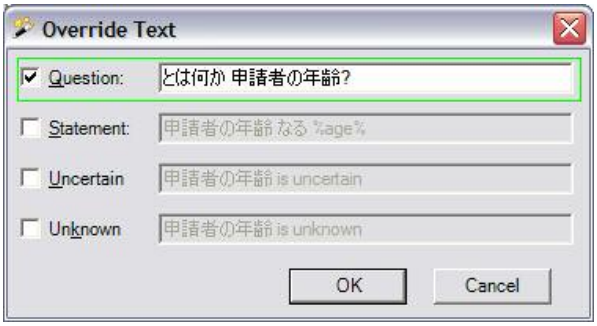
Although the generated text for 'Variable' is the same for both full parser and RLS, the full parser can apply other factors such as 'gender' and 'plural' to correctly generate state text.

Limitations and Remedy

From the example above, it is clear that the RLS parser does not have the functionality that a full parser offers to a rule author. The approach that RLS takes in generating the Boolean and Variable state may seem like a limitation, but below are the reasons why it isn't:

- The RLS is positioned as a tool for building unsupported language parsers quickly, and implementing it to demos, prototypes, or lightweight rulebase implementations.

- The OPM offers the 'override' functionality, which allows the user to override generated text for attributes that does not make sense.



RLS and the RLS parser is the perfect tool to use when dealing with small, and possibly one-off rulebases where the language is an unsupported language. As mentioned before, this makes RLS ideal for creating demonstrations, prototypes, proof-of-concepts, and similar engagements.

Completing the Picture

Apart from defining the generated states for Boolean and Variable attributes for the new language parser, RLS also allows a user to 'translate' the Key Terms, Functions, and Functions – Natural Form used for the new language. By translating the Key Terms and Functions, the Rule Authoring experience is completely written in the target language.

Key Terms

Key Terms allow the user to translate the 'keywords' used in Rule documents in the new language. Keywords such as "if, and, or" are English conjunctions, and using them inline with rule statements in the new language will not be impressive in an OPA demonstration!

With the ability to translate the Key Terms used for an RLS parser, the user can use Conjunctions in the new language.

For more information about Key Terms and Conjunctions, please see the *Rule Developer Guide*.

Functions

Functions allow the user to translate English functions such as Maximum() or AddDays() into the target language equivalent. These translated functions can then be used in Rule documents that use the RLS parser.

For more information about Functions, please see the *Rule Developer Guide*.

Functions – Natural Form

Like the Functions section, Functions – Natural Form allow the user to translate the natural-form of the English functions such as "[x] is greater than [y]" into the target language equivalent. These translated functions can then be used in Rule documents that use the RLS parser.

LThis allows the user to create impressive flowing rule statements using natural-form Functions written in the target language.

For more information about natural-form Functions, please see the *Rule Developer Guide*.

Other Features

Support for Languages without spaces
RLS also allows the user to create RLS parsers that do not use spaces. This allows languages such as Japanese, where there are no spaces between characters, can be supported.

Substitution
Substitution also works with the RLS, similar to how it works in normal full parsers, except for the following:

- it does not support pronoun substitution
- it does not support gender and plural based substitution

In addition, substitution only matches whole words (words with space before and after) unless the language does not use spaces.

Recognition of Negative Boolean sentences

Writing negative states of an attribute is still recognized by the compiler; for example, if the *default-negative* configuration is *It is not the case that {attribute-text}* and the attribute is *the man is happy*, then writing *It is not the case that the man is happy* as a Boolean conclusion/premise is recognized as a negation of *the man is happy.*

Overriding the negative state of a Boolean attribute is also recognized. If we override the negative state of *the man is happy* to *the man is not happy*, then when writing *the man is not happy* as a Boolean conclusion/premise is also recognized as a negation of *the man is happy*.

RLS UI Tool

Now that you are familiar with how RLS works, this section will teach you how to use the *RLS UI Tool* for creating and configuring your RLS parsers.

The *RLS UI Tool* is an executable file, which would have been supplied to you together with this user guide. If you have not done so already, please place it in the 'bin' directory of your OPA installation; for example, *C:|Program Files|Oracle|OPA|10.0|bin*

Creating a new parser

To create a new parser:

1. Run the *RLS UI Tool*

2. The *RLS Configuration* window is displayed, containing a list all of your RLS parsers:

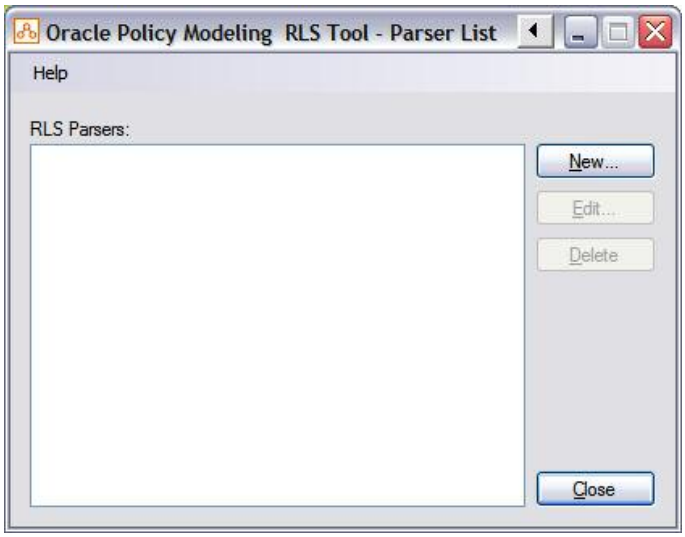


Figure 1 - RLS Configuration window

3. Click on the **New** button

4. The *New Parser* window will be displayed.

Figure 2 - New Parser window

5. Enter the name of the target language that your new RLS parser will support in the *Display Name* field
    a. This will automatically place the cursor on the Language list to the closest match of your target language
    b. Note that each language has a *culture/region* attached to it. The *culture/region* is in parenthesis. This is only relevant if you want to create multiple parsers for the same language; for example, Spanish (Mexico) and Spanish (Spain).

6. After entering the name of the target language, if the Language cursor has not selected the correct language for you, scroll through the list to select the target language
    a. You will notice that some languages are already used. This happens when a full or RLS parser has been allocated for that language-culture.

7. Click on the **Create** button. If successful, the *Edit parser* window is displayed.

Editing an Existing Parser

1. Run the *RLS UI Tool*
2. The *RLS Configuration* window is displayed, containing a list of all of your RLS parsers
3. In the *RLS Configuration* list, select the parser you want to Edit
4. Click on the **Edit** button
5. The *Edit Parser* window is displayed
    a. The *Edit Parser* window is also displayed after creating a new parser

6. You can now edit the parser's:
    a. Display Name
    b. Language-Culture allocation
    c. Usage of Space (tick the box if it uses spaces, untick if it doesn't)
    d. Sentence Forms
    e. Key Terms (e.g. Conjunctions)
    f. Functions
    g. Functions – Natural Form

Configure a Parser's Sentence Forms

1. Open the *RLS UI Tool*, and open the target Parser for editing
2. Switch to the *Sentence Form* tab

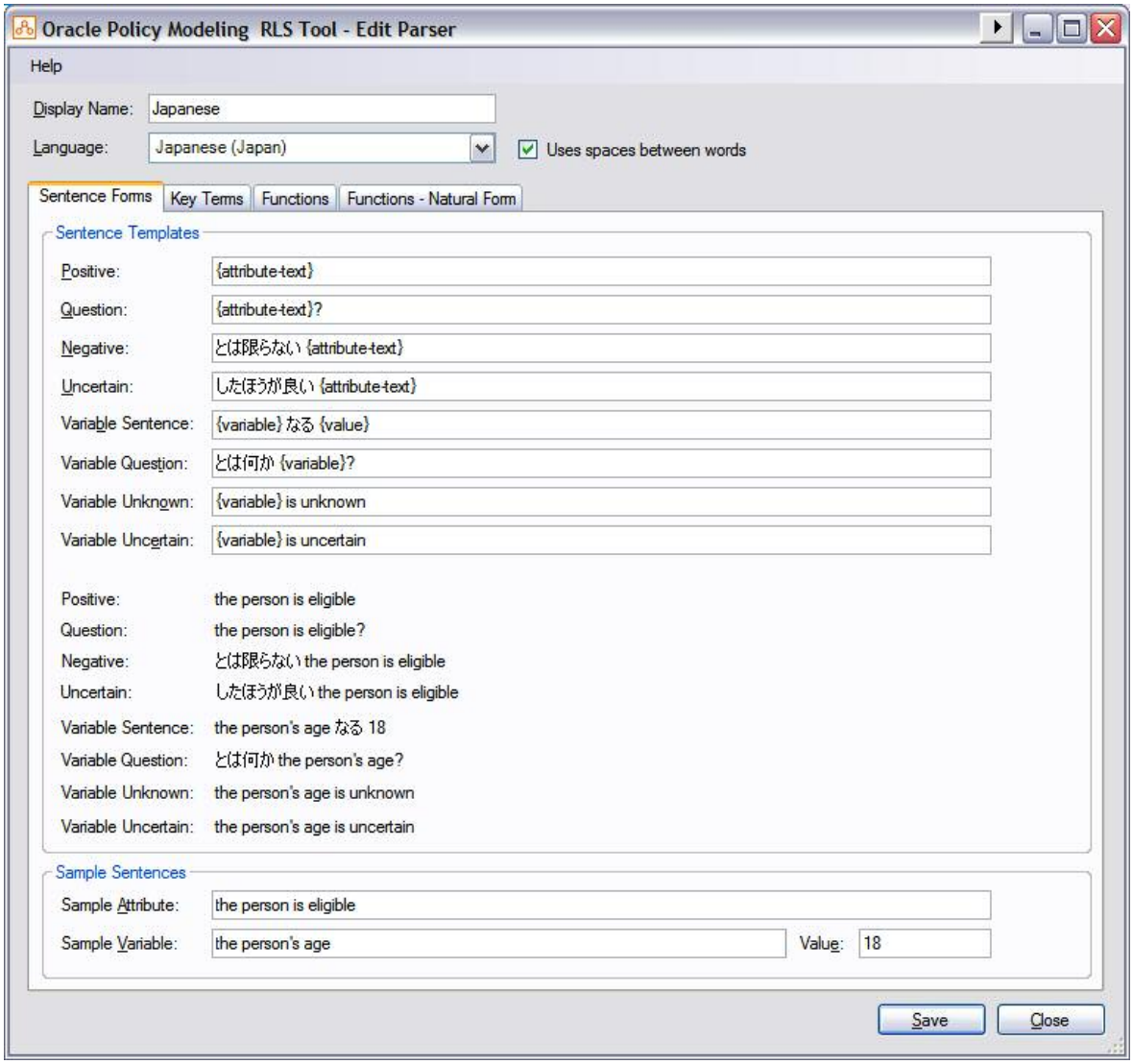


Figure 3 - Sentence Form tab

3. The *Sentence Templates* section has the following fields:
    a. Positive – define the Positive sentence template for Boolean attributes
    b. *Question* – define the Question sentence template for Boolean attributes
    c. *Negative* – define the Negative sentence template for Boolean attributes
    d. *Uncertain* - define the Uncertain sentence template for Boolean attributes
    e. *Variable Sentence* – define the Sentence template for Variable attributes
    f. *Variable Question* – define the Question template for Variable attributes
    g. *Variable Unknown – define the Unknown template for Variable attributes*
    h. *Variable Uncertain – define the Uncertain template for Variable attributes*

4. **NOTE:** The Positive template is only used during runtime, e.g. when running the rulebase in Debugger or Web Determinations.

5. When creating a new RLS parser, the eight *Sentence Templates* fields will always have default text. To set the *Sentence Templates* to your target *Language*, write your own templates in each field.
    a. For each field, notice that there are *tokens* – text surrounded by the curly braces { and }
    b. These tokens must be present after you write your own templates, and must be exactly the same; for example, {attribute-text}, not {Attribute-txt}
    c. You can place these tokens anywhere in your text, so long as they are present
    d. For *Variable Sentence*, there are two tokens

6. Below the eight *Sentence Templates* fields are five lines of text.
    a. These are the *sample* sentences for your sentence templates.
    b. Each line is generated by the corresponding sentence template, and the sample attribute text in *Sample Sentences* section.
    c. These text are useful for tweaking your sentence templates.

7. The section at the bottom is the *Sample Sentences* section which has the following fields:
    a. *Sample Attribute* – this should be the sample Boolean attribute text to be used in generating the Boolean states.
    b. *Sample Variable* – this should be the sample Variable attribute text to be used in generating the Variable states.
    c. *Value* – this should be the sample *Value* of the Variable attribute text; for example, if the Variable attribute is *the person's age*, then the Value might be *31*.

8. Click on the **Save** button to persist your changes
    a. If there are errors in your sentence templates (for example, malformed tokens), an error window will appear.

Configure a Parser's Key Terms and Functions

For this example, we will use the Key Terms tab. Modifying items in the Functions tab work exactly the same way.

1. Open the *RLS UI Tool*, and open the target *Parser* for editing
2. Switch to the *Key Terms* tab

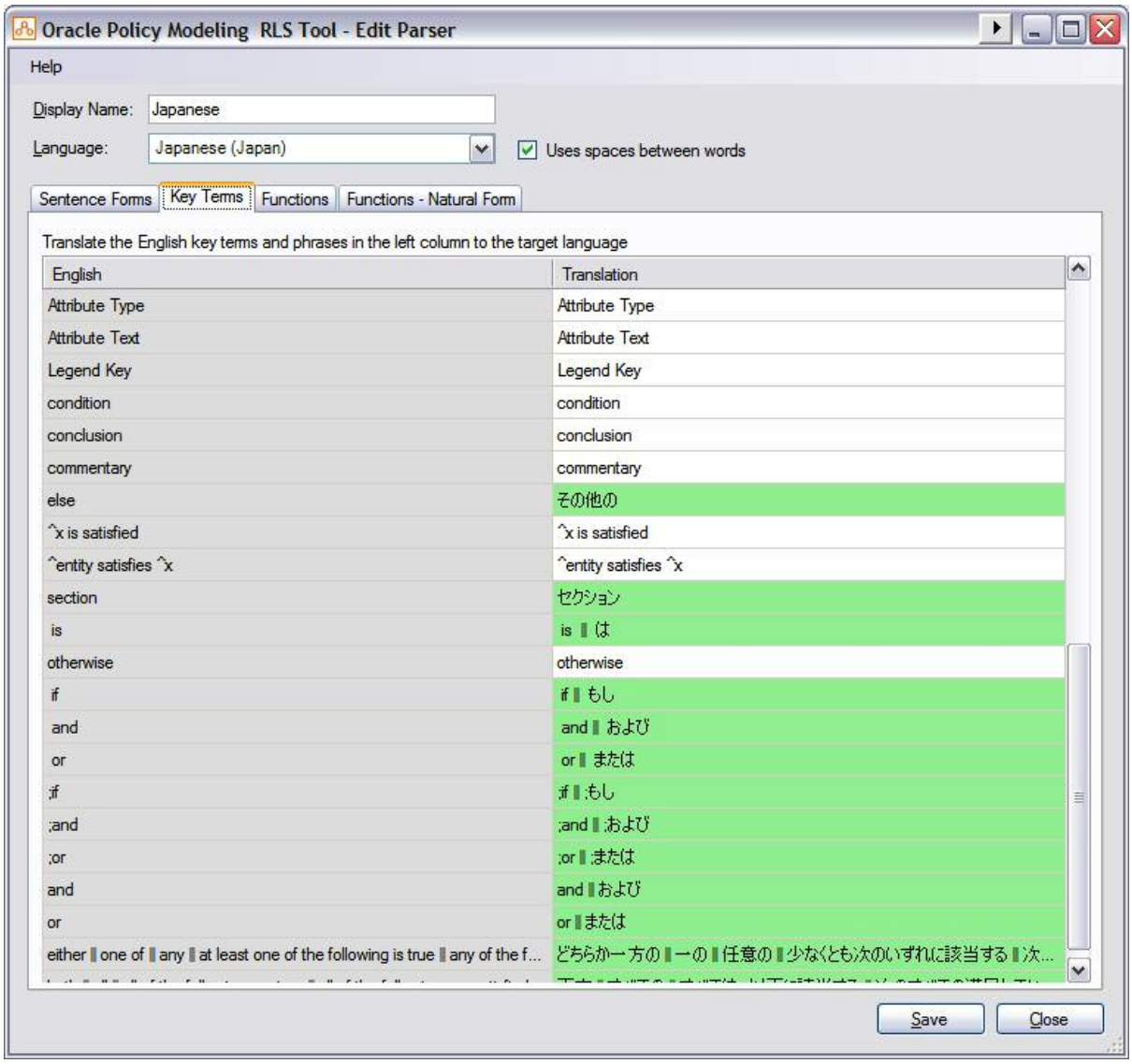


Figure 4 – Key Terms Tab

3. There are two columns – *English* and *Translation*
    a. The *English* column contains the English version of a *Key Term*. This column is read-only.
    b. The *Translation* column is editable, and defaults to the English *Key Term*.

4. Translate the *Key Term* that you need
    a. Each row in the *Translation* column can be translated.
    b. When you change the value to something that is different to the English version, it is marked in green so that you can track *Key Terms* that you have translated.
    c. You can write multiple translations for each English *Key Term* by separating each translation value with double pipes '||'.
    d. Each row in the *Translation* column must not be empty.

5. Click on the **Save** button to persist your changes
    a. If there are errors in your Key Terms (for example, empty *Key Term*), an error window will be displayed.

Configure a Parser's Natural-Form Functions

1. Open the *RLS UI Tool*, and open the target *Parser* for editing
2. Switch to the *Functions – Natural Form* tab

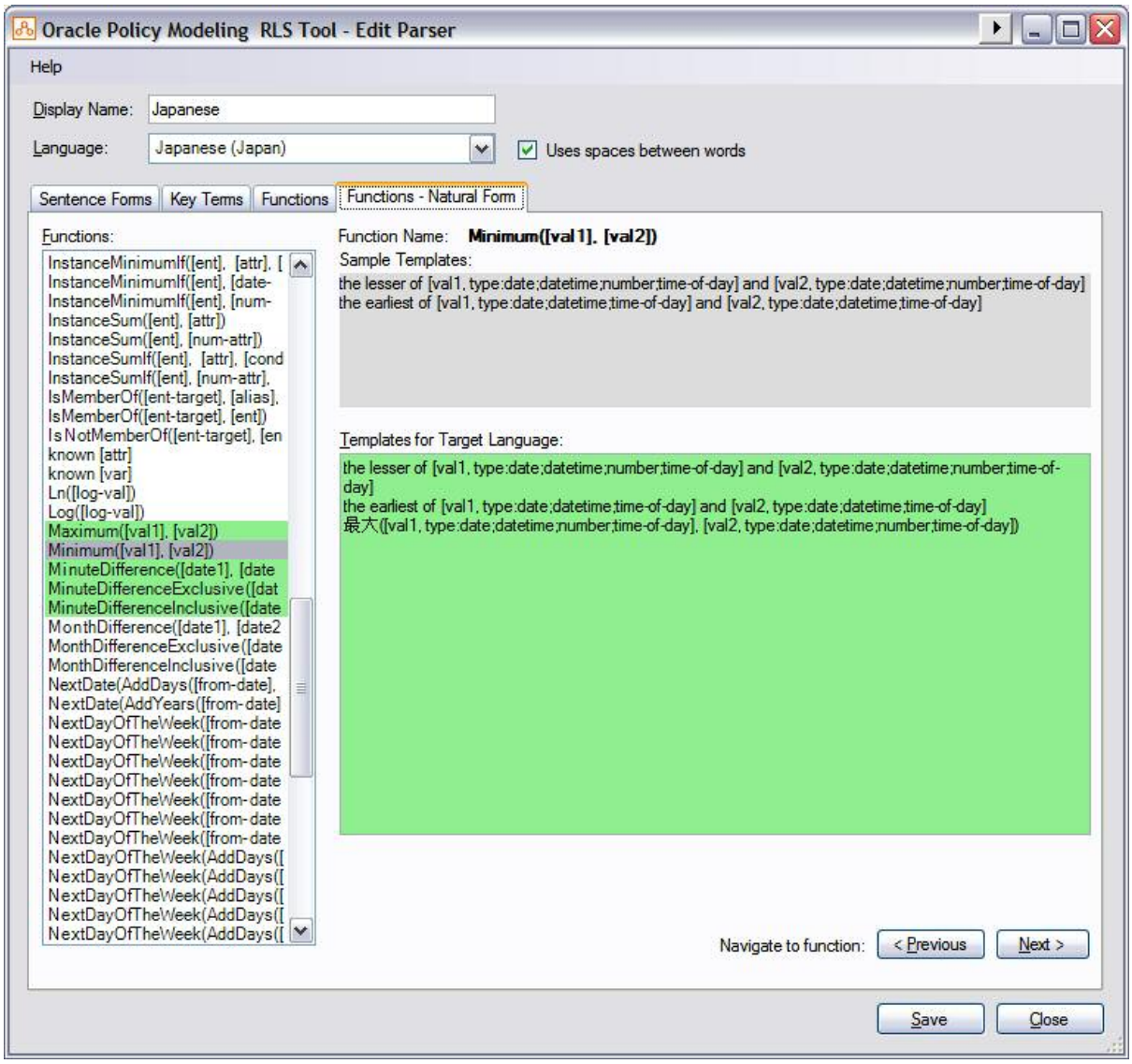


Figure 5 - Functions tab

3. There are three main sections:
    a. *Functions List* – a list of all the Functions.
    b. *Sample Templates* – displays the default English Templates for the currently selected Functions in the Function List.
    c. *Templates for Target Language* – displays the templates of the current Parser for the selected Function. By default it contains the templates in the Sample Templates.

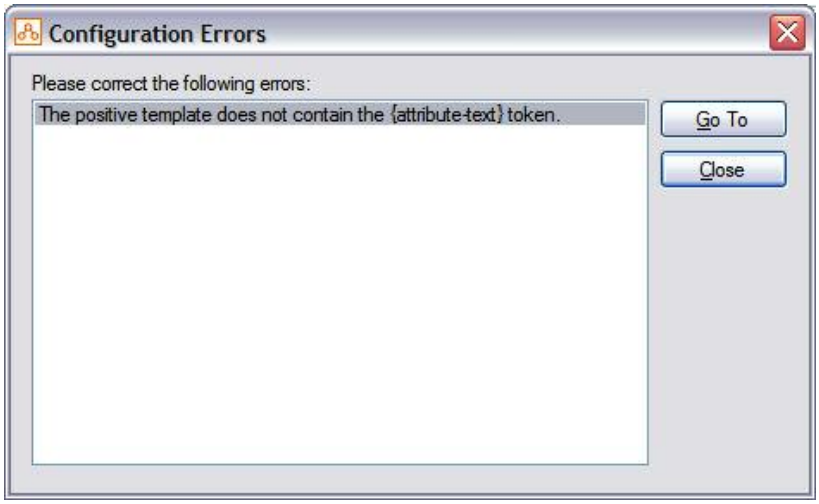
4. Select a *Function* in the *Functions List*, that you want to translate
5. The *Sample Templates* and *Templates for Target Language* text content should change to the currently selected *Function*.

6. Add as many *templates* for the selected *Function* by modifying the lines in the *Templates for Target Language* textbox
    a. Each line represents a *template*.
    b. You do not have to delete the default English Templates, you can simply add a new line below them, and write your translation of the *Function* as a new template.
    c. There must be at least one template for any *Function*.

7. Check the tokens of the *Function*
    a. Each template of a *Function* must correspond to one of the templates in the *Sample Templates List*.
    b. To correspond, your templates must have the tokens of one of the templates. They can be in any order, but the combinations must be the same.
    c. Examples: for Function [lhs] < [rhs]
        i. *[rhs] has a higher value than [lhs]* -> is valid.
        ii. *[lhs] is less than [rhs type:number]* -> is NOT valid as it is mixing tokens from different templates.

8. Select a different *Function* in the *Function List* to modify a different *Function*
9. Click on the **Save** button to persist your changes for the *Function*
    a. If there are errors in your *Function* templates (for example, malformed tokens or empty Function/s), an error window is displayed.

Handling Errors with the Error Window



The *Configuration Errors* window handles any error from all three tabs. The *Configuration Errors* window lists all error items. The user can double click on an item to focus on that tab/field, or select an error item and click **Go To** to achieve the same action.

Focusing on an error item will bring the user to the correct tab, and form input.
- For the *Sentence Forms* tab, it will put the cursor on the field with the error
- For *Key Terms* tab, it will put the cursor on the *Translation* row with the error
- For *Functions* tab, it will select the correct *Function*, and place the cursor on the textbox

Managing RLS Parsers

The user can manage the RLS to:

- Delete existing parsers
- Change the Display name and language-culture allocation of a parser.

When deleting a parser, note that it is permanently deleted and cannot be undone. Testing an RLS Parser

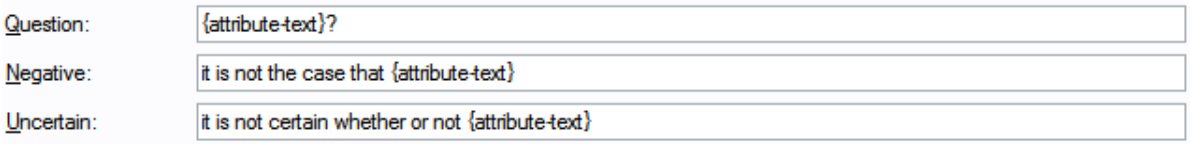
This section will detail processes and tips on how to test your RLS Parser configuration, to ensure that your configurations are being applied.

Test the Setup/Basic Configuration

The steps described here will verify that you have setup the *Sentence Forms* correctly.

Test the RLS parser's basic configuration

1. Test the parsed forms for *Boolean* attributes
   a. Open a project in *Office Rules Studio*.
   b. Create a new MS Word rule document (the filename is not critical for the test).
   c. Open the MS Word document and create two boolean attributes: one conclusion and one Level 1 premise.
   d. Compile the rulebase; the *Confirm new Attributes* popup should appear.
   e. Select one of the attributes, and click the **Edit** button.
   f. Check that the *Sentence forms for selected parse items* corresponds to the sentence forms in the *Sentence Forms* tab:



   i. *<default-question>* node value corresponds to the first item.
   ii. *<default-negative>* node value corresponds to the second item.
   iii. *<default-uncertain>* node value corresponds to the third item.

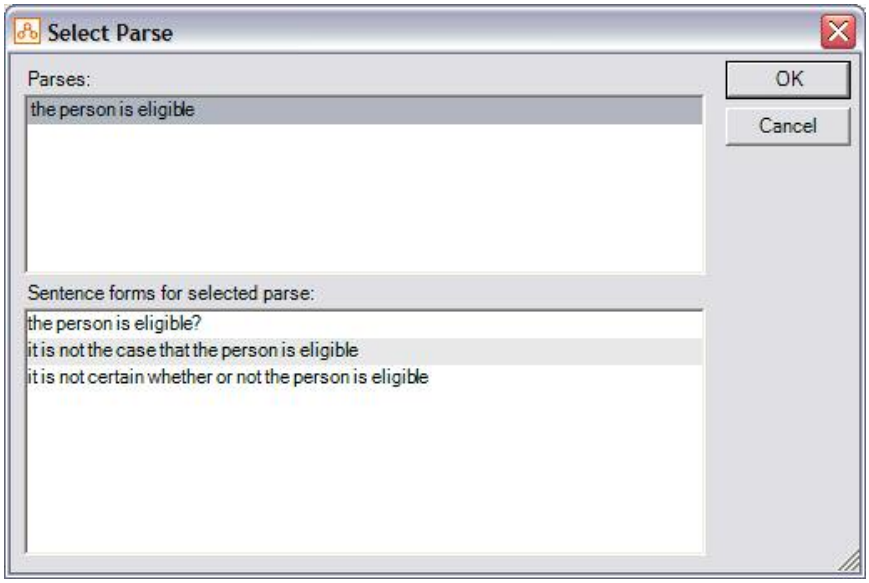


Figure 6 - Confirm new attributes window with the generated sentence forms

2. Test the parsed forms for *Variable* attributes
    a. Open a project in *Office Rules Studio.*
    b. Create a new *Properties* file.
    c. Add a new attribute.
    d. Set the *Data Type* to *Number*.
    e. Provide a *Public Name* and *Text* (for example, the person's age).
    f. **Save** the new attribute.
    g. In the Studio menu bar, select **Build -> Build and Debug**.
    h. Set the *Debug* process to *without screens*, and press **OK** – the *Debugger* window should appear with the recently-created attribute.
    i. In the list, the *Text* column of the attribute should show the sentence form of the new attribute, which should correspond to the *Variable* settings in *Sentence Forms* (for example, below it is *{variable-text} = {value}*.

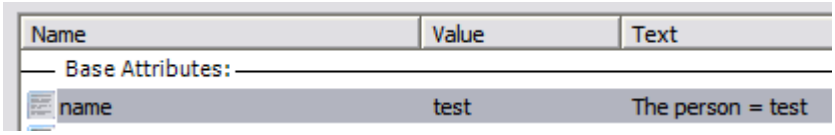


Figure 7 - Debugger list of attributes

    j. Double-click on the appropriate variable attribute from the list, and check that the question form corresponds to the *Variable Question* template (for example, below it is *{variable-text}???*)

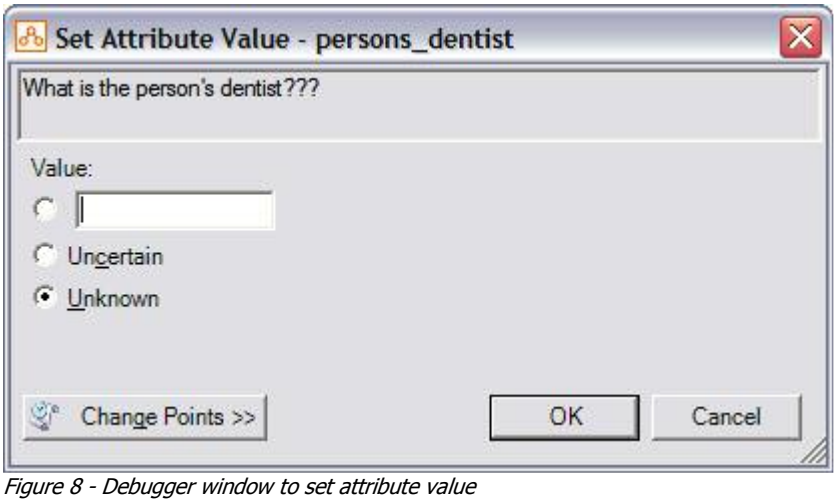


Figure 8 - Debugger window to set attribute value

Test the Advanced Configuration

These will verify that your settings in Key Terms and Functions (including Natural Form) are available when authoring rules.

Test Key Terms. This test will ensure that translated rule elements are recognized during compilation.

1. Pick one of the rule elements to test (for example, the *if* Key Term), and change the *Translated* format to a test value (for example, *testif*). **Save** the changes.

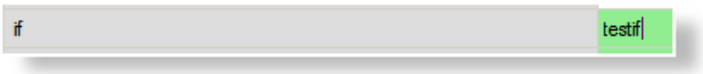


Figure 9 - Key Term setting of 'If'

2. Open a project in *Office Rules Studio*.
3. Create a new MS Word rule document, the filename is not critical for the test.
4. Open the MS Word document, and create two boolean attributes: one conclusion, one Level 1 premise.
5. Compile the rulebase; the *Confirm new Attributes* popup should appear.
6. For the conclusion attribute, ensure that the *if* Key Term is not part.

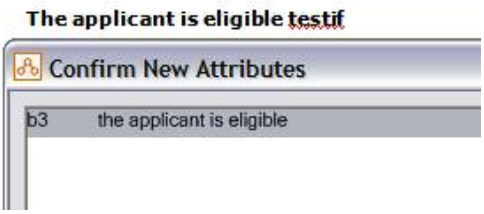


Figure 10 – Rule element configuration WORKING

For other rule elements (for example, *and*, *or*, *all*, *any*), the above process will also work (with an error *Missing Key Term* displayed after agreeing to the new attributes). The same approach applies: check in the *Confirm New Attributes* window that the rule element being tested does not appear as part of the attribute itself.

Test Functions (Terse and Natural-form)

This verifies that translated functions are recognized.

1. Select a *Function* to test (for example, template name *is greater than*), and change one of the templates to a test value (for example, *is more than*). **Save** the changes.
2. Open a project in *Office Rules Studio.*
3. Create a new MS Word rule document, the filename is not critical for the test
4. Open the MS Word document, and create one boolean attribute as the conclusion, one Level 1 premise as a number variable, using the *is more than* function (e.g. the applicant's age is more than 17)
5. Compile the rulebase; the *Confirm new Attributes* popup should appear
6. To determine whether the test is successful:
    a. The compile process must be successful
    b. The compiled attributes (red text) next to the Level 1 premise reflects the *is more than* function.

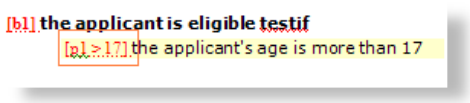


Figure 11 - the new function "is more than" compiles correctly

7. Testing other *Functions* is the same as above. The core approach is to make sure that it compiles successfully, and the compiled attributes (red text) reflects the new/translated function.

Localizing the Rulebase Authoring UI (MS Word/Excel)

The end goal of the RLS is to be able to create demonstrations of the OPM in languages that are not currently supported. The demos will try to convince the audience that OPM's natural language benefits works for their native language just as well as in OPM's main language, English.

As stated in previous sections, the RLS allows the user to write complete rulebases in an unsupported language. While this is the key functionality and selling point of the OPM, the demo can be improved by localizing the authoring user interface to further enhance OPM's simple and user-friendly approach to rule authoring.

Demonstrating the OPM with the rulebase in their native language *together* with a fully translated authoring UI, adds another level of completeness, and therefore a more effective demo. UI Localization covered in this section will allow the demo creator to translate:

- MS Word Plugin buttons
- MS Excel Plugin buttons
- error messages
- Buttons/texts in the popups displayed when using the MS Word/Excel Plugin tools

The ability to translate texts in a software is termed "localization", it is the step where actual translation takes place. "Internationalization" is when the text of the software can be translated. Therefore we can say that the Word/Excel Plugins has been" internationalized". The new OPM Interactive (OPM 2009) is fully internationalized, and therefore can be "localized". It is not covered in this section. If you need to localize the OPM Interactive, please consult the OPM Interactive manual.

This section details the steps required to localize the Authoring UI. The user must first setup the localization file for the target language, then the translation process, and finally configuring OPM Studio to use the localization file.

It is important to note that some parts of the UI can be left *unlocalized*; for example, for error messages: only error messages to be invoked needs to be translated, but for the MS Excel Plugin/Addin texts they need to be completely translated or not at all.

Note that you need XML skills to setup the localization.

Setup

To create a UI localization file for a certain language:

1. Locate the *Language Bundles* directory, on a normal installation this will be in *C:|Program Files|Oracle|Policy Modeling|language|Bundles*
2. Open the file *officeRules.xml* in a text editor (for example, Notepad).
3. Save the file as *officeRules.xx-yy.xml* where **xx-yy** is the *culture name*
    a. **xx** is the language code and **yy** is the country/region code
    b. Conventionally the country code is in uppercase, but lowercase is accepted; for example:
    officeRules.ja-JP.xml for Japanese
    officeRules.es.xml for Spanish

    Note: Some culture codes only need a 'language code' (neutral culture)
    http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx

Translation

After setting up the UI localization file, you can start translating keys in the file:

1. Open the UI localization file; for example, *C:|Program Files|Oracle|Policy Modeling|language|Bundles|officeRules.es.xml.*
2. Apart from the root xml element, all child elements will be a *string* xml node with a *key* attribute.
    a. Translate the text in the *string* element (in green); **do not** translate the text in the *key* attribute of the *string* element (in red); for example:
    <string key="Attribute Editor">Redactor de Atributo</string>

    b. After you have made your changes, save the file, and close/reopen OPM Studio; OPM Studio needs to be closed/reopened to recognize the changes in the UI localization file.

3. **Note**: some of the keys have prefixes, which reveal where they will appear
    a. "WORD_TOOLBAR" prefix – for the MS Word Plugin
    b. "EXCEL_TOOLBAR" prefix – for the MS Excel Plugin
    c. "k" prefix - Messages/Error Messages
    d. "dlg/lbl/btn" prefixes – Texts in dialog boxes or buttons or labels

Using the New UI Localization File

To use the new UI localization file, OPM Studio needs to be configured to use the new UI language:

1. At the menu of OPM Studio, go to **Tools -> Options.**
2. Go to **Environment -> General** in the left-hand pane.
3. After opening the *General* pane, the *Authoring UI Language* dropdown selection list should be visible.
4. Change the dropdown to the new language (for example, Spanish), and click **Apply**.

Figure 12 - Using a New UI Localization

Examples of UI Localization (Spanish)

MS Word Plugin localized to Spanish

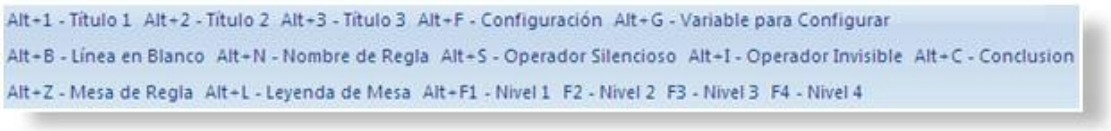


Figure 13 - MS Word Plugin Localized in Spanish (left side)

Figure 14 - MS Word Plugin Localized in Spanish (right side)

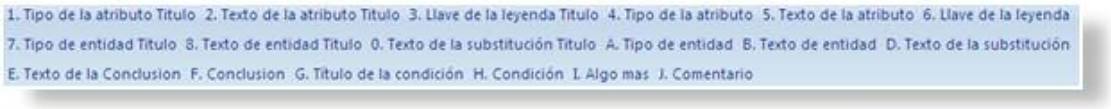


Figure 15 - MS Excel Plugin Localized in Spanish

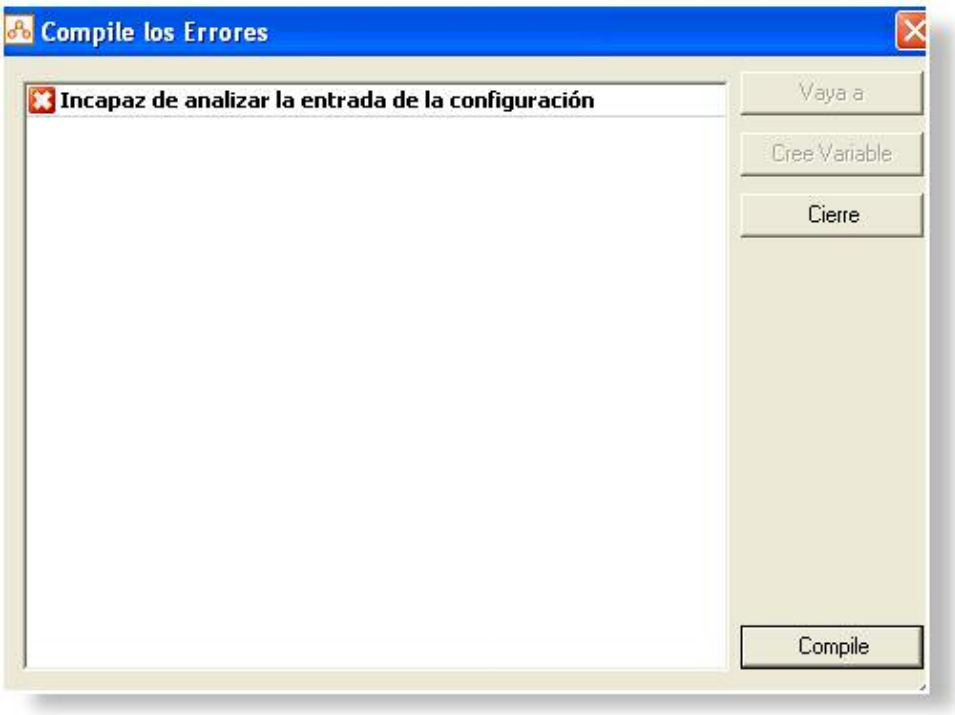


Figure 16 - Error Message, Window Title, and Buttons Translated to Spanish

Authoring in RLS

Authoring rules using an RLS parser is very similar to a normal fully-featured parser (for example, English US). The main difference, as stated previously in describing the basic configuration, is that:

- when parsing Boolean attributes, the other three states of the attribute (negative/question/uncertain forms) are generated via a generic statement; normal parsing usually generates the other three states using a more complex algorithm, using a *verbs list* to identify the very in an attribute and break it up into its Subject-Verb-Object compontents.
- When parsing variable attributes, the other two states of the attribute (statement/question forms) are also generated via a generic statement

Realistically, an approach such as this will only achieve a 10-50% success rate, depending on the language structure and the skill of the language expert. To make sure that the parsed forms for all of the attributes in the demo rulebase are correct, using the *Override* functionality is the key. This is discussed in *Properties File* topic below.

The following topics provide details and tips for authoring rules in RLS. They also outline what to expect in the various areas of the authoring process, specifically what can be written in the new language, and what stays in English.

General

Authoring rules with the RLS parser means that the user can write rulebases in a different language, using a different character set. Note that this covers rulebases, and not the authoring user interface. A lot of the authoring interfaces (Studio, MS Word/Excel/Visio Plugins, and Interactive) will still be in English even though the rulebase and its attributes are in a different language/character set (for example, Japanese characters). This is acceptable since it is a demo that will focus on OPM's natural language capability, which is focused on the rulebase and the attribute parsing.

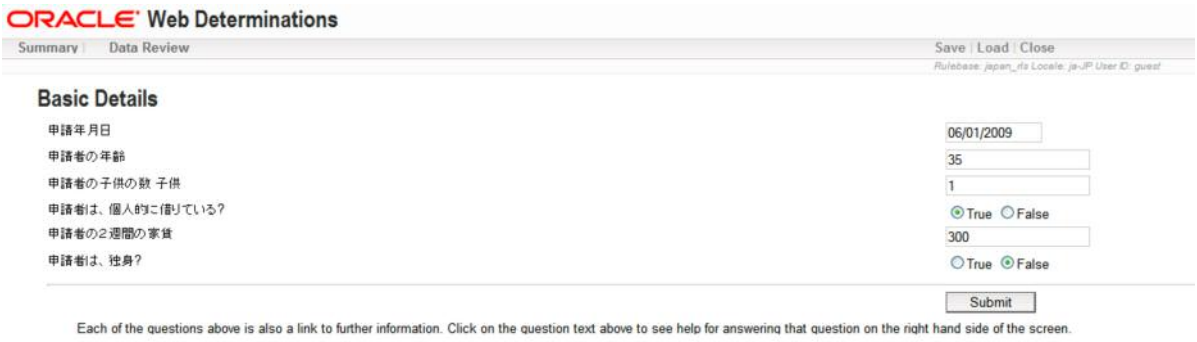


Figure 17 - Mixture of ASCII and Japanese Characters in an Interactive Session

The OPM Studio has also been *internationalized*, meaning it can handle various character sets. Because of this, all of the various tools and functionalities in Studio operate as per normal, even if the various attribute IDs and texts are written in a non-English character set. This means that authoring a rulebase for an RLS parser is still easy and efficient. For example, if the user wrote a rulebase in Japanese, he/she can use the *Find attribute usage* function in Studio by using a Japanese character to find attributes that contain that character.

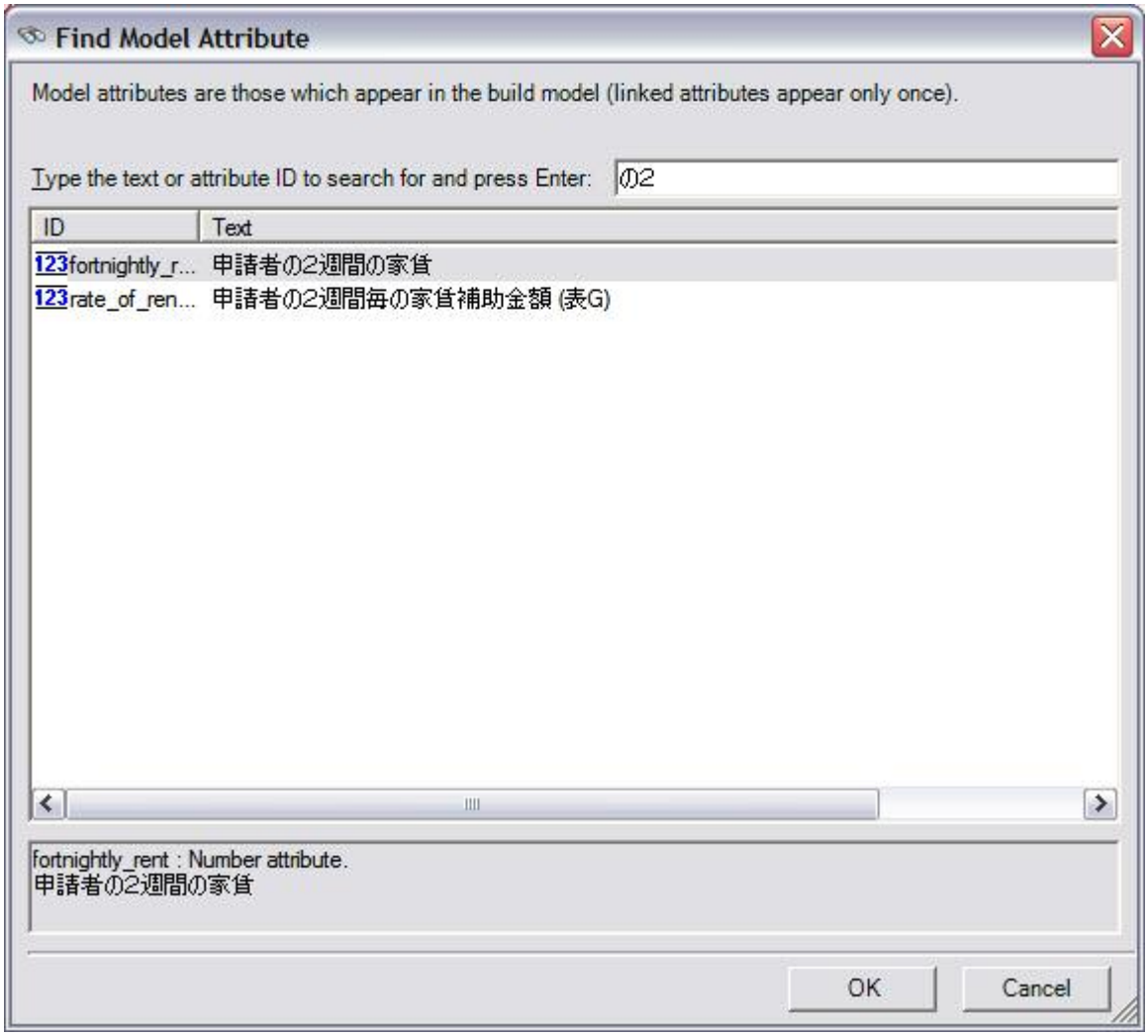


Figure 18 - Find Attribute Usage using Japanese characters

MS Word

Most of the configurations involved in basic and advanced configurations, are related to the MS Word authoring environment. Therefore it is probably the most demo-able component in that most of the texts and functionalities have been configured to work in the new language.

As mentioned before, the key difference between an RLS parser vs. a full parser, is that the RLS parser only uses a generic statement form to generate the various states of an attribute. Therefore it is recommended that all attributes are given public names to allow override of the default *generated* states.

What is/can be translated (with complete configuration/translation):
writing/parsing attributes

- rule elements; for example, if/and/or/any/all
- most functions – both terse and natural-form
- various keywords such as true/false/uncertain/unknown

- From the *Localizing the Rulebase Authoring UI (MS Word/Excel)* section:
  - error messages; for example, error messages from compile process
  - OPM Word Plugin buttons
  - OPM Word Plugin windows/popups and their texts/buttons

- Through the *MS Office Suite Language Settings*:
  - MS Word User Interface

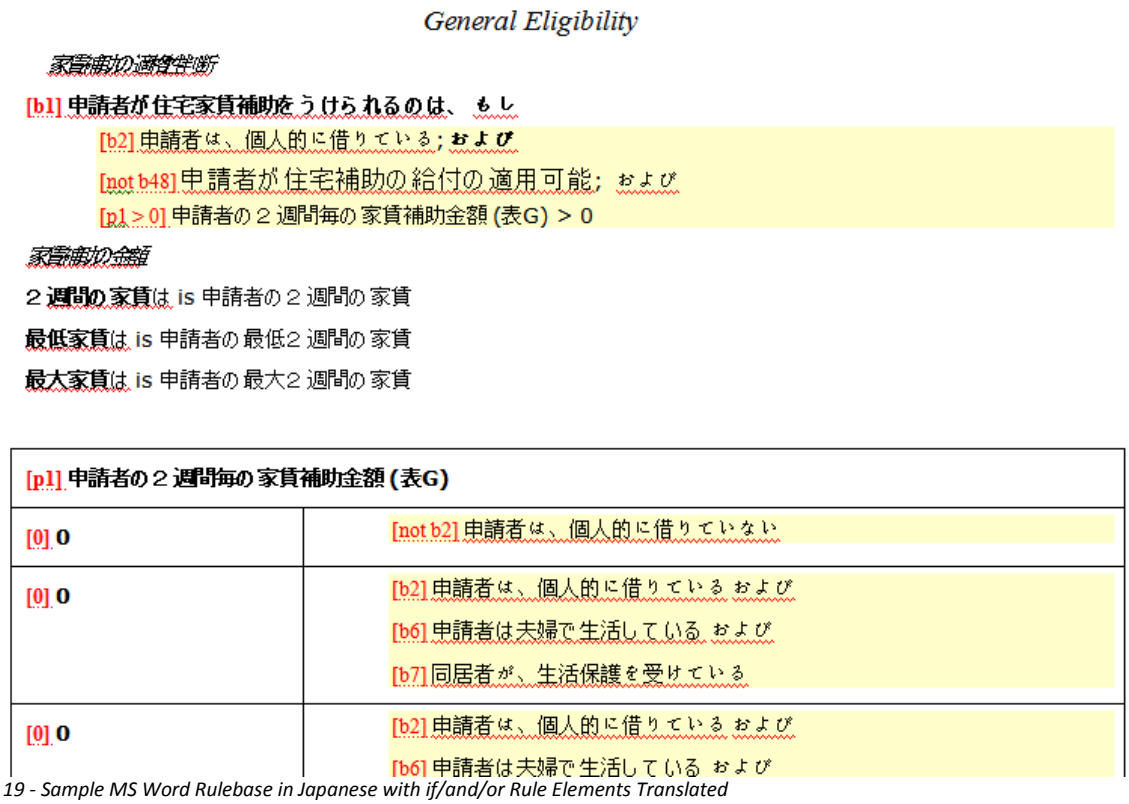
What is not translated:

- variable declarations; for example, Variable_Number
- OPM Style names
- numbers and dates
- name of days/months (except Functions that use day/month names)
- compile text (the red text)
- some functions cannot be currently translated; for example, sin, cos, tan

Authoring Tips:

writing negative states of an attribute is still recognized by the compiler; for example, if the *default-negative* configuration is *It is not the case that* and the attribute is *the man is happy*, then writing *It is not the case that the man is happy* as a Boolean conclusion/premise, the following occurs:

- if *the man is happy* is a public attribute, it compiles straightaway and the compile red text reflects it (for example, [not b48])
- if it is a new attribute, in the *Confirm new attribute* window it will recognize the attribute as *the man is happy*, not *It is not the case that the man is happy*.



General Eligibility

家賃補助の適格性判断

[b1] 申請者が住宅家賃補助をうけられるのは、 もし

[b2] 申請者は、個人的に借りている; および

[not b48] 申請者が住宅補助の給付の適用可能; および

[p1 > 0] 申請者の２週間毎の家賃補助金額 (表G) > 0

家賃補助の金額

２週間の家賃は is 申請者の２週間の家賃

最低家賃は is 申請者の最低２週間の家賃

最大家賃は is 申請者の最大２週間の家賃

[p1] 申請者の２週間毎の家賃補助金額 (表G)	
[0] 0	[not b2] 申請者は、個人的に借りていない
[0] 0	[b2] 申請者は、個人的に借りている および [b6] 申請者は夫婦で生活している および [b7] 同居者が、生活保護を受けている
[0] 0	[b2] 申請者は、個人的に借りている および [b6] 申請者は夫婦で生活している および

Figure 19 - Sample MS Word Rulebase in Japanese with if/and/or Rule Elements Translated

ORACLE

MS Excel

There is no configuration of the RLS that relates to the MS Excel authoring component, but there are translations that can be made within the MS Excel document itself. For example, in the *Declarations* sheet, by default all of the headings are in English. But these headings (both *Entity* and *Attribute* type/text/legend) can be translated, as long as the correct style is implemented on the respective cells. Also, the legend text written in a different character set works when used in the condition/conclusion headings in the *Rule Table* sheet.

What is/can be translated:

- In general
  - The name of the two sheets
  - various keywords such as true/false/uncertain/unknown

- *Declarations* sheet
  - Entity type/text/substitution text headings can be written in any charset
  - Attribute type/text/legend key headings can be written in any charset
  - Entity text and substitution text can be written in any charset
  - Attribute text and legend can be written in any charset

- *Rule Table* sheet
  - the condition/conclusion headings correspond to the declarations, therefore can be written in any charset
  - actual premise/conclusion values can be written in any charset
  - the *else* keyword (just need to set the cell to *Else style*

- Through the *Localizing the Rulebase Authoring UI (MS Word/*Excel) section:
  - error messages; for example, error messages from compile process
  - OPM Excel Plugin buttons

- Through the MS Office Suite Language Settings:
  - MS Excel User Interface

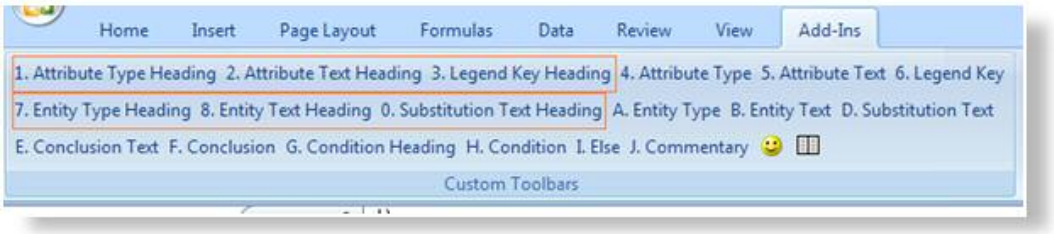
What is not translated:

- *Declarations* sheet
  - Attribute type – this must be in English; for example, Number, Boolean

How to translate heading cells

1. Select cell to edit , note its type (entity type/text/substitution text or attribute type/text/legend key)
2. Change the heading text by entering the new heading text in the *Function* field

3. Set the cell to the correct heading style based on its type by clicking the correct style button from the Oracle Policy Modeling Add-in toolbar/ribbon.
    a. Ensure that you are using the *Heading* style.



4. **Save** the MS Excel document

Note: This also works for the *Else* cell, in that it does not matter what text is inside the cell, so long as it's style is set to *Else*.

condition 申請	condition	conclusion 申請	conclusion
commentary			
	その結の		

Figure 20 - Sample Rule Table Worksheet in Japanese

Properties File

The *Properties* file is an important part of authoring rules with an RLS parser, as it gives the rulebase author the ability to *Override* various parsed forms of attributes.

The Override functionality allows the user to modify the negative/question/uncertain form for Boolean attributes, and the statement/question form for variable attributes. As explained before, due to the way the RLS parser parses attributes, it is very likely that some attributes will not make sense from the automatic parsing. The *Override* function allows the user to *fix* parsed forms of an attribute that does not make sense.

What is/can be translated:

- an attribute's *Public Name*
- a Boolean attribute's Text, including Overridden forms
- a variable attribute's Text, including Overridden forms
- Entity name (other than Global)
- Entity Relationship texts
- Filter

What is not translated:

- Properties file UI; for example, value of dropdowns, field labels
- Global entity name

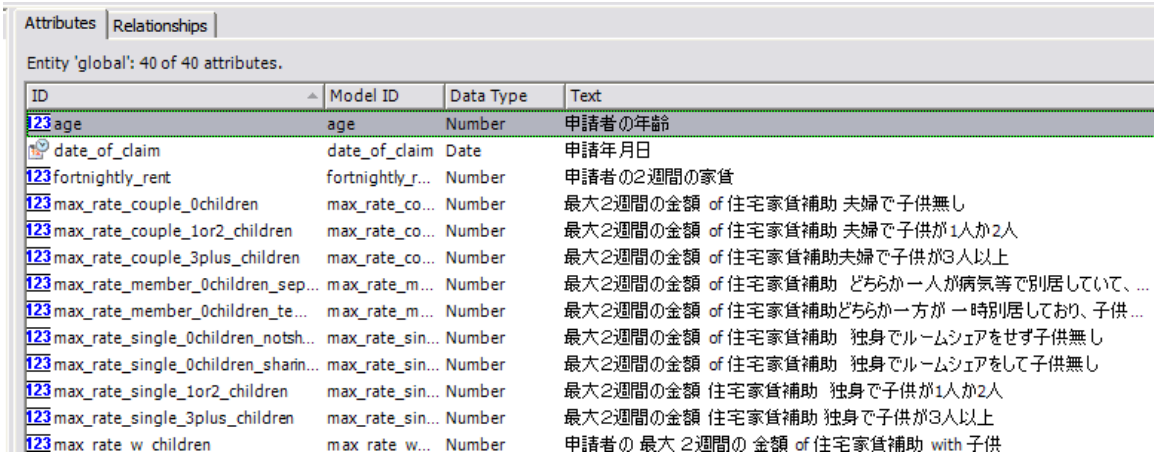
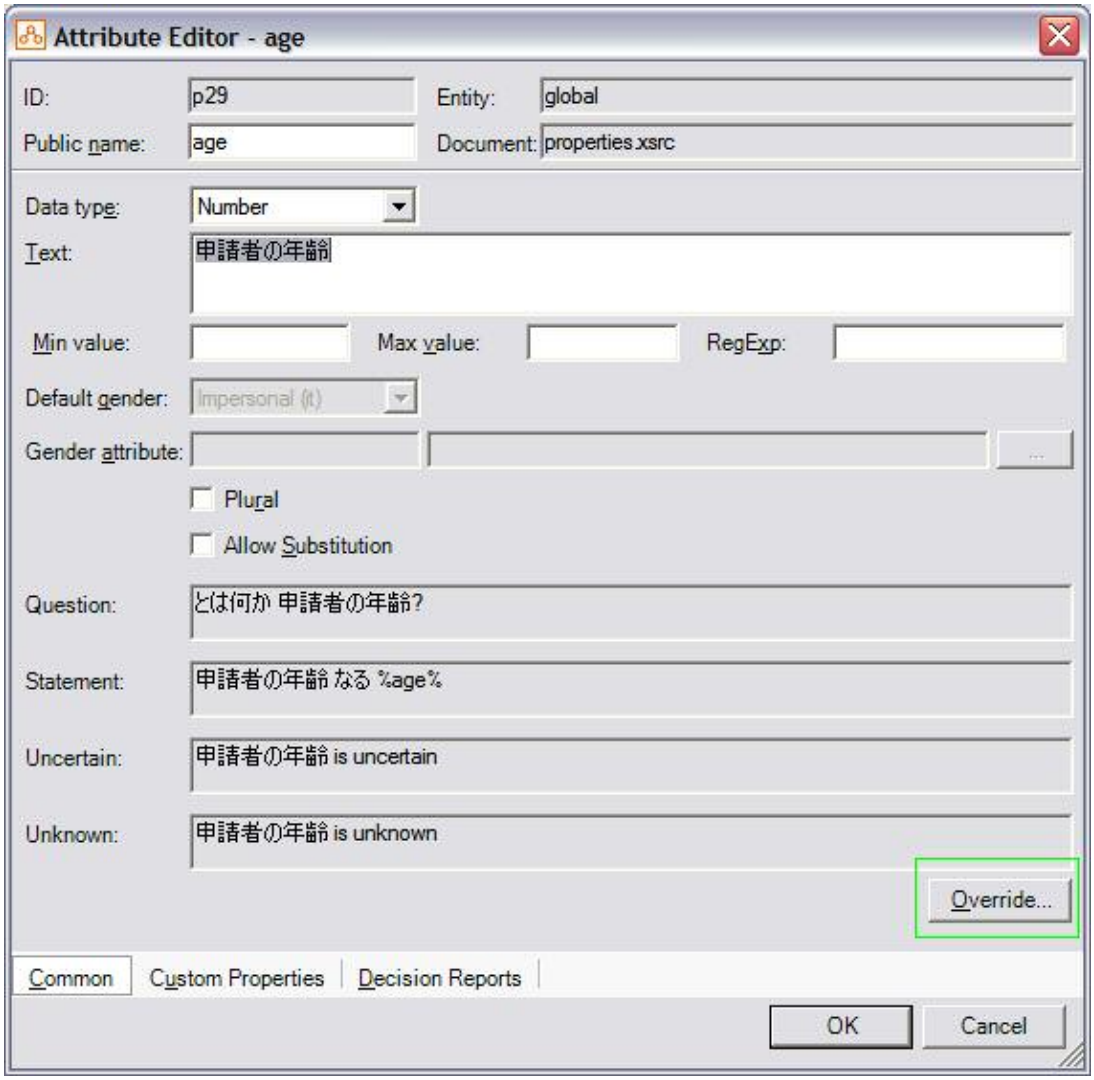


Figure 21 - Sample Properties File in Japanese

How to Override attribute forms:

Assumption: The attribute to be *Overridden* exists in the properties (.xsrc) file

1. Open the properties file.
2. Double-click on the target attribute; the *Attribute Editor* window pops up.
3. Click on the **Override...** button at the bottom right hand corner of the dialog.



4. An *Override Text* window pops up.
    a. Boolean Attributes - the *Override Text* window has three forms to modify, tick the box of a form and modify the value in the field

    b. non-Boolean Attributes – the *Override Text* window has two forms to modify, tick the box of a form and modify the value in the field

5. Click on the **OK** button on the *Override Text* window to save the changes. These changes should be visible in the *Attribute Editor* window

Screens

Creating screens used in a rulebase using an RLS parser is relatively straightforward. Like other components, the screens UI is in English, but any part of the screens UI that displays translated items such as attribute IDs or texts is displayed in the correct character set.

The user also has the options of overriding the *question form* of all attributes when authoring the screens.

What is/can be translated:
question screen names

- attribute IDs and texts
- flow names
- provided values for list inputs
- label text

What is not translated:

- Screens UI; for example, value of UI dropdowns, field labels
- summary/data review screen name cannot be changed – due to specific expectations by Studio

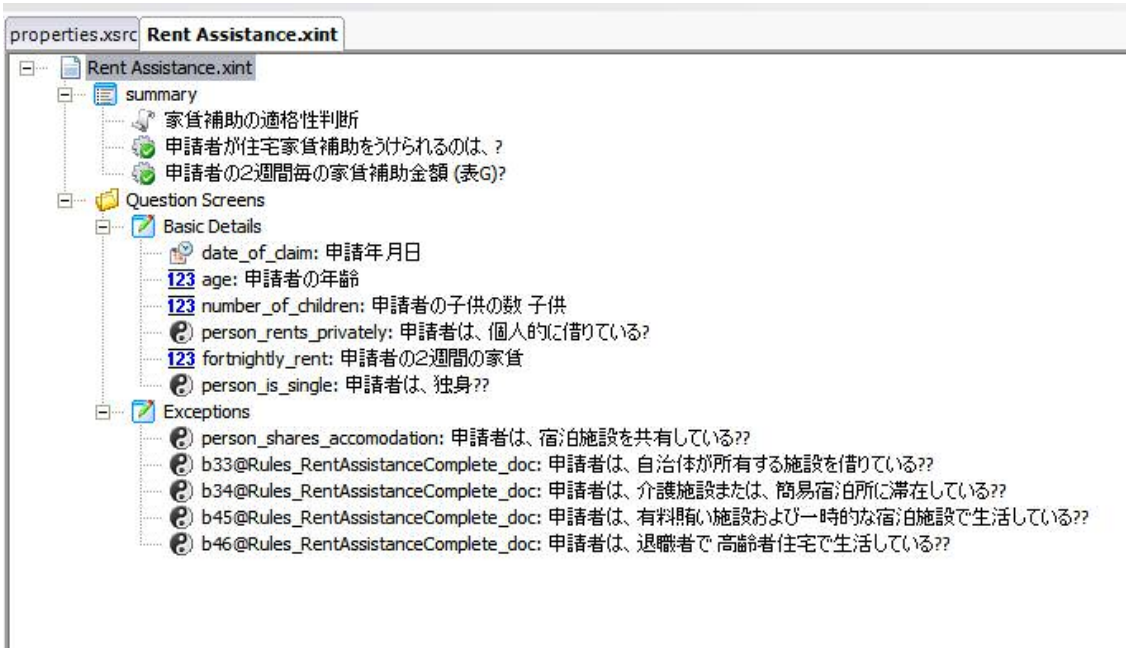


Figure 22 - Sample Screens File in Japanese

Interactive

No changes/configurations are needed for running Interactive with an RLS parser. Texts for screen names, flow names, and attribute IDs/texts are correctly displayed. The static webpage text and inputs are in English.

What is/can be translated:

- attribute IDs and texts
- screen names
- flow title
- entity names and relationships
- list values from screens

What is not translated:

- Interactive static webpage text
- Interactive form inputs and labels

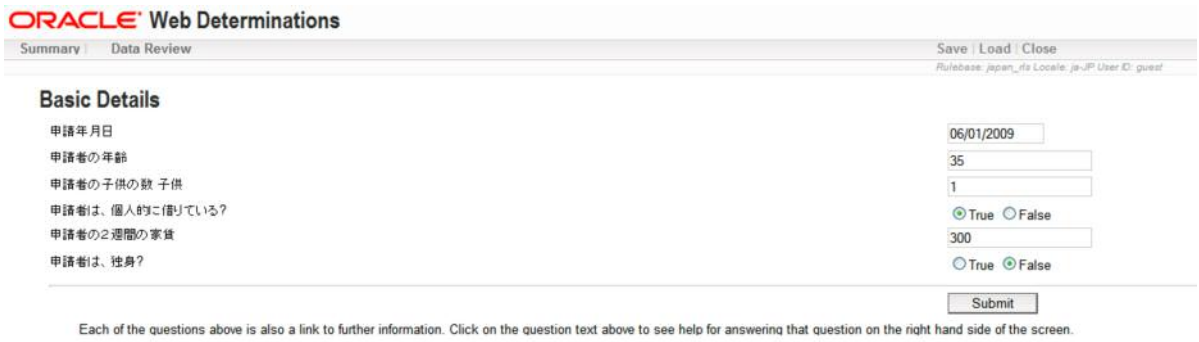


Figure 23 - Sample Interactive Question Screen in Japanese

ORACLE

Debugger

No changes/configurations are needed for running the Debugger with an RLS parser. Texts for screen names, flow names, entity name/relationships, and attribute IDs/texts are correctly displayed. The Debugger UI remains in English.

What is/can be translated:

- attribute IDs and texts
- screen names
- flow title
- entity names and relationships
- search bar (top right area)

What is not translated:

- Debugger UI; for example, form inputs and labels
- keywords and values; for example, true/false/unknown/uncertain