



# OPENRULES DIALOG

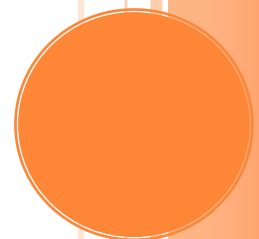
Developing Web-based Questionnaires with OpenRules

## User Manual

ORD is a software product that allows business analysts to develop and maintain web-based dialogs (questionnaires) with complex interaction logic. This tutorial describes main product concepts and components and provides concrete examples of web-based dialogs.

OpenRules, Inc.

May-2019



## Table of Contents

Main Concepts.....	3
Dialog .....	3
Pages .....	4
Sections .....	5
Questions.....	5
Question Types.....	6
Input Validation.....	9
Parent-Child Relationships among Questions .....	10
Question Properties .....	11
Answers to Multi-choice Questions .....	11
Default and Possible Answers.....	11
Defining Combo-boxes through Arrays .....	12
Automatically Calculated Responses .....	13
Navigation Rules .....	14
Update Rules .....	15
Hide/Show Pages, Sections, Questions .....	15
Hide/Show/Reset Children-Questions .....	16
Handling Update Events.....	16
Linking Combo-Boxes.....	17
Custom Questions and Actions.....	18
Generating PDF Documents.....	19
Authorized Access to Questionnaires .....	19

Saving and Restoring Dialog States.....	22
Dialog Templates .....	24
DialogType0 - simple dialog with two navigation buttons .....	24
DialogType1 – adds Login/Logout and Save/Load facilities .....	24
DialogType2 – adds a menu.....	25
DialogType3 – new question on top of the page .....	25
DialogType4 – dynamically changed dialog with parent-child relationships.....	26
Creating Your Own Questionnaire .....	28
Technical Information.....	29
Installation and Configuration .....	29
Supporting Rule Templates and Forms .....	31
Customization and Integration – Developer Notes .....	31

# OPENRULES DIALOG

## *How to Build Web-based Questionnaires*

OpenRules Dialog (ORD) is a software product built on top of the open source Business Rules Management System “[OpenRules](#)”. It allows non-technical people to develop and maintain web-based questionnaires (dialogs) using only Excel and without any necessity to learn different web programming techniques. A questionnaire is a web application that can be described in terms of pages, sections, and different types of questions. Layouts of pages, sections, questions, and complex relationships between them can be expressed in a very natural way using simple and intuitive Excel tables. This document describes the main concepts and provides concrete instructions that help you to install and use the product. It also includes specific examples of different web-based questionnaires.

## MAIN CONCEPTS

OpenRules Dialog (ORD) uses the following concepts to represent different business questionnaires as web applications. Instead of forcing a questionnaire designer to learn different web development techniques, ORD introduces the following basic concepts:

- Dialog
- Pages
- Sections
- Questions
- Answers (default and possible)
- Automatically calculated responses
- Navigation rules
- Update rules.

All these concepts could be presented in one (or several) Excel files in simple tables with predefined structures. Along with a predefined rich set of question types, a user can easily extend the ORD functionality by adding custom question types and dialog actions.

## Dialog

When a user starts a web session to fill out a questionnaire there is always one instance of the object “dialog” that represents the current state of the interaction with a user. The Dialog

object always knows what page a user is currently accessing and what last action has been executed by a user (e.g. clicked on the "Next" button, selected "Yes" for a certain Yes/No questions, click on a certain hyperlink, etc). This information is used by the navigation and update rules defined in the ORD to refresh page content based on the latest answers or to hide/show some sections, questions, calculated automatic responses, etc. As a questionnaire developer you rarely have to deal with the object "dialog" directly but it is effectively used by the ORD templates that specify the interaction logic.

## Pages

ORD considers any questionnaire as a sequence of web pages through which a user can navigate using actions "Next" and "Prev" (usually presented as push-buttons) or any other application specific mechanism such as a menu. Here is an example of the Excel table "pages":

Rules pages extends pagesTemplate						
#	Page ID	Page Name	Hidden	Section Column 1	Section Column 2	Section Column 3
1.1	Page 1	Page Title 1		section11	section12	
1.2				section13		
1.3				section14		
2.1	Page 2	Page Title 2		section21		
2.2				section22		
2.3				section23		
3.1	Page 3	Page Title 3	Yes	section31		
3.2				section32		
4	Page 4	Page Title 4		section4		

As you may guess, this questionnaire consists of 4 pages. Page 1 consists of 4 sections, and section12 is allocated on the right of section11, while section13 goes under the section11, section14 goes under section13, etc. Each page may have (or may not have) a title that will be shown on the top of the page. You may use any string as a page ID as long as all page IDs are unique. The order of pages in this table reflects the natural navigation order through them assuming that every page has navigation buttons "Next" and "Prev". When the order of pages depends on the entered answers or other factors, these special cases can be described in special navigation rules (see [below](#)).

Note. The very first column is usually used for decorative purposes only: it may contain a page order number that has no particular meaning or it may be left empty. However, you cannot remove this column or merge cells inside it.

## Sections

The previous table “**pages**” defines a relative layout for sections inside every page. The sections themselves are defined in the Excel table “**sections**” and may look as follows:

Rules <b>sections</b> extends questionsTemplate						
#	Section ID	Section Name	Hidden	Question Column 1	Question Column 2	Question Column 3
1-3	section11	Section Title 11		question1	question2	question3
4				question4		
5	section12	Section Title 12	Yes	question5		
6				question6		
7				question7		

...

As you can guess, this table defines all sections and their content (questions). For example, the section11 consists of 4 questions. Question2 will be placed on the right of the question1, question3 – on the right of the question2, and question4 goes under the question1. Each section may have (or have not) a title that will be shown on the top of the section. You may use any string as a section ID as long as all section IDs are unique. The column “#” is similar to the one for pages and can contain any information or be empty.

## Questions

The previous table “**sections**” defines a relative layout for questions inside every section. The questions themselves are defined in the Excel table “**questions**” and may look like the following one:

Rules <b>questions</b> extends questionsTemplate					
Question Id	Question Name	Question Type	Size	Hidden	Validation
question1	First name:	TextBox	18		
question2	Middle initial:	TextBox	2		
question3	Last name:	TextBox	20		
question4	Spouse's social security number:	TextBox			SSN
question5	Can somebody claim your spouse on their return?	RadioButton			
wage	Wage	TextBox			Range 1000 10000000

...

You may use any string as a question ID as long as all question IDs are unique. There are no limits for question names, but you should keep in mind that long names could affect automatically adjusted question/section layouts.


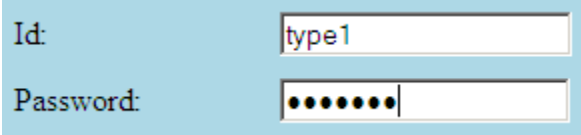
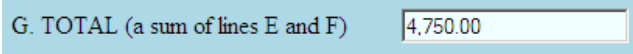

The column “Size” allows you to change the size (number of displayed characters) in a TextBox’s input field. The default size is 20. The validation criteria are described [below](#).


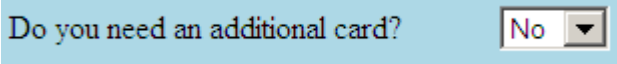
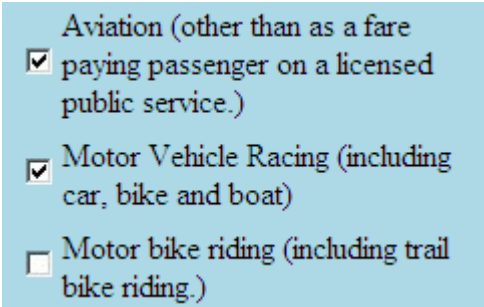
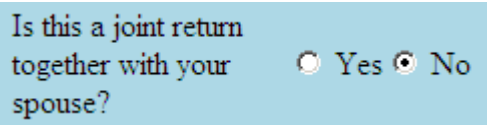
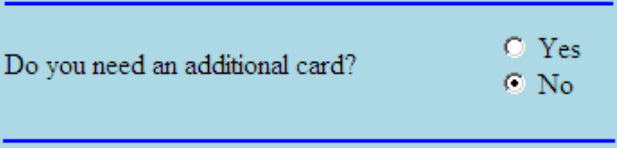
There are no limits to the number of questions and usually all questions for all sections are defined in the table with the name “**questions**”. If there are too many questions you may split them in separate tables that look like this one but have different names, for example “questions1” and “questions2”. Then you have to add another table with the name “questions” that will invoke these two tables:

```
Method void questions(Section section, String questionId)
questions1(section, questionId);
questions2(section, questionId);
```

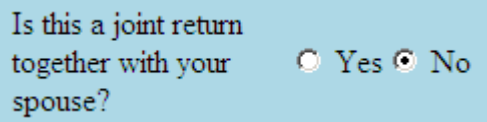
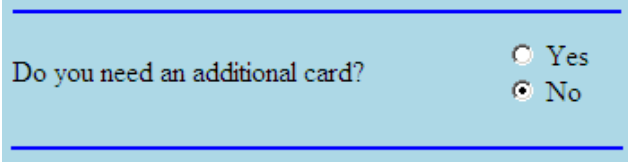
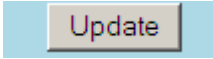

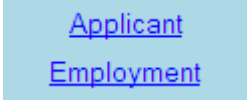
### Question Types

The current version of OpenRules Dialog supports the following question types:

Question Type	Comment	Example
TextBox	A regular question with a label (name) and an input field of a certain size	
Password	Similar to TextBox but the entered text would be display as stars	
ReadOnly	Similar to the TextBox but the input field is automatically calculated and cannot be changed (indicated by a slightly different background – Azure)	
TextArea	Similar to the TextBox but the input field may consist of several lines	

Question Type	Comment	Example
ComboBox	Label with input that allows selection from possible answers	
ComboBox Submit	The same as a regular ComboBox but refreshes the page upon a selection	 <p>Selection of “Yes” may show a hidden information for an additional card</p>
ComboBox Domain	Similar to ComboBox but allows you to define different display and actual choices	<p>You may specify the question’s domain as “1,Yes;0,No”. Then the combo-box will look like above but when a user selects “Yes” the actual answer is “1”. When a user selects “No” the actual answer is “0”.</p>
CheckBoxButton	Allows the selection of one or more multiple alternatives. Each alternative should be defined by a separate check button. The question’s name is used as the check button text. The question’s ID is used as a part of the name of the proper action, e.g. Check-Aviation or Uncheck-Aviation	
RadioButton	Allows the selection of one choice from among multiple alternatives	
RadioButton Vertical	The same as a regular RadioButton but places the choices vertically	



Question Type	Comment	Example
RadioButtonSubmit	Allows the selection of one choice from among multiple alternatives and submits this choice to the server to refresh other pages, e.g. to hide/show related questions	
RadioButtonVerticalSubmit	The same as a regular RadioButtonSubmit but places the choices vertically	
RadioButtonVerticalDomain	Similar to RadioButtonVertical but allows you to define different display and actual choices	You may specify the question's domain as "1,Yes;0,No". Then the buttons will look like above but when a user selects "Yes" the actual answer is "1". When a user selects "No" the actual answer is "0".
ActionButton	Creates a push-button whose name could be associated with some action defined in navigation rules.	 <p>If no action is defined, pushing (selecting) the button will force the page to be refreshed with a possible recalculation of all auto-responses</p>
ActionImage	Creates a push-button with a custom image whose path is defined by the question's name. The question ID is used to define the last user action that could be used in the navigation rules.	 <p>An image that represents a delete button</p>
ActionHyperlink	Creates a hyperlink whose text corresponds to the question's name. The question ID is used to define the last user action that could be used in the navigation rules.	 <p>Can be used to represents different menus</p>

Question Type	Comment	Example
Message	Displays a text defined as the question name	Your credit card application has been submitted.
Empty	Creates an empty question that can be used as a cell filler	
Custom	This question can be defined by a customer using the rules table “customQuestions”	<a href="#">Click here to Preview and/or Print Your Tax Return using Adobe Acrobat</a> This custom hyperlink generates and displays a PDF document base on the answered questions

### Input Validation

You may associate a validation criterion with any TextBox – see for example “SSN” and “Range 1000 100000000” in the above table “questions”. ORD will compare the entered answer with the question’s validation criterion, and if it is violated, ORD will place an error message (in red) right below the invalid answer. Here is the list of currently supported validation criteria:

Validation Criterion	Comment	Example of the Error Message
RANGE <from> <to>	Validates that the answer to the question is an integer or real number that should be within the interval [from;to]	<b>ERROR: The answer should be within 1 and 100</b>
DATE	Validates that the answer is a valid date for the current locale, e.g. 20/5/2009 is invalid for the US locale	<b>ERROR: invalid date format for the current locale</b>
EMAIL	Validates that the answer is a valid email address	<b>ERROR: invalid email address</b>
URL	Validates that the answer is a valid URL address	<b>ERROR: invalid URL address</b>
SSN	Validates that the answer has a valid Social Security Number format	<b>ERROR: SSN should have a format 999-99-9999</b>

CREDITCARD	Validates that the answer has a valid CREDIT Card Format	<b>ERROR: invalid credit card number</b>
REGEX <pattern>	Validates if the answer matches the regular expression defined by regular expression “pattern”, e.g. SSN corresponds to the regular expression "[0-9]{3}-[0-9]{2}-[0-9]{4}"	<b>ERROR: &lt;answer&gt; doesn't match the pattern &lt;pattern&gt;</b>
USERID	Validates if the answer is a valid user ID by calling a custom method “ <i>validateUserID</i> ”	<b>ERROR: invalid user ID</b>
PASSWORD	Validates if the answer is a valid password by calling a custom method “ <i>validatePassword</i> ”	<b>ERROR: invalid password</b>

More validation criteria could be added without programming (see the file DialogForms.xls).

Another way to validate user input is to add special conditions directly in the table “[navigateRules](#)” – see an example in the Dialog1040EZ project.

### Parent-Child Relationships among Questions

Some dialogs are hierarchical by nature when some questions should be shown only when their parent-questions are shown and/or have certain answers. If you hide a parent-question then all its “children” and all their descendants (if any) should also be hidden. For example, consider the following dialog:

It was created using the following table “questions” with a column “A5” specifying parents for some questions, e.g. the question “WIRE1” is a parent for the questions “WIRE1.1”:

Rules questions extends questionsTemplate					
C1	A1				A5
Question Id	Question Name	Question Type	Size	Hidden	Parent Question
WIRE1	What type of wire transfers do you anticipate in this account?	ComboBoxSubmit			
WIRE1.1	-- What type of wires do you anticipate receiving in this account?	ComboBoxSubmit			WIRE1
WIRE1.1.1	---- Which Country(s) will incoming international wires come from? (Check all that apply)	ComboBox			WIRE1.1
WIRE1.1.2	---- What is the expected monthly dollar amount of incoming international wires?	ComboBox			WIRE1.1
WIRE1.1.3	---- What is the source of incoming international wires? (Check all that apply)	ComboBoxSubmit			WIRE1.1
WIRE1.1.3.1	----- Specify "Other":	TextBox		Yes	WIRE1.1.3
WIRE1.1.4	---- What is the expected monthly dollar amount of incoming domestic wires?	ComboBox			WIRE1.1

By default, when you hide a question, all its descendants will be also hidden. You may define your own logic for hiding, showing, or resetting questions and its children using different actions in the standard table "updateRules" as described below.

### Question Properties

You may associate with each question certain properties by adding additional columns to the table "questions". For example, in the sample project DialogType3 we added properties "Country", "Category", and "Comments" to the table "questions":

Rules questions extends questionsTemplate								
C1	A1					A2		
Question Id	Question Name	Question Type	Size	Hidden	Validation	Country	Category	Comments
ID	User Id:	TextBox			USERID			
Password	Password:	Password			PASSWORD			
Login	Login	ActionButton						
NewOldDialog	Restore previous dialog session?	RadioButton						
Country	Country:	ComboBox						
Category	Category:	ComboBox						
Q11	Page 1 Question 1	YesNo				.	.	Comments for Page 1 Question 1
Q12	Page 1 Question 2	YesNo				DE,FR	.	Comments for Page 1 Question 2
Q13	Page 1 Question 3	YesNo				US,FR	.	Comments for Page 1 Question 3

Now we may display only questions that correspond to certain countries and categories and produce additional comments when we receive answers to certain questions. See the project DialogType3 for more details.

### Answers to Multi-choice Questions

You may define possible answers for multi-choice questions using combo-boxes.

### Default and Possible Answers

You may define the default and possible answers for some or all questions using an Excel table similar to this one:



Data Answers answers		
Question Id	Default Answer	Possible Answers
YourFirstName	John	
YourMiddleInitial	N.	
YourLastName	Smith	
AddressLine1	25 Maple Street	
AddressLine2	Apt. 3C	
City	Edison	
State	NJ	USstates
ZipCode	08840	
YourSSN	164-86-2298	
MarriedFillingJointly	No	yesno

Here the possible answers “USstates” and “yesno” refer to the custom Excel Data table “possibleAnswers”:

Data PossibleAnswers possibleAnswers	
id	choices
ID	Answers
employmentTypes	Employed
	Unemployed
	Full Time Student
	Retired
gender	Male
	Female
yesno	Yes
	No
USstates	AL
	AK
	AR
	...

### Defining Combo-boxes through Arrays

There is another way to define possible answers for combo-boxes. Possible answers may be defined in Data array using predefined Data types such as ArrayString – see

[http://openrules.com/docs/man\\_data.html#Predefined Datatypes](http://openrules.com/docs/man_data.html#Predefined Datatypes):

Data ArrayString LOBs	
value	
LOB	
HMO	
EPO	
PPO	

You may have many arrays like this one and they can be used as possible answers for some combo-boxes in the following way:

Rules questions extends questionsTemplate						
C1	A1					A3
Question Id	Question Name	Question Type	Size	Hidden	Validation	Answers
Q1_LOB	LOB	ComboBox			NO_EMPTY	:=getArray(d,"LOBs")
Q1_InOut	In or Out Network	ComboBox			NO_EMPTY	:=getArray(d,"InOutNetworks")
Q1_Location	Service Location	ComboBox			NO_EMPTY	:=getArray(d,"ServiceLocations")
Q1_Service	Service	ComboBox			NO_EMPTY	:=getArray(d,"Services")

Here the table “questions” uses an optional column “A3” from the template “questionsTemplate”. It requires that the cells in this column have the standard OpenRules type ArrayOfStrings. You may convert arrays like LOBs to the type ArrayOfStrings using the method “getArrayOfStrings” defined in the Dialog.xls. Then for efficiency reason (and for better expressiveness) you may add them one table with unique names:

Rules void createArraysOfAnswers(Dialog d)	
A1	
d.put(id,array);	
String id	ArrayOfStrings array
ID	Array of Strings
LOBs	:=getArrayOfStrings(LOBs)
InOutNetworks	:=getArrayOfStrings(inOutNetworks)
ServiceLocations	:=getArrayOfStrings(serviceLocations)
Services	:=getArrayOfStrings(services)

The method:

Method ArrayOfStrings getArray(Dialog d, String id)
return (ArrayOfStrings) d.get(id);

is used within above table “questions” to define possible answers for combo-boxes.

## Automatically Calculated Responses

Answers to some questions could be automatically calculated based on the answers to other questions and possible other information (for example, data coming from a database). The rules for such auto-responses should be placed in the table “*autoResponses*” that may look like this one:

Rules autoResponses extends autoResponsesTemplate	
Question Id	Auto Response
AdjustedGrossIncome	<pre>{ double wages = d.getDoubleAnswer("Wages"); double taxableInterest = d.getDoubleAnswer("TaxableInterest"); double unemploymentCompensation =     d.getDoubleAnswer("UnemploymentCompensation"); double answer = wages + taxableInterest +     unemploymentCompensation; format(answer); }</pre>
A	<pre>{ double wages = d.getDoubleAnswer("Wages"); format(wages + 250); }</pre>
B	:= format(750)
C	<pre>{ double a = d.getDoubleAnswer("A"); out("A="+a); double b = d.getDoubleAnswer("B"); out("B="+b); format(Math.max(a,b)); }</pre>
D	<pre>{ double answer = 4750; Question marriedFilingJointly = d.getQuestion("MarriedFilingJointly"); if (marriedFilingJointly.isAnswerTrue())     answer = 9500; format(answer); }</pre>

## Navigation Rules

ORD provides a built-in support for navigation between pages. The order of pages in the table “**pages**” reflects the natural navigation order assuming that every page has navigation buttons “Next” and “Prev”. When the order of pages depends on the entered answers or other factors, these special cases can be described in special navigation rules table “*navigationRules*” that may look like the tables below:

Rules navigationRules extends navigationTemplate					
C1	C2	C3			A1
IF Current Page is	AND Action is	AND			THEN Go to Page
		Answer to Question	Is or IsNot	Value	
TaxpayerGeneral Information	Next	MarriedFilingJointly	IsNot	Yes	IncomeData
IncomeData	Prev	MarriedFilingJointly	IsNot	Yes	TaxpayerGeneral Information

You may create different navigation conditions and actions from the template “*navigateDialogTemplate*” described in the configuration file *DialogRules.xls*.

## Update Rules

ORD allows a questionnaire designer to define special conditions and execute the proper actions when the content of certain pages is updated by a user. For example, the update rules define conditions when some pages, sections or questions should be hidden or shown. The updating logic can be defined in the decision table “*updateRules*” that may look like this one:

Rules updateRules extends updateDialogTemplate									
C1	C2	C3			A1		A2		A4
IF Current Page is	AND Action is	AND			THEN Hide/Show Question		AND Hide/Show Section		AND Set Dialog Status
		Answer to Question	Is or IsNot	Value	Hide	Question	Hide	Section	
Page2	Refresh								Refreshed
Page3		Q32	Is	Retired	Show	Q321			
Page3		Q32	IsNot	Retired	Hide	Q321			
Page4		Q44	Is	Yes			Show	Section5	
Page4		Q44	Is	No			Hide	Section5	

This decision table is created based on a template “*updateDialogTemplate*” defined in the standard file “*DialogRules.xls*”.

## Hide/Show Pages, Sections, Questions

The updating rules allow you to effectively implement hide/show features on any level: page, section, or question. The standard Dialog object “d” supports the following methods:

- `hidePage(String pageID)`
- `showPage(String pageID)`
- `hideSection(String sectionID)`
- `showSection(String sectionID)`
- `hideQuestion(String questionID)`
- `showQuestion(String questionID)`

You may invoke these methods as a code in the column “Execute Code” or use more convenient columns “Hide/Show Page”, “Hide/Show Section”, and “Hide/Show Question”. Naturally, when you invoke the method “hidePage” it hides all sections in this page. When you invoke the method “hideSection” it hides all questions inside this section. Conversely, when you call “showSection” it shows all questions from this section unless some of them were specifically hidden earlier.



You may also use this table to respond to a question X with an answer Y, set the dialog status, and much more – see the template “updateDialogTemplate” in the file dialogRules.xls.

Note. For efficiency reasons the hidden pages, sections or questions are not analyzed during the page “refresh” that occurs during a user session whenever ORD delivers a new page or updates the current one.

### Hide/Show/Reset Children-Questions

When you hide a question, all its children (and “grand-children) also will be hidden. The following “updateRules” table shows different conditions and actions available to you to express your own business logic for hiding, showing or resetting questions and its descendants:

Rules updateRulesForPageAccounts3 extends updateDialogTemplate										
C5	C3			A1		A6		A10	A11	
IF Update Event is	IF			Hide/Show Question		Hide/Show Children of the		Reset Default Answers for Question and its children	Reset Default Answers for children of the Question	
	Answer to Question	Is or IsNot	Value	Hide	Question	Hide	Question			
CASH1									CASH1	
	CASH1	Is	[Select]			Hide	CASH1		CASH1	
	CASH1	Is	Incoming	Show	CASH1.1					
			Hide	CASH1.2					CASH1.2	
			Outgoing	Show	CASH1.2					
			Hide	CASH1.1					CASH1.1	
		Both		Show	CASH1.2					
			Show	CASH1.1						

Hiding and resetting actions work recursively for all children and grandchildren questions. For example, the action “Hide CASH1.1” in the column A1 will hide the questions CASH1.1 and all its descendants. The actions “Hide CASH1” in the column A6 will hide all children of the question CASH1 but not the question itself. The actions in the columns A10 allow you to reset default answers to a question and all its children (for A10) but the actions in the column A11 reset only the children (descendants) of the question. You may see various examples of such rules in the standard sample project [DialogType4](#).

### Handling Update Events

During an interaction with a user your application may catch certain “update events” initiated by an end user. The updating rules allow you to validate if some questions were answered during the latest interaction cycle by selecting values from a combo-box or by clicking on a radio-button. The following questions produce update events:

- ActionButton
- ComboBoxCommit
- RadioButtonCommit
- YesNoSubmit.

Such update events carry the names of the proper questions. For example, in the above table “updateRules” the condition C5 validates if the user selected a value from the combo-box “CASH1” and the proper update event “CASH1” was actually generated by ORD. If yes, then this particular update-logic will reset all children-questions of the question “CASH1” to their defaults (as described in the action A11).

## Linking Combo-Boxes

The updating rules allow you to link several combo-boxes in such a way that when a user makes a selection in one combo-box the choices in other combo-boxes may be modified. For examples, let’s define two questions Q35 “Car Make” (BMW, Lexus, Acura, ...) and Q36 “Car Model” (5 Series, X3, X5, GS 350, RL, TL, ...). Each question is a combo-box but when a user selects “BMW” for the first question we want to display only BMW’s models as possible choice for the second question.

Here is one possible way to do it. First, we will add different car makes to the list of possible questions used to define Q35:

Data PossibleAnswers possibleAnswers	
ID	Answers
employmentTypes	Employed
	Unemployed
	Full Time Student
	Retired
gender	Male
	Female
yesno	Yes
	No
Car-Makes	Select
	BMW
	Lexus
	Acura

Data Answers answers		
Question Id	Default Answer	Possible Answers
Q21	11371	
Q31	MA	USstates
Q32	Employed	employmentTypes
Q33	Female	gender
Q35		Car-Makes

Both questions Q35 and Q36 will belong to Page3 and are defined as a ComboBox. However, to make sure that Page3 will be updated every time a user makes a selection inside the Q35 combo-box, this question should be defined as “ComboBoxSubmit”!

To define the dynamic content of the question Q36 “Car Model”, we will define 3 arrays of possible models for BMW, Lexus, and Acura:

```
Method String[]
getBMWModels()

return ArrayString.getValues
(BMWModels);
```

```
Method String[]
getLexusModels()

return ArrayString.getValues
(LexusModels);
```

```
Method String[]
getAcuraModels()

return ArrayString.getValues
(AcuraModels);
```

Data ArrayString BMWModels	
Models	
Select	
1 Series	
3 Series	
5 Series	
X3	
X5	

Data ArrayString LexusModels	
Models	
Select	
ES 350	
GS 350	
GS 450	
GS 450h	
LS 460	

Data ArrayString AcuraModels	
Models	
Select	
MDX	
RDX	
RL	
TL	
TSX	
ZDX	

To link these arrays to question Q36 to generate different answers for question Q35, we expand table “UpdateRules” as follows:

Rules updateRules extends updateDialogTemplate											
C1	C2	C3			A1		A2		A4	A7	
IF Current Page is	AND Action is	AND			THEN Hide/Show Question		AND Hide/Show Section		AND Set Dialog Status	AND Limit Answers to	
		Answer to Question	Is or IsNot	Value	Hide	Question	Hide	Section		Question	to Array of Strings
Page2	Refresh								Refreshed		
Page3		Q32	Is	Retired	Show	Q321					
Page3		Q32	IsNot	Retired	Hide	Q321					
Page4		Q44	Is	Yes			Show	Section5			
Page4		Q44	Is	No			Hide	Section5			
Page3		Q35	Is	BMW						Q36	getBMWModels
Page3	Lexus								getLexusModels		
Page3	Acura								getAcuraModels		

Another action “A7” was used in the last three rules that say exactly what we need, e.g.:

*IF Answer to Question Q35 is <BMW>  
THEN Limit Answers to Question Q36 to Array <getBMWModels>*

See the sample project DialogType0 for the actual implementation. Similarly we can connect more combo-boxes. More sophisticated ways to define complex inter-question relationships are under development and will be provided in the next ORD releases.

## Custom Questions and Actions

Along with predefined question and action types you may create your own questions/actions using type “Custom”. To do this you have to define the proper layout in the table “*customQuestions*” similar to the following Excel tables:

Rules <b>customQuestions</b> extends customQuestionTemplate	
Question Id	Custom Question Layout
GeneratePDF	:= layoutGeneratePDF(q.dialog)

Layout TableLayout layoutGeneratePDF(Dialog d)
<pre>&lt;a href="{makePDFHref(d)}" target="_blank"&gt; Click here to Preview and/or Print Your Tax Return using Adobe Acrobat &lt;/a&gt;</pre>

This layout invokes a separately described Excel table “makePDFHref” that actually maps the answered questions to the proper PDF form. Your layout may execute your own methods such as any java method.

## Generating PDF Documents

OpenRules Dialog includes a special module "**pdfgen**" that allows you to generate PDF documents based on your filled out questionnaire. You may look at the sample project “Dialog1040EZ” that explains how to generate a pdf document. This example is based on the well-known US tax form 1040EZ. The application takes the standard 1040EZ pdf form publicly available from the IRS web site and fills it in after an interactive session with a taxpayer. The detailed tutorial Dialog100EZ can be found [here](#).

The **pdfgen** module uses 3rd party PDF generation software developed by Big Faceless Organization (BFO) - see <http://big.faceless.org>. An evaluation version of the BFO software is included. However, for production implementations you will need a commercial copy of the BFO software that could be purchased directly from BFO or from OpenRules.

## Authorized Access to Questionnaires

Usually an authorized access is provided through Login/Logout functionality. Here is an example of a Login screen:

This screen can be easily implemented as the very first page of your questionnaire with one section “Login”:

Rules sections extends sectionsTemplate				
#	Section ID	Section Name	Hidden	Question Column 1
	Login	Login		ID
				Password
				NewOldDialog
				Login

The proper questions may be defined in the table “questions” as follows:

Rules questions extends questionsTemplate					
Question Id	Question Name	Question Type	Size	Hidden	Validation
ID	Id:	TextBox			USERID
Password	Password:	Password			PASSWORD
Login	Login	ActionButton			
NewOldDialog	Restore previous dialog session?	RadioButton			

There are two special validation criteria USERID and PASSWORD that are associated with two custom (user-defined) validation methods. These methods should be specified in a separate rules table “*customValidate*” in the file Main.xls:

Rules customValidate extends validateAnswerTemplate		
IF Validation Pattern	AND Answer Defined	THEN Validation Result
PASSWORD		:= validatePassword(q)
USERID		:= validateUserID(q)

Here we used two validation methods: *validateUserID* and *validatePassword*. These methods have a predefined signature but the actual implementation depends on the way you want to keep you user identification information. For example, in the template-project [DialogType2](#)

we use the following implementation placed in the file Main.xls. First, we define the datatype DialogUser:

Datatype DialogUser	
<b>String</b>	<b>id</b>
<b>String</b>	<b>password</b>

Then we create the table “users” that contains different pairs “UserID” and “Password”:

Data DialogUser users	
id	password
User ID	Password
Robinson	1234
Smith	4567
Green	1111

Our validation methods should try to find the answers to the questions “ID” and “Password” in the array “users”. To find out if a user with a certain ID is located in the array “users” we created a simple method:

```
Method DialogUser findDialogUser(String id)
for(int i=0; i<users.length; i++) {
    if (users[i].id.equals(id))
        return users[i];
}
return null;
```

Now we can implement the method “validateUserID” as follows:

```
Method String validateUserID(Question question)
Dialog d = question.dialog;
String msg = "";
if (d.isAction("Init"))
    return msg;
DialogUser user = findDialogUser(question.answer);
if (user == null) {
    d.addError();
    msg = "ERROR: Invalid User ID";
}
return msg;
```

If there are no users whose ID corresponds to an answer to the question “ID”, this method will add the proper error to the Dialog and this error will be displayed during Login.

Similarly one can define the method “validatePassword”:

```

Method String validatePassword(Question question)
Dialog d = question.dialog;
String msg = "";
if (d.isAction("Init"))
    return msg;
String id = d.getAnswer("ID");
DialogUser user = findDialogUser(id);
if (user == null || !user.password.equals(question.answer)) {
    d.addError();
    msg = "ERROR: Invalid password";
}
return msg;

```

The actual business logic that validates the entered ID and Password depends on your specific implementation which can be realized in two methods similar to the described ones. For example, you may associate the list of users and their password with a table in your database using simple Excel import facilities. Or you may use Java methods that “talk” to a database through a JDBC interface.

In all cases if an answer to the question “ID” or “Password” is invalid, the appropriate message will be displayed and a user will remain on the “InitialPage” with the Login button. When the answers are valid, a user will see the next page “Page1” as defined in the table [“navigateRules”](#).

Note. The question “Restore previous dialog session?” allows a user to start filling in the questionnaire from scratch (“No”) or to restore previously entered answers (“Yes”).

## Saving and Restoring Dialog States

While your web interface will guide a user through a set of pages and questions, the user does not have to answer all the questions during one session. The user may stop at any point and all entered information may be saved. The user may start a new session after s/he obtains more data and is ready to enter it.

In the Login screen on the picture [above](#) we used the question “Restore previous dialog session?” If a user selects “Yes” as an answer to this question, the last saved state of this questionnaire (if any) will be restored and all already answered questions will have previously defined answers. Otherwise, a user will start filling in the questionnaire from scratch. How might this functionality be implemented?

In the template project DialogType2 we defined a reaction to the answer “Yes” for this question in the table [“navigateRules”](#) as a special action:

```
:= loadDialog( d, d.getAnswer("ID"))
```

The method *loadDialog* (as well as its counterpart *storeDialog*) is defined in the file Main.xls in the following way:

```

Method String loadDialog(Dialog d,String id)

String filename = "./work/" + id + ".answers";
if (d.load(filename) == null) {
    d.setStatus("ERROR: Unable to load dialog from " + filename);
    d.addError();
}
else
    d.setStatus("Dialog has been reloaded from " + filename);
return filename;

```

It loads answers to all questions as they were stored in the file “*Robinson.answers*” where “*Robinson*” is the entered user’s ID. This file is located in the subdirectory “work” of the “basedir” directory of the web server, on which the web application was deployed. In particular, it can be found in the Tomcat’s “*webapps/work*” subdirectory.

To store the current state of the questionnaire, a user may select an action such as “Save” that is defined as a hyperlink “Save” in the template-project DialogType2:

DialogType2	
Menu	Page 4
<a href="#">Page1</a>	
<a href="#">Page2</a>	<b>Radio Buttons</b>
<a href="#">Page3</a>	RadioButton <input checked="" type="radio"/> Yes <input type="radio"/> No
<a href="#">Page4</a>	RadioButtonSubmit <input type="radio"/> Yes <input checked="" type="radio"/> No
<a href="#">Page5</a>	RadioButtonVertical <input checked="" type="radio"/> Yes
<a href="#">Save</a>	<input type="radio"/> No
<a href="#">Logout</a>	<input type="radio"/> Yes
	<input checked="" type="radio"/> No
	Show Check Buttons? (RadioButtonVerticalSubmit)

A reaction to the action “Save” is defined in the table “*navigateRules*” as a special action:

```
:= storeDialog( d, d.getAnswer("ID"))
```

The method ‘*storeDialog*’ is defined in the file Main.xls as the following:

```

Method String storeDialog(Dialog d,String id)

String filename = "./work/" + id + ".answers";
if (d.store(filename) == null) {
    d.setStatus("ERROR: Unable to store dialog at " + filename);
    d.addError();
}
else
    d.setStatus("Dialog has been stored at " + filename);
return filename;

```



A user may use different names for the methods or completely change the way a dialog is being stored and loaded. The standard class Dialog provides the following convenience methods that give a user some flexibility:

```

store(String fileName)
load(String fileName)
storeToXML(OutputStream outStream)
loadFromXML(InputStream outStream)
store(OutputStream outStream)
load(InputStream outStream)

```

In particular, storing the questionnaire answers in an XML file may simplify its interaction with other applications.

## DIALOG TEMPLATES

ORD comes with a set of template-projects that allow a user to build different types of questionnaires. These templates have been created by OpenRules and/or contributed by customers. Below we describe the currently available templates with their sample views.

### DialogType0 - simple dialog with two navigation buttons

This template includes all types of questions and two buttons “Next” and “Prev” that allow you to navigate through pages. Here is a typical view:

DialogType0	
Question with an integer or double answer in a range from 1 to 99999	<input type="text" value="11371"/>
Email	<input type="text" value="info@openrules.com"/>
Date (mm/dd/yy)	<input type="text" value="02/29/2009"/>
ERROR: invalid date format for the current locale	
Social Security Number	<input type="text" value="111-99-9999"/>
Phone (999-999-9999)	<input type="text" value="732-993-3131"/>
TextBox with expected format: REGEX [0-9]{3}-[0-9]{3}-[0-9]{4}	<input type="text"/>

### DialogType1 – adds Login/Logout and Save/Load facilities

DialogType1 demonstrates how to implement an authorized access with Login/Logout functionality. A user may Save the current state of the interaction and later Restore this state (during Login) with the saved answers. Here is a typical view:

**DialogType1**

**Welcome to ORD**

---

**Login**

Id:

Password:

**ERROR: Invalid password**

Restore previous dialog session?  Yes  No

[OpenRules](#)

## DialogType2 – adds a menu

In addition to the DialogType1 features, the DialogType2 demonstrates how to create a menu as in the following view:

**DialogType2**

**Page 3**

---

**Menu**

- [Page1](#)
- [Page2](#)
- [Page3](#)
- [Page4](#)
- [Page5](#)
- [Save](#)
- [Logout](#)

**Combo-Boxes**

State (ComboBox)

Employment Type (ComboBoxSubmit)

Retirement Date

Gender (ComboBox)

YesNo (ComboBoxSubmit)  Yes  No

## DialogType3 – new question on top of the page

This is a different type of dialog (the layout was proposed by Ronnie Barkan. While supporting the same capabilities as the DialogType2 template, this dialog allows a user:

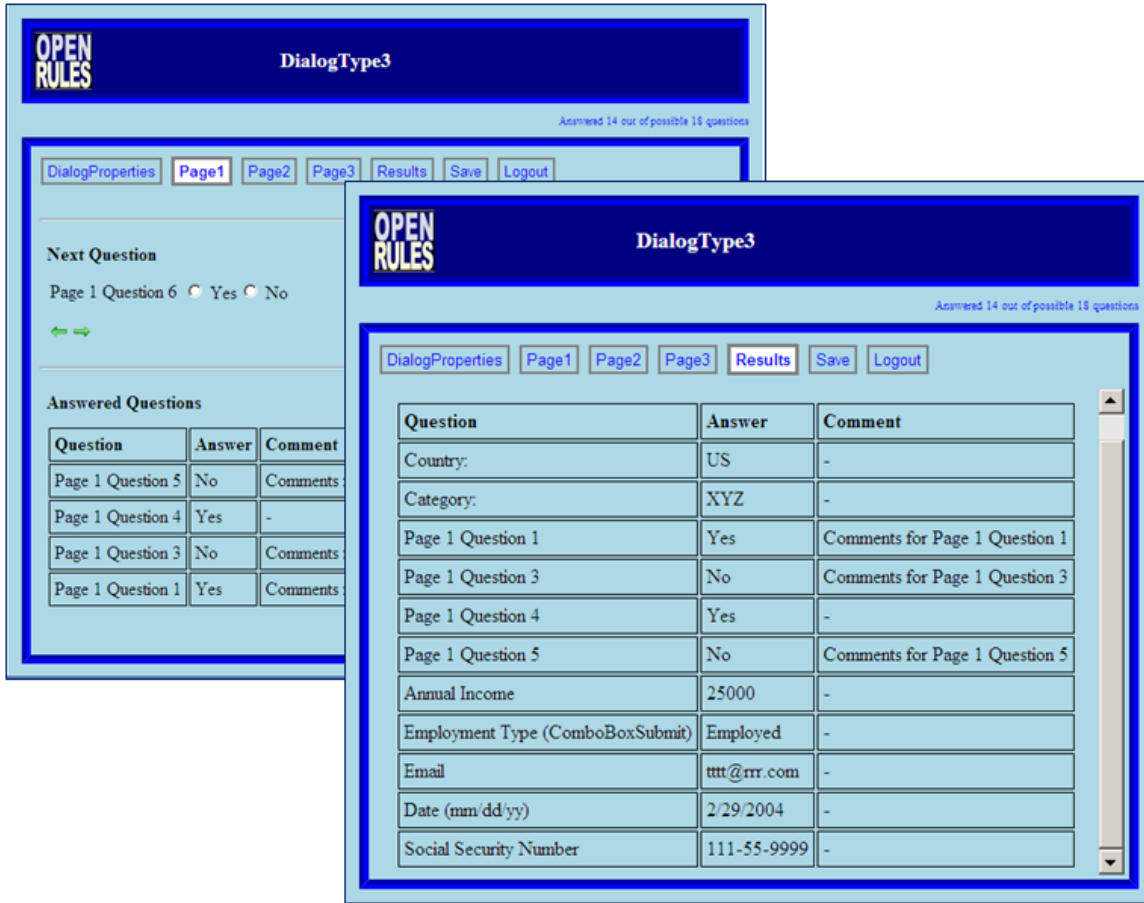
- To enter one question at a time. This new question is always shown on the top of the current page
- To show already answered questions at the bottom of the page starting with the last answered question.

Additionally, DialogType3 demonstrates how:

- To assign properties (such as “Country”) to different questions and to ask only questions with certain properties

- To generate a results page that displays all answered questions (and possibly gives additional information associated with those questions properties.)

Here are typical views:



## DialogType4 – dynamically changed dialog with parent-child relationships

This type of dialogs consists of a single, highly dynamic page that may constantly modify its content based on the user selections. In particular, in the DialogType4 there is one main page that represents a list of customer's accounts. When a user selects an account its content is being downloaded and based on the account type, different contents of the page will be displayed. This sample demonstrates how to:

- Load/Save the selected data

- Refresh the main page without necessity to create a separate page in the standard ORD table “pages”
- Hide/Show/Reset different questions based on the user answers to the parent questions
- Handle user-specific update events.

**OPEN BANK** OpenRules-based Example of Dynamically Modified Web Dialog **OPEN RULES**

Customer: OpenRules

**Summary of Accounts**

Select	Account #	Account Name	Comments
<input checked="" type="radio"/>	00012	Business Checking	3 more questions remains
<input type="radio"/>	00014	Business Saving	to be filled

**Selected Account: 00012 - Business Checking**

CASH ACTIVITY  MONETARY INSTRUMENTS  WIRE TRANSFERS

What type of wire transfers do you anticipate in this account?

-- What type of wires do you anticipate receiving in this account?

---- What is the expected monthly dollar amount of incoming domestic wires?

---- What is the source of incoming domestic wires? (Check all that apply)

----- Specify "Other":

-- What type of wires do you anticipate sending from this account?

---- Which country(s) will outgoing international wires be sent to? (Check all that apply)

---- What is the expected monthly dollar amount of outgoing international wires?

---- What is the purpose of outgoing international wires? (Check all that apply)

---- What is the expected monthly amount of outgoing domestic wires

---- What is the purpose of outgoing domestic wires? (Check all that apply)

----- Specify "Other":

The complete sample application “DialogType4” is included in the standard ORD installation – see the folder “openrules.dialog”.

## CREATING YOUR OWN QUESTIONNAIRE

To create your own questionnaire using the Eclipse IDE, follow these steps:

- 1) Copy and paste any template-project DialogType<X> under your own name, say “DialogMy”
- 2) In the file “*build.properties*” replace `deploy.name= DialogType<X>` to `deploy.name=DialogMy`
- 3) In the file “*run.html*” replace DialogType<X> with DialogMy in the proper URL address. Also, you may adjust the title of your application in this simple html-launcher. If you acquired a commercial ORD license, enter it as a parameter `license=<your-license-code>` instead of `license=eval`
- 4) Double-click to `deploy.bat` to deploy your application on the Tomcat. It should create the subdirectory “DialogMy” in the `webapps` directory of your Tomcat.
- 5) Make sure that Tomcat is up and running. Double-click to `DialogMy/run.html` and make sure that your DialogMy works as DialogCreditCard used to work.
- 6) Your project directory `DialogMy/war/rules/` contains two files:
  1. **Main.xls**: describes the structure of your project and its look-and-feel
  2. **Rules.xls**: describes the content and interaction logic of your questionnaire.
- 7) Start making changes in the file `Rules.xls`. You should put your own information into the following tables:
  - **pages** – a list of all your pages
  - **sections** – a list of all your sections
  - **questions** – a list of all your questions
  - **answers** – a list of defaults and possible answers. If there are no answers you still need to keep this table (even when it is empty). This table may use answer lists defined in the table similar to the one defined in the table “possibleAnswers” (this is not a key word and standard lists may be organized differently)
  - **autoResponses** (optional) – a list of all your automatically calculated responses
  - **navigateRules** – this decision table defines special conditions when the default sequential order of pages, sections, or questions is violated and

different order of pages is required. If you do not have special navigation rules, you may omit this rules table

- **updateRules** – this decision table defines special conditions for pages updates such as: a hide/show question, a hide/show section, a response to a question X with an answer Y, etc. If you do not have special update rules, you may omit this rules table.
  - **customQuestions** (optional) – this table defines application-specific layouts for custom questions.
- 8) You may customize the layout of your application by making changes in the file DialogMy/war/rules/Main.xls. You should put your own information into the following tables:
1. **Environment** – this table should always be in place. It describes the default structure of your rules project. You may modify this table only if you want to add your own Excel tables or Java classes – read more at [http://www.openrules.com/docs/man\\_api.html#Integration with Java and XML](http://www.openrules.com/docs/man_api.html#Integration with Java and XML)
  2. **mainLayout** – this table define the default main layout of your application with buttons “Next” and “Prev”. You may customize it to your own needs – see for example the sample projects “DialogCreditCard” and “Dialog1040EZ”.

You may rename the file Main.xls and Rules.xls but in that case please make a corresponding change to the file “*index.jsp*”. You may move some tables from this file to other Excel files – just make sure that you properly modify the Environment table.

## TECHNICAL INFORMATION

### Installation and Configuration

The current OpenRules Dialog (ORD) distribution is oriented to be used with Eclipse or a similar IDE, but it can also be used as a stand-alone system with just Excel and Windows Explorer. ORD is based on the OpenRules BRMS starting with the version 5.3.2.

To install ORD, first make sure that you have OpenRules 5.3.2 (or higher) installed and the following configuration projects are in place:

- openrules.config
- openrules.forms.lib
- pdfgen (optinal).

You also should install any Java-based web application server. We will assume that you have already installed a free Apache Tomcat server – see configuration details at [http://www.openrules.com/docs/man\\_config.html](http://www.openrules.com/docs/man_config.html).

The ORD can be downloaded from <http://www.openrules.com/ORD.htm>. After downloading the file “ORD.zip”, unzip it in the same folder where you already have the main OpenRules configuration project “openrules.config”. You will see a set of ORD projects:

- DialogType0: a simple dialog with Next/Prev navigation buttons
- DialogType1: a dialog of the type 0 with a special Login page and an ability to store/reload the dialog
- DialogType2: a dialog of the type 1 with a menu instead of Next/Prev buttons
- DialogType3: a dialog with long pages that always displays the next question to ask on the top of the page
- DialogType4: a dialog with one highly dynamic page
- DialogCreditCard: a simple credit card application
- Dialog1040EZ.: a real application that allows a user to fill in US tax form 1040EZ and generate a 1040EZ PDF document.

You may import these projects in your Eclipse workspace or in your stand-alone OpenRules installation directory: just make sure that all these projects are at the same level as “openrules.config”.

### Installation Steps:

**Step 1.** Deploy openrules.forms.lib at the Tomcat by double-clicking on “openrules.forms.lib/ deploy.bat”.

**Step 2.** Deploy other sample projects (at least DialogType0) at the Tomcat by double-clicking on file “deploy.bat”.

**Step 3** (optional). If you plan to generate PDF documents, deploy the project “pdfgen” at the Tomcat by double-clicking on “pdfgen/ deploy.bat”.

**Step 4.** To check that installation was successful, start your Tomcat and double-click on the file “run.htm” for a selected sample project. You should be able to run your ORD-based web questionnaire.

Any installation or configuration issues? Contact [support@openrules.com](mailto:support@openrules.com).

## Supporting Rule Templates and Forms

The basic ORD concepts such as Dialog, Page, Section, Question, and Answer are defined as Java classes inside the standard OpenRules package “*com.openrules.forms.gui.jsp*”. Thus, they are automatically included in the standard OpenRules release (since 5.3.2).

All standard forms (layouts) and rule templates are defined in the Excel files supplied with the standard OpenRules project “*openrules.forms.lib*”:

- Dialog.xls
- DialogMain.xls
- DialogForms.xls
- DialogRules.xls
- Validators.xls.

Appropriate style sheets are defined in the file *openrules.forms.lib/css/lib.css*, but can be overwritten by project specific style sheets such as DialogCreditCard/war/css/project.css.

## Customization and Integration – Developer Notes

You may essentially customize your application by changing only layout tables in the file Main.xls. You also may create your own version of the library “openrules.forms.lib” and completely change the look&feel of the generated questionnaire (web application).

You may integrate your questionnaire with any 3<sup>rd</sup> party Java package by adding the proper import-statements to the Environment table in Main.xls. If you want to use your own Java object inside your custom rules, you may add these objects to the instance of the class Dialog that is inherited from the standard Java HashMap. For example, in the file “index.jsp” you may create your own object “customer” and add it to the dialog as follows:

```
dialog.put("customer",customer);
```

Then, inside your Excel rules, you can always gain access to this object by using an instance “d” of the Dialog:

```
Customer customer = (Customer) d.get("customer");
```



This mechanism gives you complete control over the integration of the Excel-based questionnaire with your Java application.

If necessary, ORD allows you to bring the entire power of OpenRules Forms – see [http://openrules.com/docs/man\\_forms.html](http://openrules.com/docs/man_forms.html). The project DialogType3 provides a good example of how to do it.