



AGILEDIALOGS DESIGN GUIDE

Table of Contents

1.	Introduction	4
1.1	Disclaimer of warranty	4
2.	AgileDialogs Basics	5
2.1	AgileDialogs Design Concepts	5
2.1.1	Dialog Model	5
2.1.2	Question	6
2.1.3	Page	6
2.1.4	SubDialog	7
2.1.5	Validation	7
2.1.6	Context	8
3.	AgileDialogs Design	9
3.1	XRM Connection Window	10
3.1.1	Connection Tab	10
3.1.2	Entity Tab	10
3.1.3	AgileDialogs tab	11
3.1.4	Process Name tab	13
4.	AgileDialogs Shapes	14
4.1	Page Form Shape	15
4.1.1	Form Configuration Window	16
4.1.2	Question Configuration	18
4.1.3	Page Canvas	21
4.1.4	Question Configuration Advanced Tab	21
4.1.5	Values for Combo, Radio and Check controls	23
4.2	Question Types	25
4.2.1	Textbox	26
4.2.2	Combo	26
4.2.3	Radio	27
4.2.4	Check	27
4.2.5	Info	28
4.2.6	Yes/No	28
4.2.7	Calendar	28
4.2.8	Lookup	29
4.2.9	XRM Grid	30
4.2.10	File	31
4.2.11	Custom Control	33
4.2.12	Password	33
4.3	SubDialog shape	34
4.4	Dialog Task	35
4.5	Intranet Dialog Task	38
4.6	Internet Dialog Task	40
5.	Localization	42

6.	Advanced Features	45
6.1	JavaScript Extensions	45
6.1.1	Change Control Value	46
6.1.2	Add options to a selection control	47
6.1.3	Remove options from a selection control	47
6.1.4	Hide/Show controls	47
6.1.5	Add validation extensions	47
6.1.6	Subscribe to control value changes	48
6.1.7	Collapse or expand Comments control.	48
6.1.8	Get Dialog Context	48
6.2	Breadcrumbs	48
6.3	Custom Controls SDK	49
6.4	Dialogs Review	50
6.5	Preview Configuration	52
6.6	Exporting and Importing Form configuration	53
6.7	AgileDialogs cancelation	54
6.8	Setting a fixed width for the dialog at runtime	56
6.9	Launching AgileDialogs on Demand	57
6.10	Creating links to complete tasks associated to a Dialog	57
6.11	AgileDialogs Security	59

AgileDialogs Design Guide

1. Introduction

This document describes the usage of AgileDialogs from dialog designer perspective.

With AgileDialogs, Business Power Users can create dialogs or scripts to gather information from users and save this information either in Microsoft Dynamics CRM or other systems.

The terms 'AgileDialog' and 'dialog' are used interchangeably in the rest of the document. To avoid confusion, *CRM's own dialogs*, are referred to as such, in *italic*.

1.1 Disclaimer of warranty

AgilePoint Inc. makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Copyright © 2012, AgilePoint Inc. All rights reserved.

GOVERNMENT RIGHTS LEGEND: Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the applicable AgilePoint Inc. license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

'AgilePoint Inc.' and all its products are trademarks of AgilePoint Inc.. References to other companies and their products use trademarks owned by the respective companies and are for reference purposes only.

2. AgileDialogs Basics

An AgileDialog is a visually designed model comprising of a set of UI steps (Page Forms) and optionally, automatic system steps. The interface presents data and questions to an end-user to inform or get input from the end-user. The flow is modeled using MS Visio.

AgileDialogs can be used for many purposes, for example:

- Self-help guides
- Troubleshooting wizards
- Call Scripts in Call Centers
- Help users fill out a complex form
- Create Surveys
- Create Tests & Exams

AgileDialogs can be executed standalone, meaning the dialog is not part of an AgileXRM process, or they could be part of a business process, that is, the execution of the dialog is a task assigned to an end-user. From the end-user's perspective, a dialog is a set of questions that he/she must answer in order to perform a task. Within the dialog, the user can go forward or backward in order to fulfill the required task. The dialog is presented to the end-user using a Silverlight-based Dialog Render Engine.

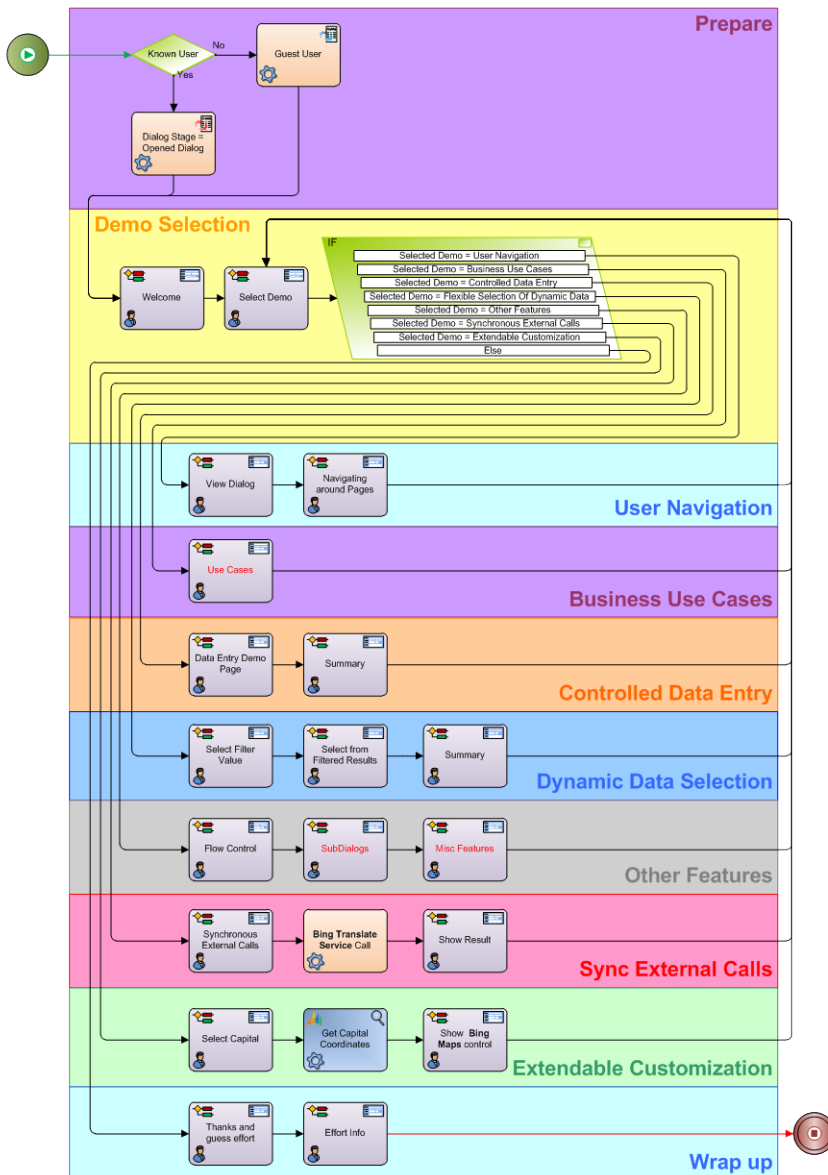
2.1 AgileDialogs Design Concepts

These are the basic concepts that an AgileDialog designer must know.

2.1.1 Dialog Model

An AgileDialog Model is a business process drawn and configured in Envision (MS Visio), used for "holding a conversation" with an end-user. This process is different from other AgileXRM processes, as all the tasks are executed **synchronously** by the end-user; there are no task assignments or tasks lists involved.

AgileDialogs can contain manual tasks (form pages), decision points (based on context data) and automatic activities (to interact with XRM repository or other systems).



2.1.2 Question

This is the basic unit of a dialog. Each question corresponds to a control (e.g. Textbox, Listbox, Radio Button, etc.) that is presented at runtime to get information from the user. Each question has a caption, where the designer can specify the content of the question. Beside this caption, the designer can optionally add a tooltip and further help. The content of the controls can be static or dynamic. For example, there is a control that presents a grid using a query to retrieve data from the XRM repository.

2.1.3 Page

Dialog questions are grouped in pages (a page is a form with a set of controls). Each Page is a step in an AgileDialog model as defined in Envision. Between pages, the modeler can include decisions and automatic activities to control the flow and interact with the XRM repository or other systems.

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting

■ Address Type ■ CL
 Billing Avenue

■ Street name ■ Address 2 ■ n

■ Zip Code ■ City
 Madrid

■ Picklist
 crm

■ password ■ confirm

2.1.4 SubDialog

This is for calling an AgileDialog from another AgileDialog. It helps organize complex dialogs into manageable pieces, as well as for creating reusable SubDialogs that can be reutilized in different dialogs.

2.1.5 Validation

AgileDialogs provides validation features.

All controls have a *Required* property, when this is TRUE, the user must fill the control to continue with the dialog.

[This page shows some mechanisms available in **AgileDialogs** pages to ensure we get the right data from the user. Below are various types of data entry controls: *Text*, *Password*, *DateTime* and *File Upload*. Some have *validation* associated to them. Hover on the exclamation triangle to see the validation error message. Note that everything can have a dynamic value, including the validation rule or even the validation error message. Controls can have a *Tip*, shown by hovering on the question mark icon, with the possibility of adding a further *Help page*, shown by clicking the **Read more...** link in the Tip. Also conditional behavior of a control based on another control's value are demonstrated]

Standard Textbox Textbox with **Dynamic Default Value**
 `${di_firstname} ${di_lastname}`

Textbox with **Dynamic Validation Rule** ■ Textbox with **Required** flag set ⚠ This Field is Required

Password Control DateTime Control
 <M/d/yyyy> 15

Option Control (select an option) Control that is shown conditionally
☐ United States
☐ Europe
☒ Other

In the case of Text controls, there is specific validations to check max length for strings, validate using regular expressions or type (validate that a string is an integer, decimal,...). Additionally, the designer can include JavaScript code to perform more complex validations.

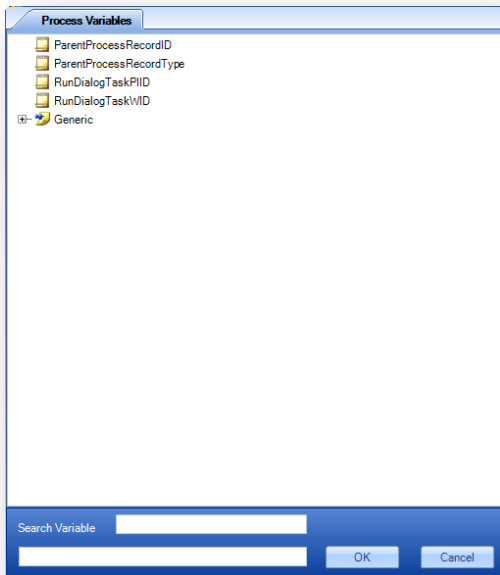
2.1.6 Context

The dialog has access to a context. These are dynamic values which are used in the dialog.

The context available depends on these criteria:

	Context available in dialog
No Main Entity set	1. Preset Process Variables 2. Responses from Questions
Main Entity Set	1. Preset Process Variables 2. Responses from Questions 3. Main Entity Schema

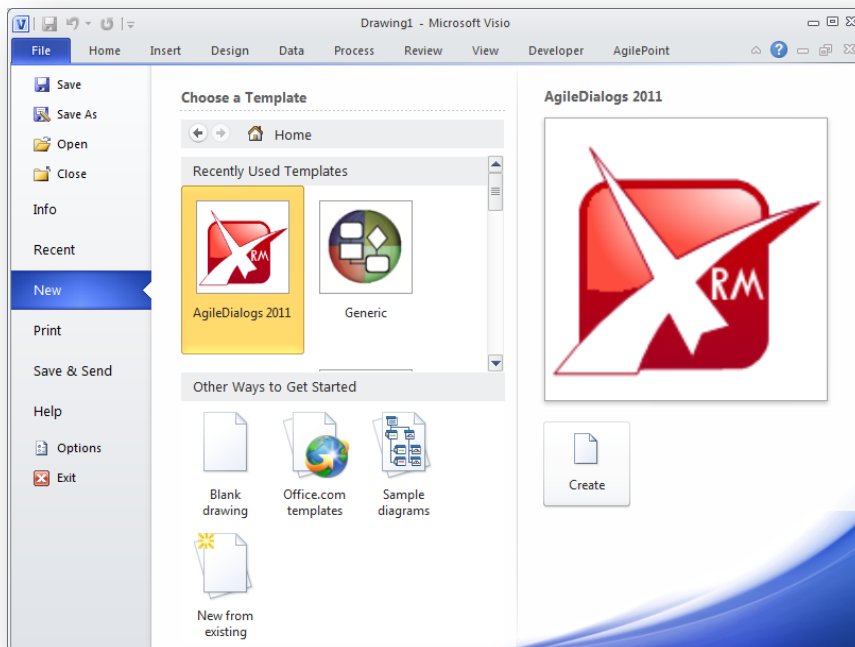
The following 4 Preset Process Variables are only usable if the AgileDialog is executed as a Task in an AgileXRM Process, and help to interact with the parent process and the entity record it is managing:



- *ParentProcessRecordID*: ID of the main entity in the parent process that contains the task. This ID can be used to work with the record in the dialog (e.g. Update its status)
- *ParentProcessRecordType*: Type of the main entity in the parent process (e.g. *account*)
- *RunDialogTaskPIID*: Process instance ID of the process that contains the task that corresponds to the execution of the Dialog. This can be used, for example, to update a variable in the context of the parent process from within the dialog.
- *RunDialogTaskWID*: ID of the *Workitem* that corresponds to the execution of the Dialog. This is for information purpose only and has no other use.

3. AgileDialogs Design

There is a specific Visio template to create AgileDialogs:

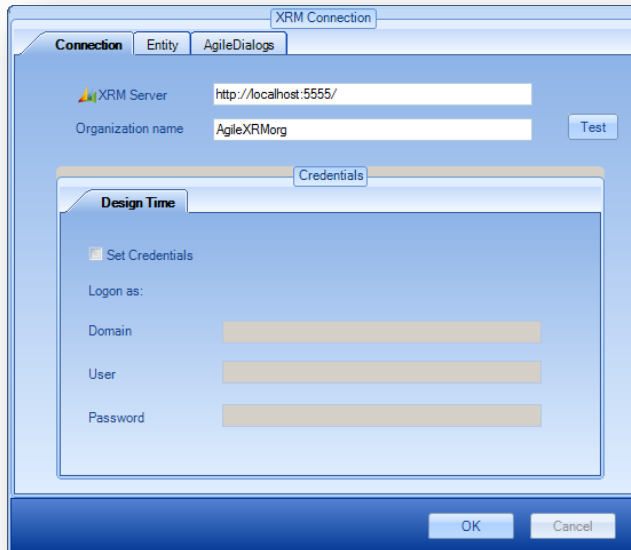


When this template is selected a new AgileDialog is created. The first thing to do is to configure the interaction between the dialog and XRM in *XRM Connection* window.

3.1 XRM Connection Window

This window can be reopened at any time from the AgileXRM Ribbon.

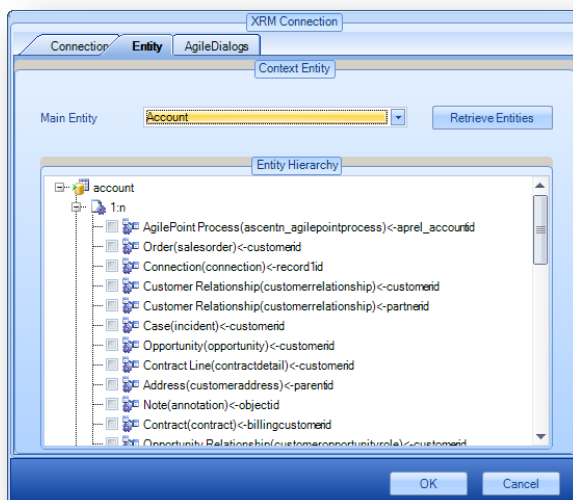
3.1.1 Connection Tab



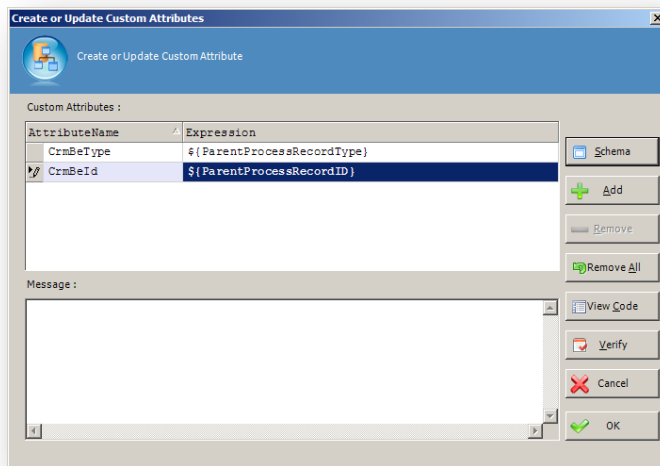
In this tab, design-time connection configuration is set. These values do not affect runtime, only design-time. So when a dialog is moved from one environment to another, there are no hard-coded values in it.

3.1.2 Entity Tab

Setting Main Entity is **optional** in AgileDialogs. There is no need to set a relationship between the dialog and a record in XRM.



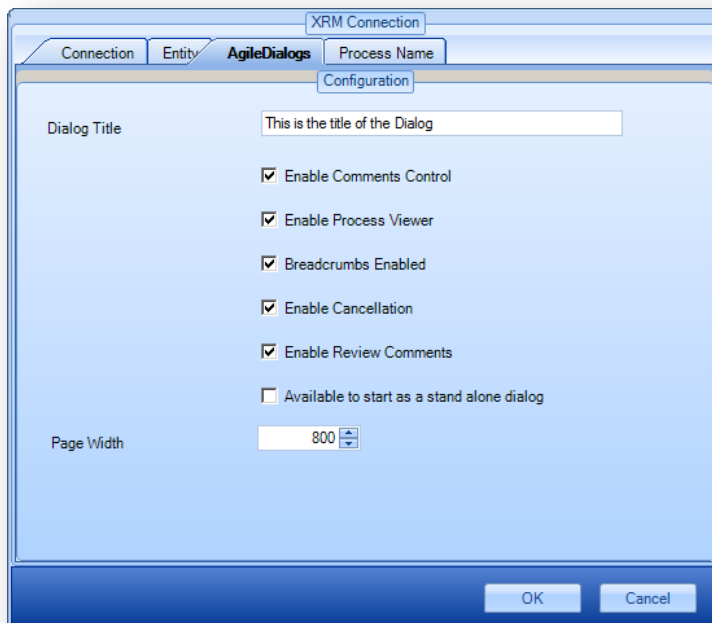
Tip: If you wish to set the relationship between the dialog and the record in XRM (this is useful for using context data filters in AgileMonitor), then set *CrmBeType* and *CrmBeId* parameters using an *Update Custom Attributes* shape at the beginning of the dialog:



This also means that the dialog will be shown in the *View Process* tab for the related record in XRM.

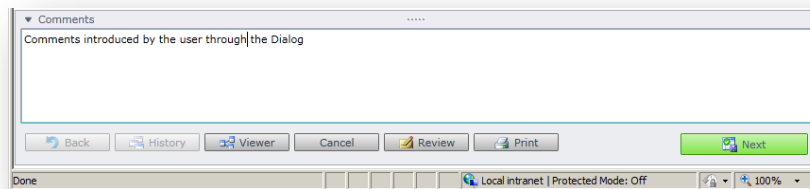
3.1.3 AgileDialogs tab

In this tab, the title to show to the user is set. This title is presented at runtime at the top of the AgileDialogs runtime window.



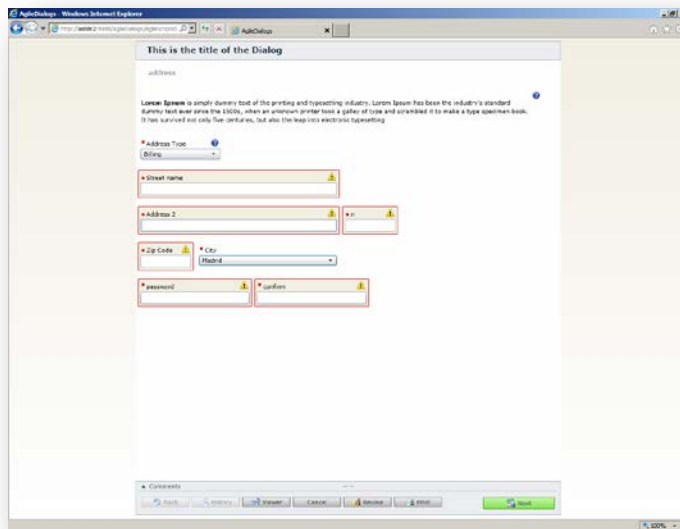
Also there are several parameters that can be configured here:

- *Enable Comments Control*: when checked, a textbox is shown throughout the dialog to get comments from the user. These comments are persisted in a dialog context variable called **ProcessNotes**. For example, the value of the comments can be added as a *Note* to a *Case* using an *Add Note* shape
- *Enable Process Viewer*: when checked the user will see a Viewer button for accessing the AgileXRM Process Viewer. This is useful for dialog testing and demo purposes. **NOTE**: It is not for production environments and should never be set
- *Breadcrumbs Enabled*: Breadcrumbs are used to break a sequential process into stages and informing the user in which stage they are (e.g. Login → Order → Payment → Confirmation). This should be only selected if all steps in the AgileDialog are Sub-Dialog shapes. If checked, a *breadcrumb* is shown at the top of the dialog at runtime. As execution moves from Sub-Dialog to Sub-Dialog, the title of the active Sub-Dialog is highlighted in the Breadcrumbs control at runtime
- *Enable Cancellation*: When checked the user can cancel the dialog in the middle of the execution
- *Enable review comments*: while creating the dialog, this option can be used to allow testers to send feedback to the dialog designer. When this option is checked, a Review button is shown at runtime and the tester can add comments at any step of the dialog



- *Available to start as standalone dialog*: When this check is selected the dialog can be started from AgileXRM SharePoint WebPart if the user has permission.

- **Width:** This is the width (in pixels) of the pages of the dialog at runtime. If the value is 0 the page width will be all space available. For instance, this is a dialog with width set to 800:

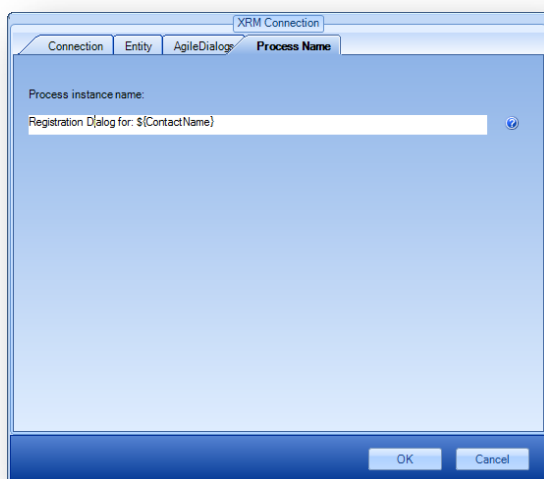


3.1.4 Process Name tab

In this tab the designer can configure the name of the process instance that will be associated to the dialog. This name is presented in Process Manager and is useful to know the context of the dialog while monitoring it. For instance, if the dialog is associated to a contact, it is useful to set the name of the contact as instance name to know who the dialog is associated to.

The value of the name can be parameterized with context data. The name is set when the dialog is started so the parameters used in the name must have value when the dialog is started.

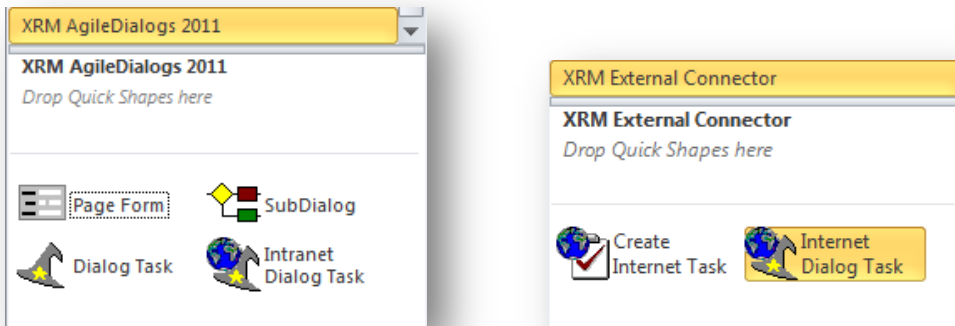
This name is not the same as the dialog title; the dialog title will be presented to the user that runs the dialog, whereas process name is used internally by process managers.



This tab is available when opening this window from AgileXRM ribbon (using AgileXRM Settings button).

4. AgileDialogs Shapes

In Envision, these stencils contain the shapes specific to AgileDialogs:



There are 5 shapes in total, four in the *XRM AgileDialogs 2011* stencil and one in the *XRM External Connector* stencil:

1. *Page Form*: a step in an AgileDialog where the form is configured. The form can contain multiple controls of various types
2. *SubDialog*: Allows calling another AgileDialog where this step is used. There are two important differences with *CRM's own sub-dialog* functionality:
 1. A SubDialog step can be used anywhere in an AgileDialog, and not just as the last step
 2. The *Back* button works correctly no matter how complex the hierarchy. In *CRM dialogs*, it is not possible to go back to the parent dialog; breaking the end-user experience.
3. *Dialog Task*: used in an AgileXRM Process as a Task for a Power User. Upon opening the task in CRM, the user is presented with the associated AgileDialog. The Task is *Completed* when the user completes the dialog.
4. *Intranet Dialog Task*: used in an AgileXRM Process as a Task for a Standard User. Upon opening the task in the Portal, the user is presented with the associated AgileDialog. The Task is *Completed* when the user completes the dialog. User authentication in the Portal is via Active Directory.
5. *Internet Dialog Task*: used in an AgileXRM Process as a Task for an External User. Upon opening the task in the Portal, the user is presented with the associated AgileDialog. The Task is *Completed* when the user completes the dialog. User authentication in the Portal is via a custom *MembershipProvider* using CRM Contacts.

4.1 Page Form Shape



This shape is only usable in AgileDialogs. It cannot be used in AgileXRM processes. When a dialog is run, this shape presents a form to the user.

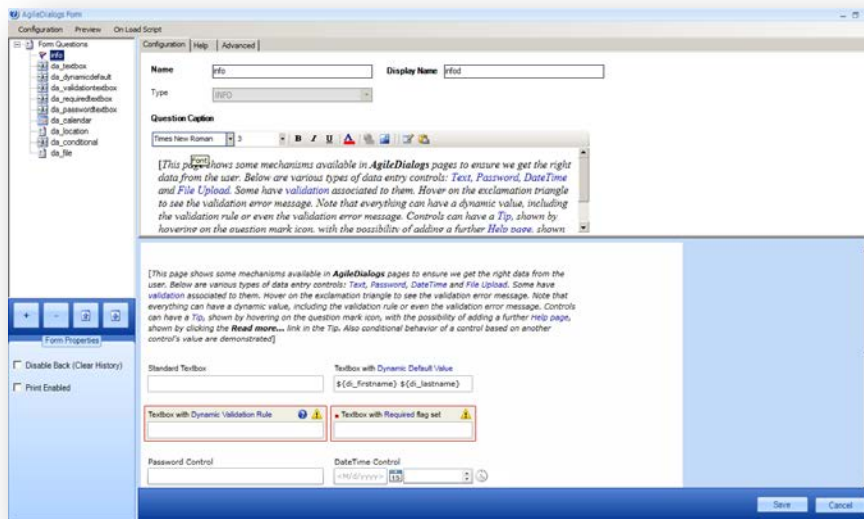
A screenshot of a web browser window titled "AgileDialogs - Windows Internet Explorer". The address bar shows a local URL. The page content includes a title bar "This is the title of the Dialog Changed", a text input field labeled "address", a paragraph of Lorem Ipsum text, and several form fields: "Address Type" (Billing), "CL" (Calle), "Street name", "Address 2", "Zip Code", "City" (Madrid), "Picklist" (London), "password", and "confirm". At the bottom, there are buttons for "Back", "History", "View Dialog", "Cancel", and "Next". The status bar at the bottom indicates "Local intranet | Protected Mode: Off" and "100%".

At runtime, the AgileDialogs window has several parts:

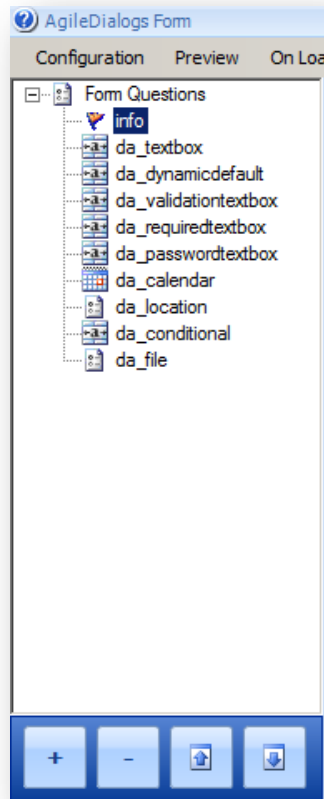
- The top part shows the title of the dialog. This title is shared by all steps in the same dialog
- Under the title of the dialog, the title of the page is presented
- In the main area, the questions/controls are presented. This is the section that changes to show the corresponding questions for each Page Form
- In the bottom of the window there is a button set that allows the user to interact with the dialog

4.1.1 Form Configuration Window

This window is used to configure a form in AgileDialogs, and is comprised of 4 sections:

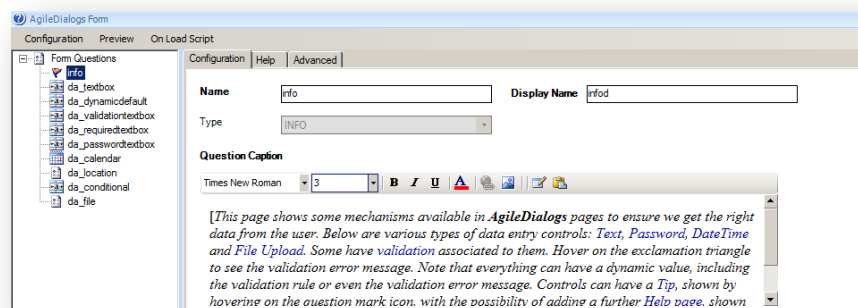


1. **Question List:** In the top left part of the window it lists the questions/controls that are already added to the form. These are identified by the name of the variable which will store the value provided by the user:



The buttons on the bottom are used to *Add* and *Remove* questions and to move them *Up* and *Down* in the form.

2. **Question Configuration:** To configure a specific question, select it in the list, and the properties of the selected question are presented on the top right:



3. **Page Preview:** The bottom right side of the window shows a preview of what would appear at runtime:

[This page shows some mechanisms available in **AgileDialogs** pages to ensure we get the right data from the user. Below are various types of data entry controls: *Text*, *Password*, *DateTime* and *File Upload*. Some have *validation* associated to them. Hover on the exclamation triangle to see the validation error message. Note that everything can have a *dynamic value*, including the validation rule or even the validation error message. Controls can have a *Tip*, shown by hovering on the question mark icon, with the possibility of adding a further *Help page*, shown by clicking the **Read more...** link in the Tip. Also conditional behavior of a control based on another control's value are demonstrated]

Standard Textbox

Textbox with Dynamic Default Value
\${di_firstname} \${di_lastname}

Textbox with Dynamic Validation Rule

Textbox with Required flag set

Password Control

DateTime Control
<M/d/yyyy> 15

Option Control (select an option)
☐ United States
☐ Europe
☒ Other

Control that is shown conditionally

File Upload Control
Browse...

4. Form Properties

Disable Back (Clear History)	If checked, when the dialog arrives to this page, the dialog history is cleared and the <i>Back</i> button becomes disabled. This is useful when the dialog designer decides to block the user from going back to previous steps to this one. Used when previous steps include automatic system steps that should not be repeated
Print enabled	When checked the <i>Print</i> button is presented so that the user can print the content of the page. Especially useful for Confirmation or Summary type pages

4.1.2 Question Configuration

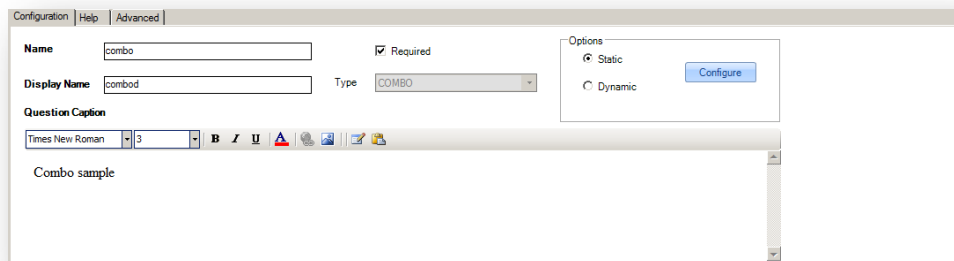
These are the properties used in the configuration of a question:

- **Name (ValueVariable):** this is the name of the variable to store the value that the user enters/selects at runtime. This name **must be unique** at page level. It does not need to be unique in the whole dialog, but its value is to be referenced and used in another place, using this format **\${ValueVariable}**, then the dialog designer should make sure a duplicate variable in another question is not interfering inadvertently.

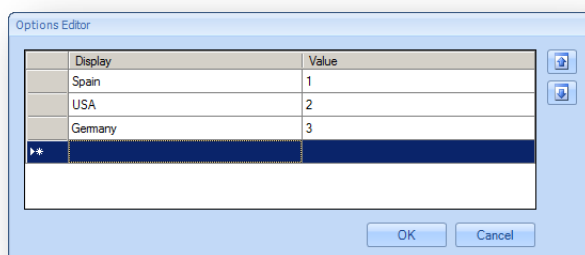
- **Display Name** (*DisplayVariable*): when the control is a checkbox or a radio, there are 2 values to take into account when the user selects an option, one is the internal value (stored in the *ValueVariable*) and the other is the display name. For instance in a list of customers in a combo, when a user selects one, the *ValueVariable* is set to the ID of the customer and *DisplayVariable* is set to the name of the customer (as seen on the form). This value can be used in other steps using this format ***\${DisplayVariable}***
- **Type**: This is the type of the control. This field is read-only and is set when a new Question is added to the Page.
- **Question caption**: This text area contains the text presented to the user. This text can be formatted (color, size, italics, bold, etc.). This field can contain images. Due to security restrictions in Silverlight these images must be in the same web server that AgileDialogs runs, if these images are in other server, *History* feature will not work.

The content of the question can include dynamic values using ***\${variablename}*** syntax. The context button  can be used to open the context window and include dialog variables in the caption. To paste content from other applications use the Paste button  provided. Not all HTML content is supported by this configuration tool and content copied from applications like Word, for instance, can cause unexpected behaviors.

- **Options**: When the selected control shows a set of options to the user (combo control, radio, ...), these options can be configured as static (set at design time) or dynamic (can be linked to a filtered lookup in the XRM repository). For this controls the *Options* region is enabled:

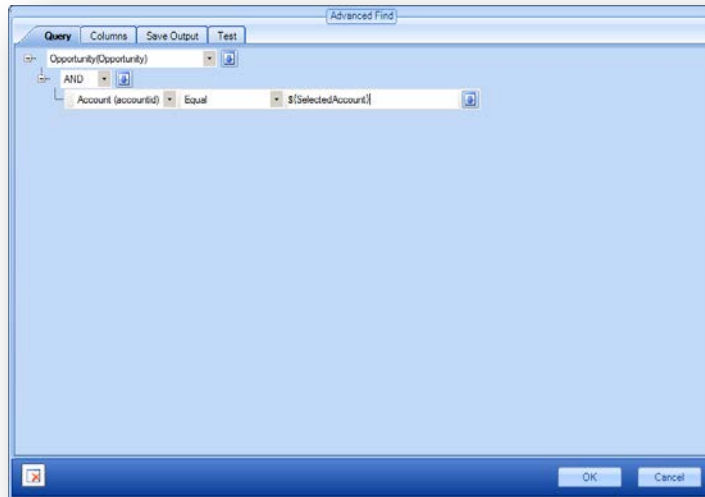


This window is used to configure static options:



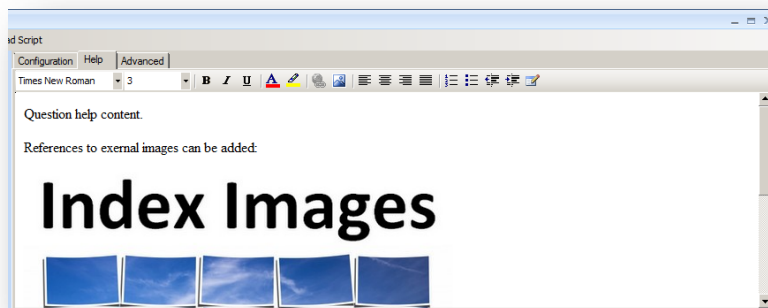
Display	Value
Spain	1
USA	2
Germany	3

This window is used to configure a query to get the values to be shown in the control:



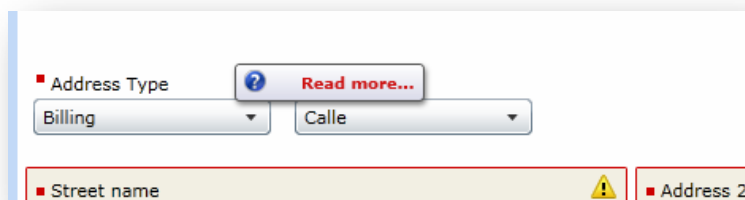
These queries can have dynamic filters (e.g. show opportunities whose Account is the value of `$(SelectedAccount)`).

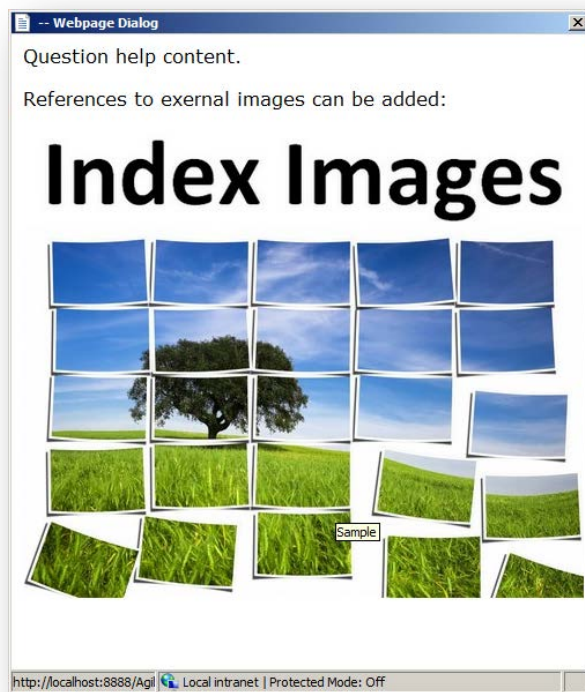
- Help tab is used to configure HTML content to help the user with this question. This help is presented in a modal window:



This HTML content can have references to external images. The images should be hosted in a web server and the URL should be accessible at runtime by the user.

At runtime, the help is accessible in the tool tip for the control via the link *Read more...*:





4.1.3 Page Canvas

To permit an optimum use of the screen real-estate, AgileDialogs allow presenting the controls, not only one below the other, but also in a grid format.

When the form is presented, the runtime engine measures the available horizontal space and based on this, calculates the number of columns available (each column has a default size of 55 pixels).

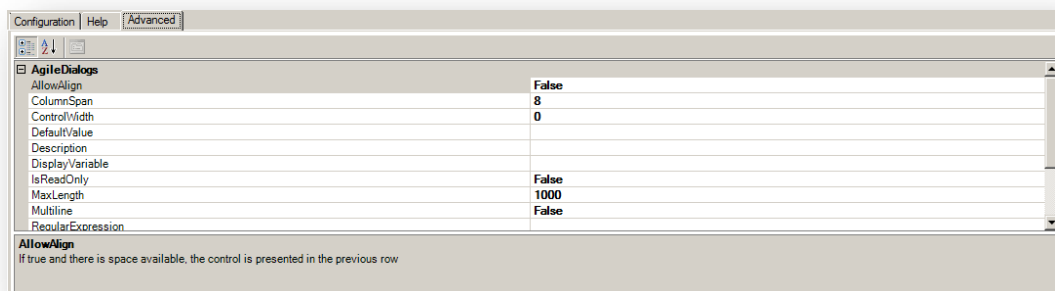
Questions can be expanded to several columns using *ColumnSpan* property and can be put concatenated to the previous row using *AllowAlign* property. These setting are in the Advanced Tab.

4.1.4 Question Configuration Advanced Tab

Each type of control has a set of specific properties to configure its behavior.

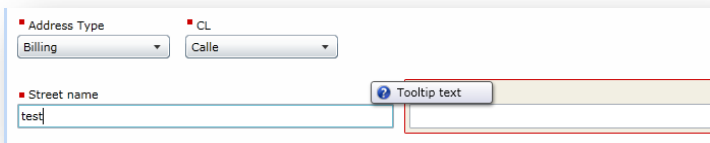
These specific properties are configured in the *Advanced* tab.

Besides these specific properties, there are some properties common to all controls.



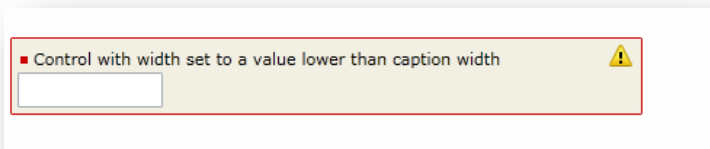
Common properties are:

- **Required:** When *true*, the user must select or introduce a value in order to continue
- **AllowAlign:** When this value is true, the control will be concatenated to the previous row (if space is available). This is used for presenting several controls in the same row
- **ColumnSpan:** This is the number of columns that the caption will use. The total number of columns will depend on the space available in the client window. Controls will be rendered using the value of this property in combination with *AllowAlign* property
- **Description:** This is the text of the tooltip to show with the control



A screenshot of a form interface. It contains two dropdown menus at the top: 'Address Type' with 'Billing' selected, and 'CL' with 'Calle' selected. Below these is a text input field labeled 'Street name' containing the text 'test'. A tooltip is visible over the text input field, displaying the text 'Tooltip text'.

- **ControlWidth:** this is the width of the control in pixels (for instance the textbox) in the question. By default the width of the control is the same that the width of the caption



A screenshot of a form interface showing a warning message. The message text is 'Control with width set to a value lower than caption width' and is accompanied by a yellow warning triangle icon. Below the message is a text input field.

- **DefaultValue:** This value is set when the question is loaded at runtime. This is the default value of the control. In selection controls, this value must be the value of the option, not the display name.
- **IsReadOnly:** When *true* the control does not allow the user to change its value.

Question-specific parameters are explained for each Question-type below.

4.1.5 Values for Combo, Radio and Check controls

Static Values

To configure static values, select *Static* in the options region and click *Configure*:

Configuration | Help | Advanced

Name: ☒ Required

Display Name: Type: COMBO

Question Caption: Times New Roman 3 **B I U**

Combo sample

Options: ☒ Static ☐ Dynamic [Configure](#)

The *Options Editor* will be open:

Display	Value
One	1
Two	2
Three	3
*	

OK Cancel

When the options are presented to the user and he/she selects it, the Display content is stored in the dialog context in *DisplayVariable* and the Value content is stored in *ValueVariable*.

The order of the options can be changed using the Up & Down icons buttons.

Dynamic values using a XRM query

To configure a dynamic lookup to retrieve the options, select *Dynamic* in the options region and click *Configure*:

Configuration | Help | Advanced

Name: ☒ Required

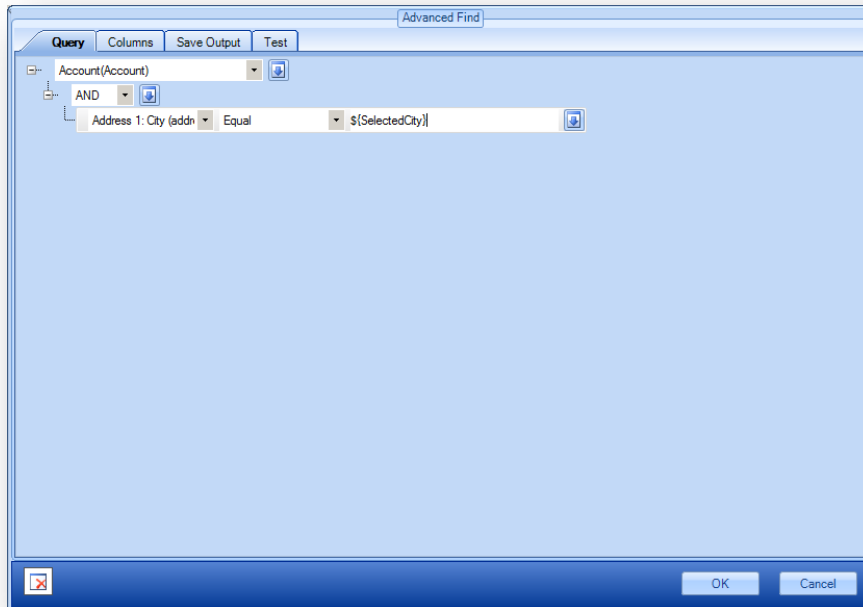
Display Name: Type: COMBO

Question Caption: Times New Roman 3 **B I U**

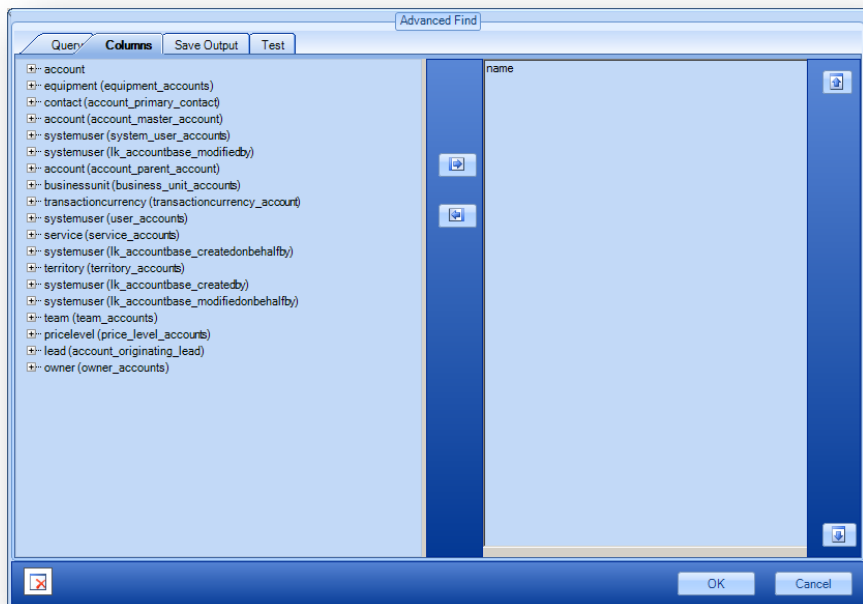
Combo sample

Options: ☐ Static ☒ Dynamic [Configure](#)

This window is presented to configure the query:



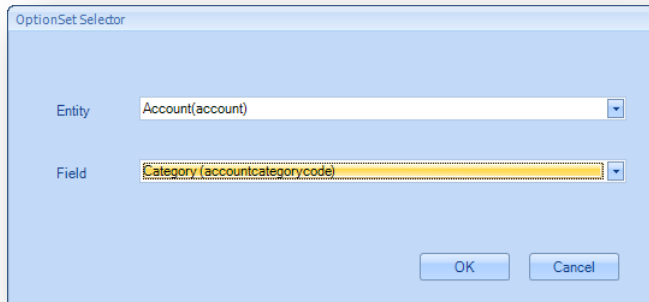
The filters can be dynamically built using dynamic values (using `${ContextVariableName}`). In the *Columns* tab select a single field as the Display value for the control. The Value part of the control will be the record ID:



NOTE: When values are dynamic, the control is not shown in the Preview area.

Dynamic values from an Option Set in XRM

Combo and radio controls allow to be filled with values from *Option Sets* configured in XRM. To configure this behavior go to *Advanced* tab and configure the *FieldAndEntity* property to open the *OptionSet Selector*:



4.2 Question Types

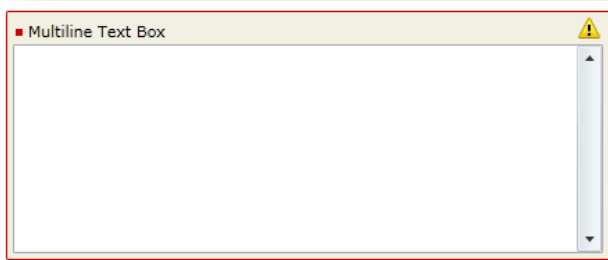
Currently the following question types are available:

- Textbox
- Combo
- Radio
- Check
- Info
- Yes/No
- Calendar
- Lookup
- XRM Grid
- File
- Custom Control
- Password

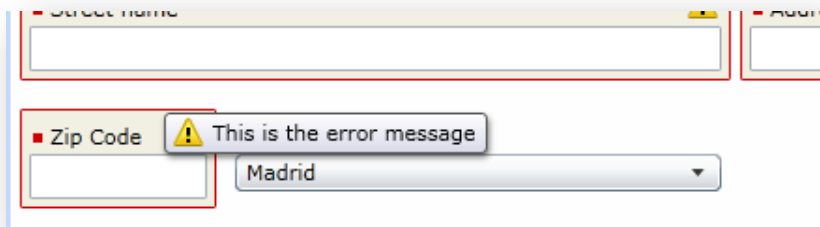
Each type has specific configuration besides the common properties.

4.2.1 Textbox

- *MaxLength*: This is the maximum length of characters allowed in the textbox
- *Multiline*: When *true*, the control allows multiple lines of text:



- *RegularExpression*: This field allows including a regular expression to validate the content of the textbox (the content of the validation message can be set in *ValidationMessage* parameter)
- *TextControlContentType*: The values allowed are: String, Number, Decimal and DateTime. The format of the text is validated to check that it corresponds to the type specified in this field (the content of the validation message can be set in *ValidationMessage* parameter)
- *ValidationMessage*: The message that will be presented when the content not match the validation rule:



4.2.2 Combo

This control presents a set of options in a combo box (Currently the edit part of the combo is disabled and the combo control behaves as a standard drop-down list box).

Combo can show:

- Static values
- Dynamic values

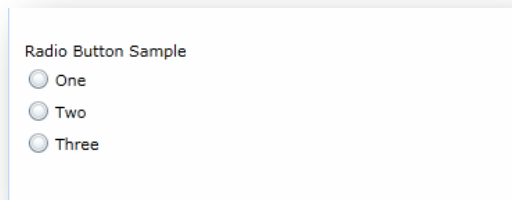
See Values for Combo, Radio and Check controls for details.

4.2.3 Radio

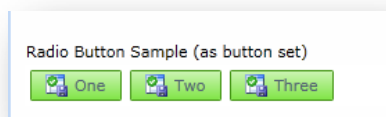
This control is used to show a set of options to the user and allows selecting only one of them.

These values can be:

- Static, set at design time
- Dynamic



This control can also be used show a set of Action buttons. By selecting the option, the user has in effect selected the option and clicked the *Next* button. In order to activate this behavior, in *Advanced* tab set the *ShowAsButtonSet* property to *true*. This action also hides the standard *Next* button:

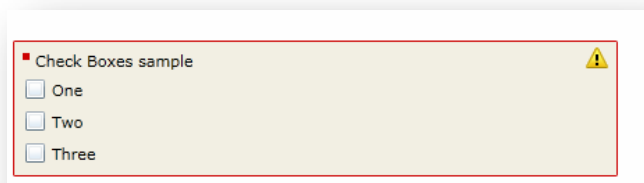


4.2.4 Check

This control is used to show a set of checkboxes to the user and allows multiple selections.

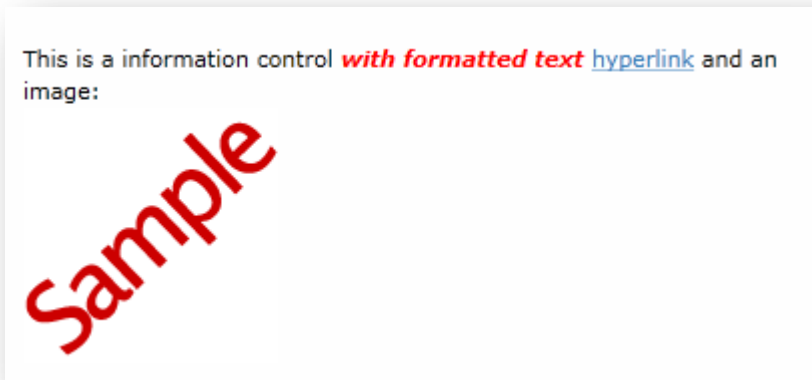
The values shown to the user can be:

- Static, set at design time
- Dynamic



4.2.5 Info

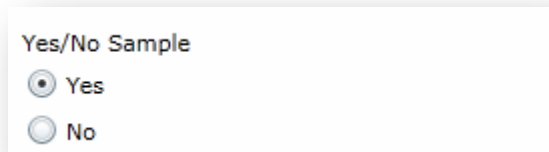
This control is used to present information to the user. There is no input associated with this control. Dynamic values can be included using `${variable}` syntax.



4.2.6 Yes/No

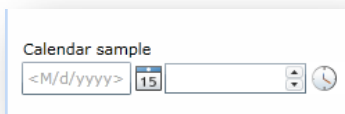
This control is the same as a *Radio* control with 2 preset options:

- Yes (true)
- No (false)



4.2.7 Calendar

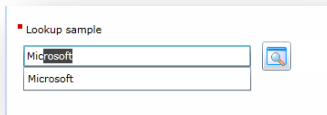
This control shows a date picker and an optional time picker to the user:



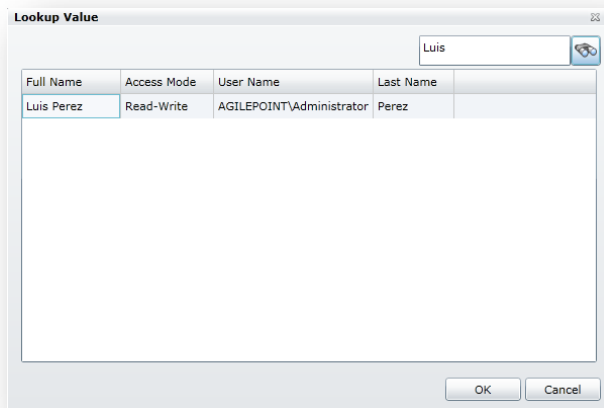
In order to only show the date picker, in the *Advanced* tab set *DateOnly* property to *true*.

4.2.8 Lookup

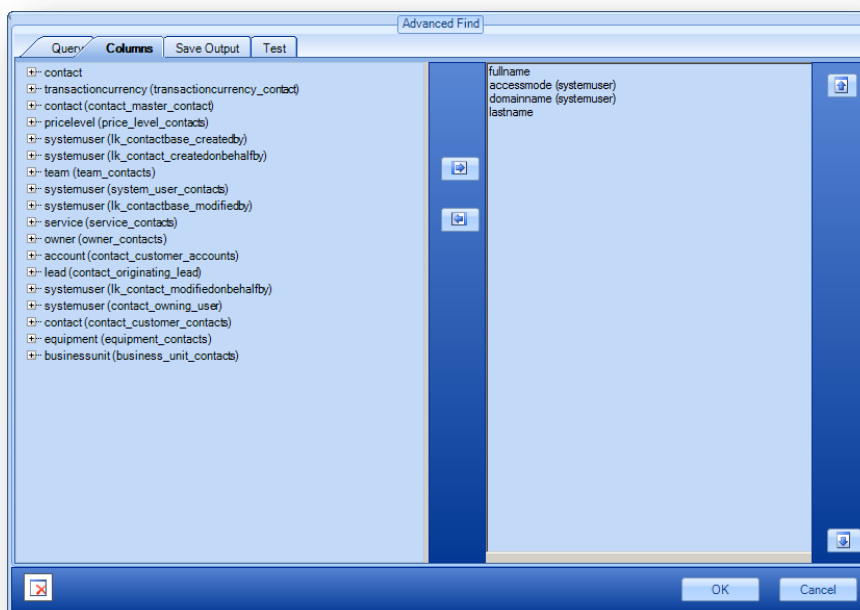
This control presents a control that allows looking up for records in XRM repository and provides auto-complete features (after typing the 3rd character) to help user select a value.



When the user clicks the Search icon on the right, the lookup window is opened:



The query used to filter the records presented to the user, is configured by clicking the *Configure Lookup* button. The fields presented in this lookup window are configured in *Columns* tab of the *Advanced Find* window.



The first column is used for the auto complete textbox as display value.

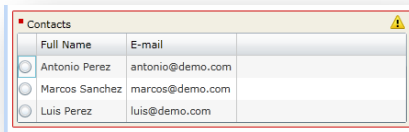
The records returned by this query can be paginated using the property *PageSize* in *Advanced* tab. By default the value is 0 (no pagination is used).

4.2.9 XRM Grid

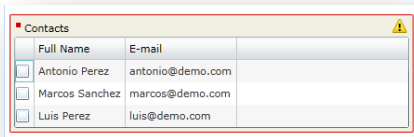
This control presents a grid with the records selected with a query configured using *Configure Lookup* button.

This control can be configured to behave in 3 different ways:

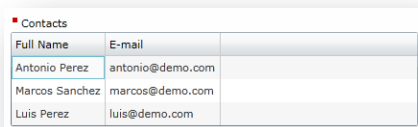
- **Allow single selection:** in *Advance* tab set *AllowMultiple* property to *false*.



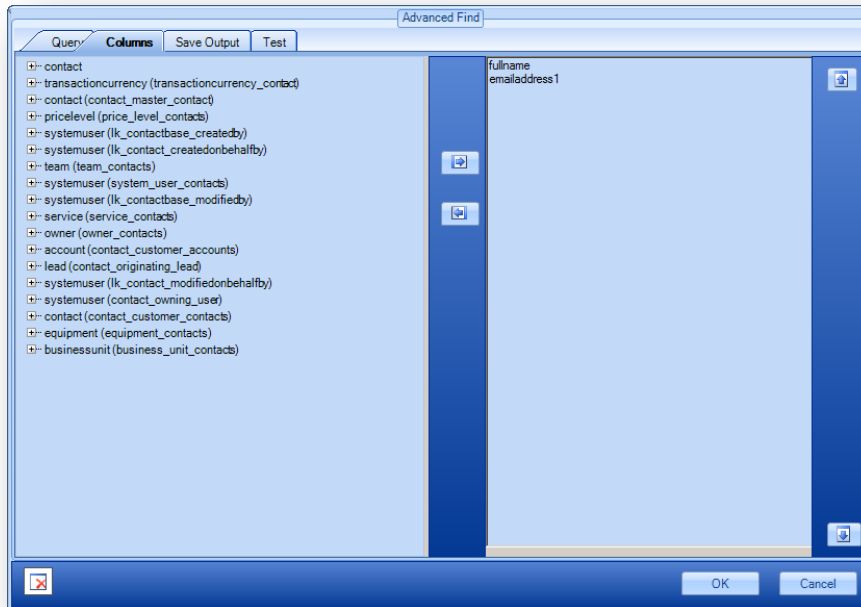
- **Allow multiple selections:** in *Advance* tab set *AllowMultiple* property to *true*.



- **Readonly:** In *Advance* tab set *ReadOnlyGrid* property to *true*. This behavior is useful for summary pages.



The fields presented in the grid are configured using *Configure Lookup* button, and configuration in the *Columns* tab:



When multiple selection is allowed, the values selected by the user (the GUIDs of the records) are stored in the Value variable separated by comma. This format can be used in other AgileXRM shapes like *Update Entity (Multi)*.

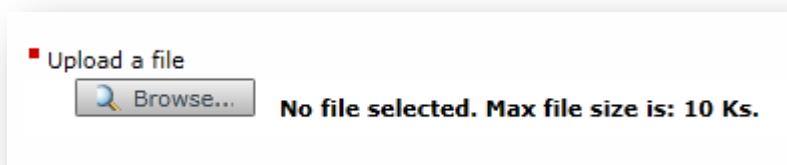
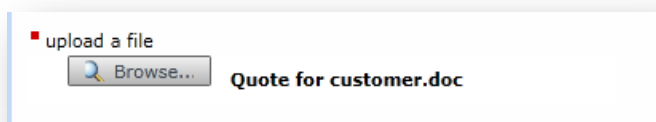
In this control *DisplayVariable* has the same value as the *ValueVariable*.

The records returned by this query can be paginated using the property *PageSize* in *Advanced* tab. By default the value is 0 (no pagination is used).

4.2.10 File

This control is used to allow the user to upload files.

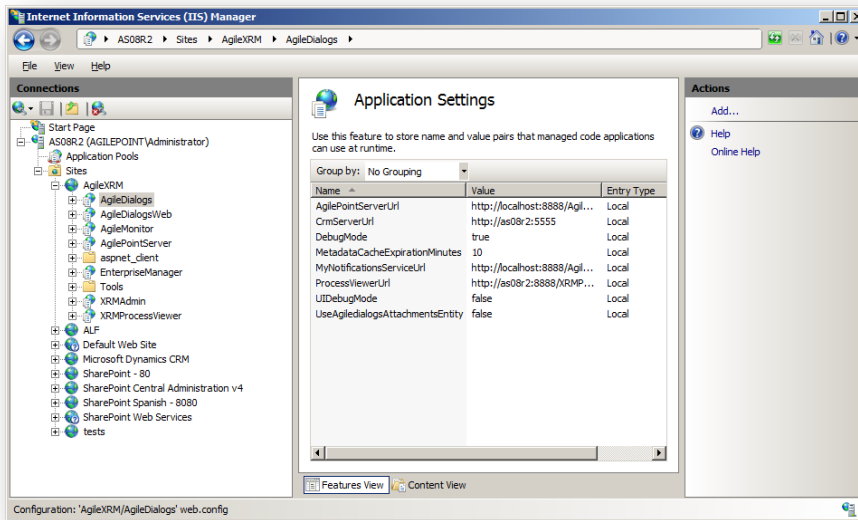
The maximum size allowed can be configured in the parameter *MaxFileSize*.



There are 2 options to manage uploaded files:

- Upload them to server file system (recommended for testing purposes only).
- Upload them as an attachment to a custom entity in XRM called *AgileDialogs Temp Attachments* (this entity is created by AgileXRM setup). This is the approach recommended for production environments.

This behavior is configured at server level in AgileDialogs application:



In order to use the XRM entity, the *UseAgileDialogsAttachmentsEntity* setting must be set to *true*. When a file is uploaded a unique name is assigned to it and stored in *ValueVariable* value (*DisplayVariable* contains the actual name of the file).

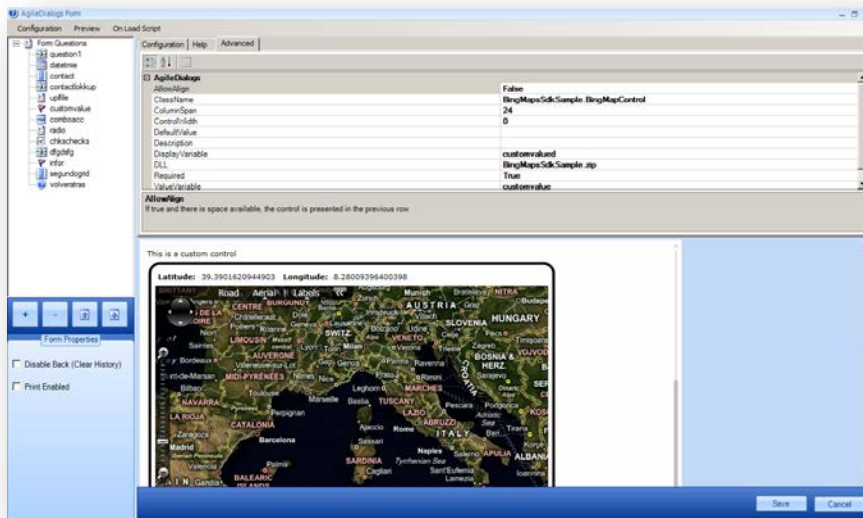
This name can be used either to find the file in the file system or to find the record in the entity *AgileDialogs Temp Attachments* that contains the attachment. The Name field (*ascentn_name*) contains this unique name.

4.2.11 Custom Control

This control is used to present a custom control developed in Silverlight using AgileDialogs SDK. See **AgileXRM Developer Guide** for more information.

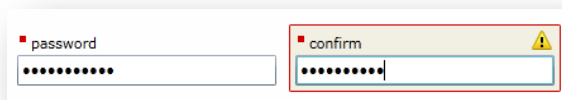
To configure the control, set *DLL* property in *Advanced* tab to the corresponding *DLL* or ZIP file (containing the DLL and DLLs it references), and set *ClassName* property to the name of the class of the custom control.

This is a sample custom control using Bing Maps:



4.2.12 Password

This control is used to present a control to get passwords. The password can be encrypted to be stored securely. To enable encryption, set *Encrypt* parameter to *true* in *Advanced* tab.



4.3 SubDialog shape



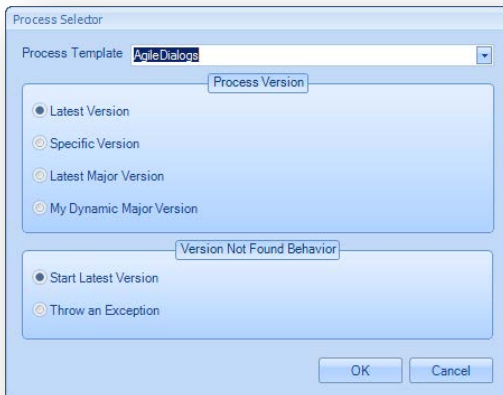
This shape is used to launch a SubDialog.

Configuration

A SubDialog is similar to a sub-process in AgileXRM. SubDialogs always share the context with the parent dialog. They can be included anywhere in the parent dialog and multiple levels are allowed.

Dialog Property

This window is used to configure the SubDialog to run:



Process Template	Select which dialog to run, from the list of AgileDialogs already deployed to the Process Engine
Latest Version	The latest version of the SubDialog will be used at runtime
Specific Version (*)	The selected version will always be used
Latest Major Version (*)	The highest minor version of the selected major version is used
My Dynamic Major Version (*)	The same as Latest Major Version but the major version number is the calling dialog's Major Version
Start Latest Version (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will behave as if Latest Version was selected
Throw an exception (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will make the shape throw an Exception (i.e. process hangs in this shape until the problem is remedied)

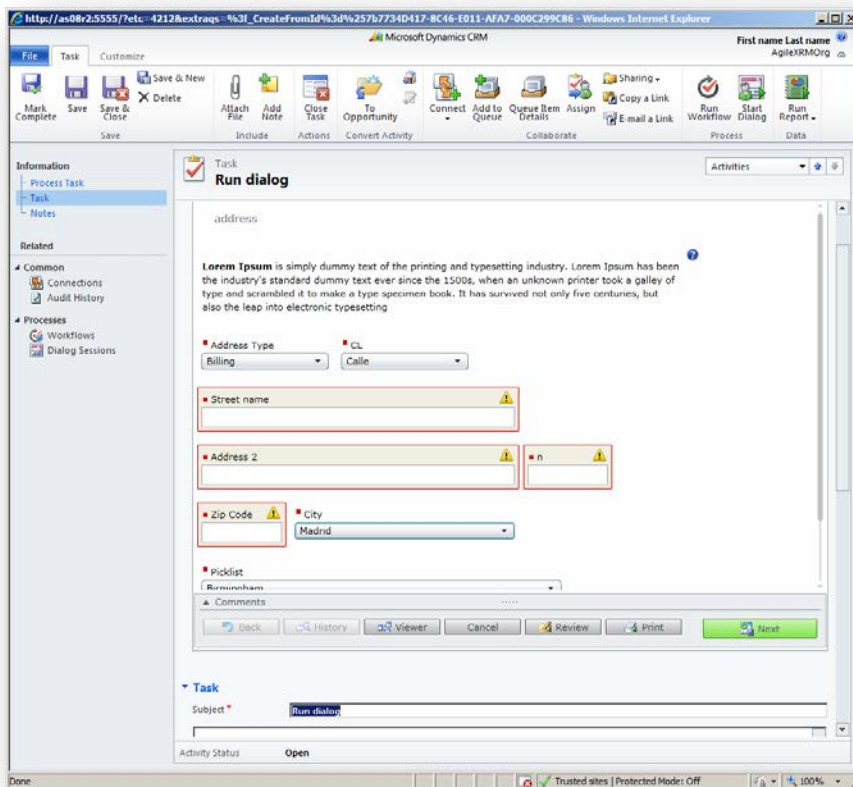
NamePrefix property

The *NamePrefix* property can be used to set a readable name for the SubDialog instance, this name is used in *AgileXRM Process Manager* to show a hyperlink to the SubDialog.

4.4 Dialog Task



This activity can only be used in an AgileXRM process template. It cannot be used in AgileDialogs templates. The activity creates a manual task that is completed when a Dialog is completed. The AgileXRM process will continue when the dialog is completed. If the dialog is canceled, a process variable called *DialogCanceled* is set to *true*. This variable can be used to take decisions in the process. The dialog is presented as an IFrame in task form:

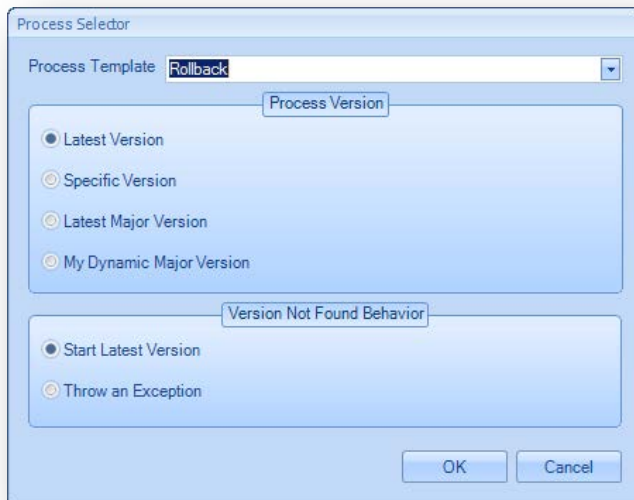


Configuration

The configuration is the same as the AgileXRM Manual Task shape; the only difference is that there is a *Dialog* property instead of a *Form* property.

Dialog property

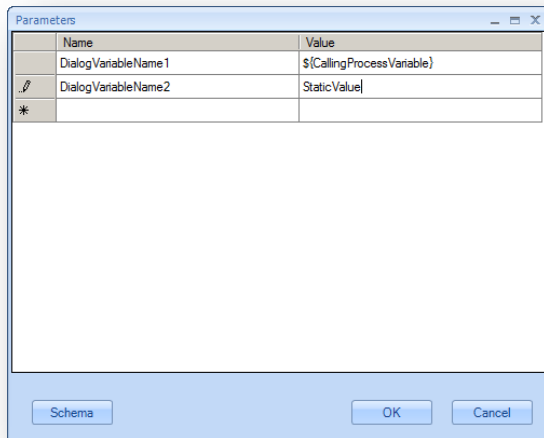
The dialog to be executed by the user is selected using *Dialog* property:



Process Template	Select which dialog to run, from the list of AgileDialogs already deployed to the Process Engine
Latest Version	The latest version of the SubDialog will be used at runtime
Specific Version (*)	The selected version will always be used
Latest Major Version (*)	The highest minor version of the selected major version is used
My Dynamic Major Version (*)	The same as Latest Major Version but the major version number is the calling dialog's Major Version
Start Latest Version (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will behave as if Latest Version was selected
Throw an exception (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will make the shape throw an Exception (i.e. process hangs in this shape until the problem is remedied)

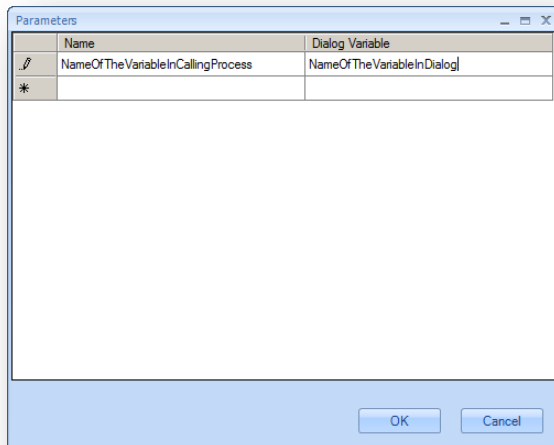
InParameters property

The dialog can receive input parameter from the calling process. These parameters can be either static values or dynamic values using the `${variable}` syntax.



OutParameters property

This configuration is used to set context variables in calling process from values gathered in the dialog.



The *Name* column contains the name of the variable to create in calling process and *Dialog Variable* column contains the name of the variable in dialog to get value from.

4.5 Intranet Dialog Task



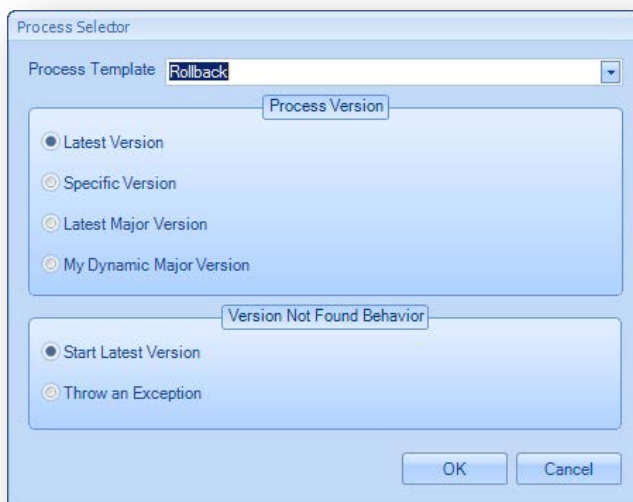
This activity can only be used in an AgileXRM process template. It cannot be used in AgileDialogs templates. The activity creates a manual task that is completed when a Dialog is completed. The AgileXRM process will continue when the dialog is completed. If the dialog is canceled, a process variable called *DialogCanceled* is set to *true*. This variable can be used to take decisions in the process. The dialog is presented to the user in the Portal.

Configuration

The configuration is the same as the AgileXRM Manual Task shape; the only difference is that there is a *Dialog* property instead of a *Form* property.

Dialog property

The dialog to be executed by the user is selected using *Dialog* property:



Process Template	Select which dialog to run, from the list of AgileDialogs already deployed to the Process Engine
Latest Version	The latest version of the SubDialog will be used at runtime
Specific Version (*)	The selected version will always be used
Latest Major Version (*)	The highest minor version of the selected major version is used
My Dynamic Major Version (*)	The same as Latest Major Version but the major version number is the calling dialog's Major Version
Start Latest Version (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will behave as if Latest Version

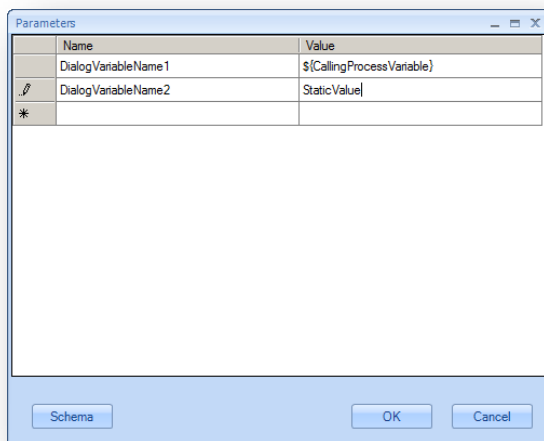
Throw an exception (if not found)

was selected

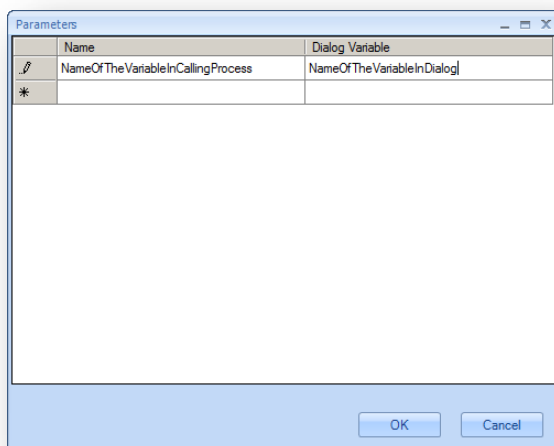
This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will make the shape throw an Exception (i.e. process hangs in this shape until the problem is remedied)

InParameters property

The dialog can receive input parameter from the calling process. These parameters can be either static values or dynamic values using the `${variable}` syntax.

**OutParameters property**

This configuration is used to set context variables in calling process from values gathered in the dialog.



The *Name* column contains the name of the variable to create in calling process and *Dialog Variable* column contains the name of the variable in dialog to get value from.

4.6 Internet Dialog Task



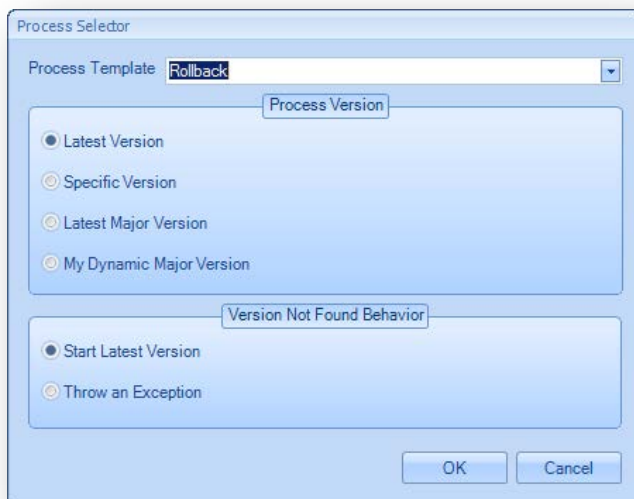
This activity can only be used in an AgileXRM process template. It cannot be used in AgileDialogs templates. The activity creates a manual task that is completed when a Dialog is completed. The AgileXRM process will continue when the dialog is completed. If the dialog is canceled, a process variable called *DialogCanceled* is set to *true*. This variable can be used to take decisions in the process. The dialog is presented to the external user in the Portal.

Configuration

The configuration is the same as the AgileXRM Manual Task shape; the only difference is that there is a *Dialog* property instead of a *Form* property.

Dialog property

The dialog to be executed by the user is selected using *Dialog* property:



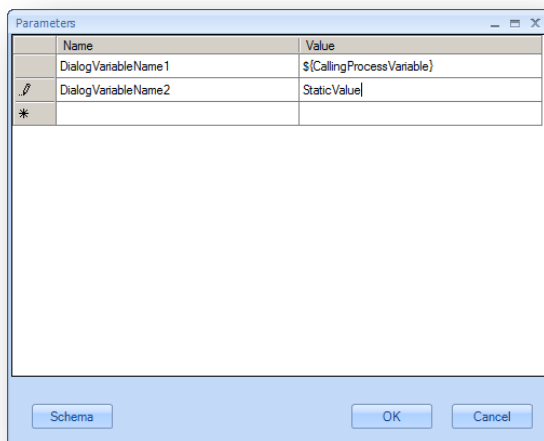
Process Template	Select which dialog to run, from the list of AgileDialogs already deployed to the Process Engine
Latest Version	The latest version of the SubDialog will be used at runtime
Specific Version (*)	The selected version will always be used
Latest Major Version (*)	The highest minor version of the selected major version is used
My Dynamic Major Version (*)	The same as Latest Major Version but the major version number is the calling dialog's Major Version
Start Latest Version (if not found)	This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will behave as if

Throw an exception (if not found)**Latest Version was selected**

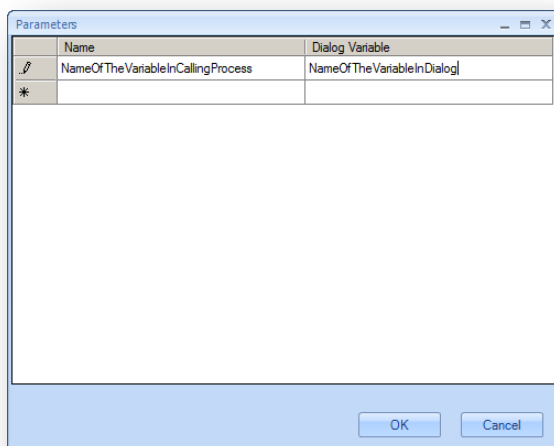
This configures the behavior of the shape if the version setting above is one of the ones with an (*) and that version does not exist at runtime. In the case it will make the shape throw an Exception (i.e. process hangs in this shape until the problem is remedied)

InParameters property

The dialog can receive input parameter from the calling process. These parameters can be either static values or dynamic values using the `${variable}` syntax.

**OutParameters property**

This configuration is used to set context variables in calling process from values gathered in the dialog.



The *Name* column contains the name of the variable to create in calling process and *Dialog Variable* column contains the name of the variable in dialog to get value from.

5. Localization

AgileDialogs contents can be localized in order to provide localized pages to the end user. This way, the same dialog can be presented to different people in different languages, avoiding the maintenance of multiple dialogs (one for each language).

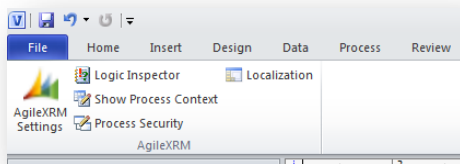
For Power Users, the language choice is based on the user's language setting in XRM.

For Standard or External Users, the language choice is based on their browser's language setting.

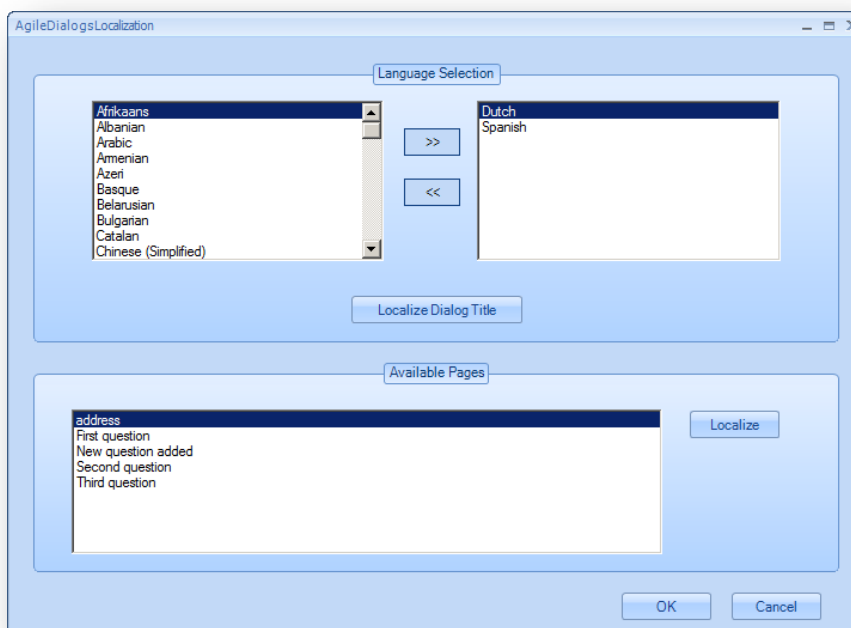
The elements that can be localized are:

- Dialog Title
- Page Title
- Question Caption
- Validation Messages
- Display values for options, checkboxes and combos
- Question Tip (Description) and extended Help

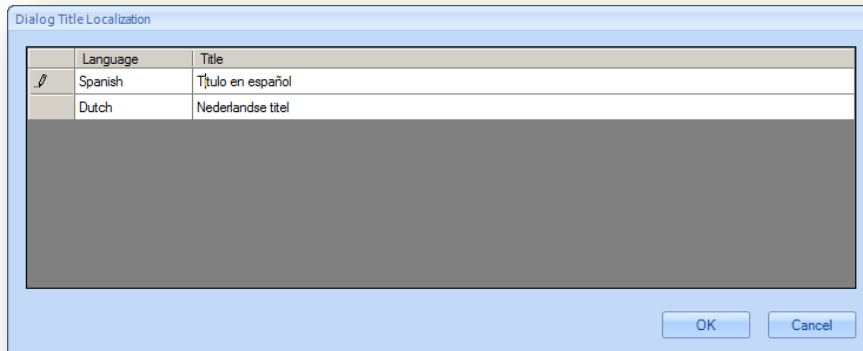
AgileDialogs localization is managed using Localization button in AgileXRM ribbon:



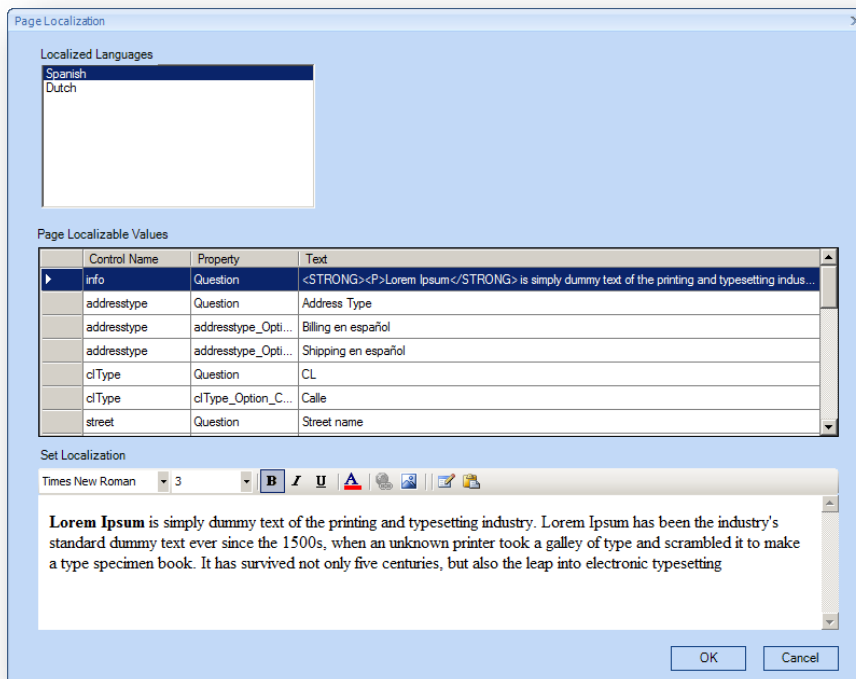
This window allows configuring localization for all dialog pages:



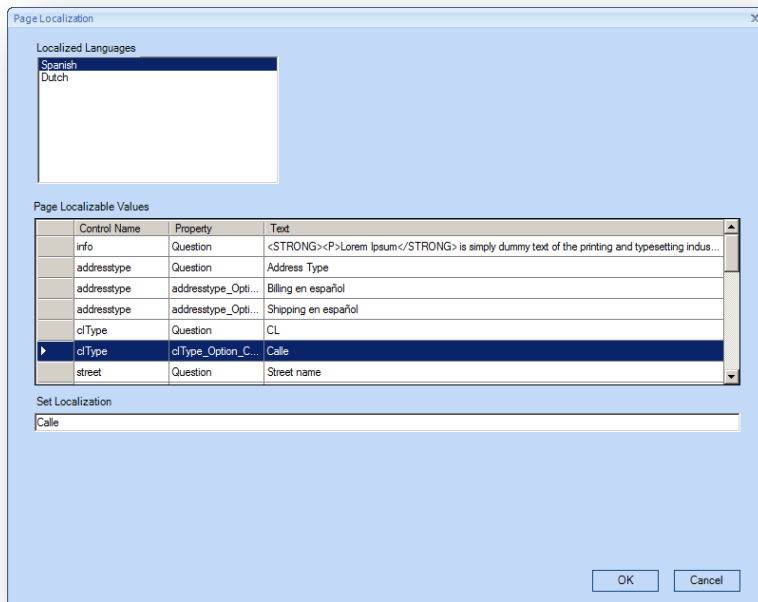
In the upper part of the window, select the languages that you want to translate the Dialog to. Click *Localize Dialog Title* to localize the title of the Dialog to the selected languages:



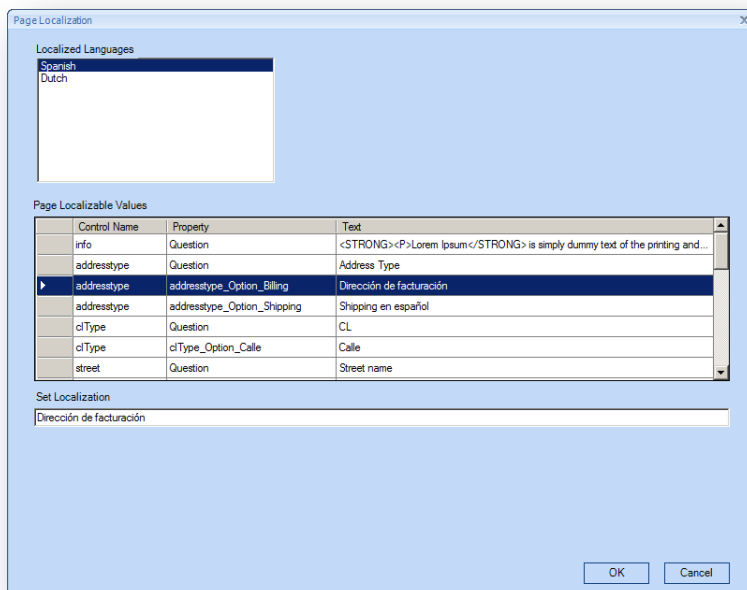
To localize the pages, select them in the list and click *Localize*:



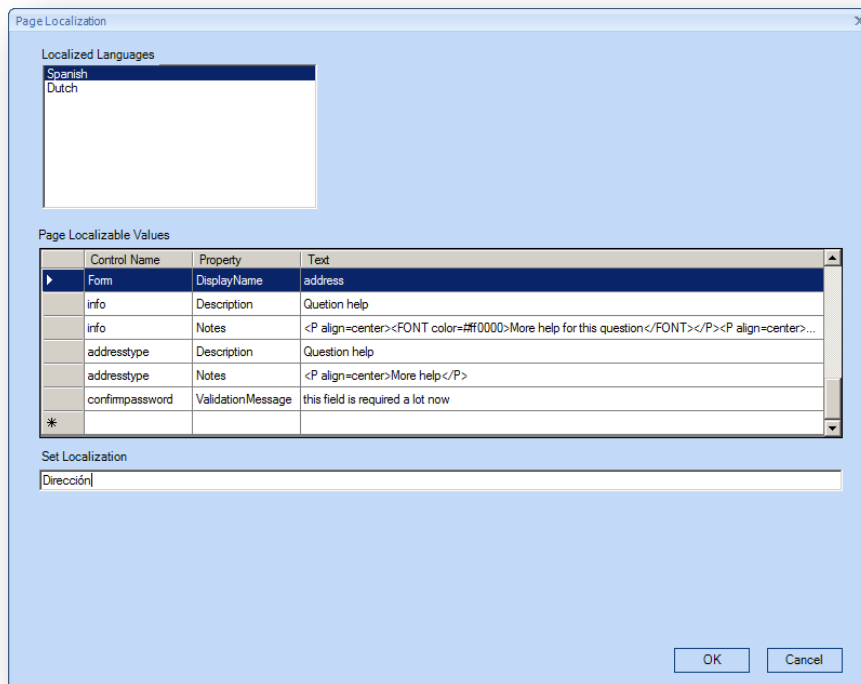
Select each literal to translate in the list of Page Localizable Values and set the localized values. The control to set the localized value changes depending on the type of value, when value allows formatting the advanced editor is presented. For simple values a textbox is presented:



This window allows the translation the display of the options for option, combo and check boxes with static values.



To localize the title of the step select the row where Control Name column value is *Form* and Property column value is *DisplayName*:



To set localizations for different languages, select them in the list of *Localized Languages* and fill localization for each localizable string.

6. Advanced Features

6.1 JavaScript Extensions

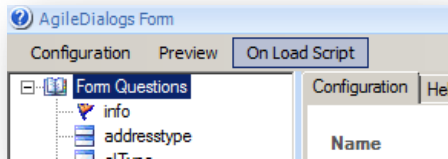
Process modeler can add JavaScript code in order to extend AgileDialogs default behavior.

These code extensions allow:

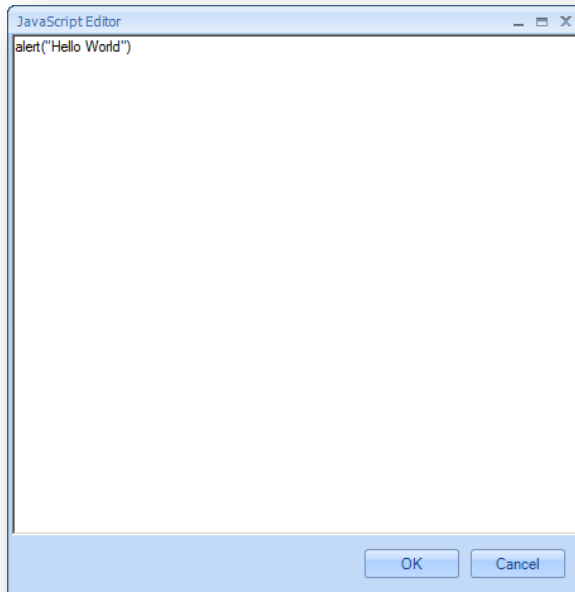
- Change control values.
- Add options to selection controls
- Remove options.
- Hide/Show controls.
- Add validation extensions.
- Subscribe to control value changes.
- Collapse or expand Comments control.
- Get Dialog Context.

AgileDialogs page includes a reference to jQuery 1.4 and JSON, so jQuery and JSON components can be used while adding code to AgileDialogs.

To add code to a Dialog lick *On Load Script* in AgileDialogs form window:



Doing so the script editor is opened to include custom code:



The code included in this window is executed after the form has been loaded in the client.

6.1.1 Change Control Value

To change a control value, add this code:

```
DialogsEngine.setSelectedValue(<controlName>, <newValue>);
```

Where:

- **controlName** is the value of *ValueVariable* parameter in the question.
- **newValue** is the value to set.

This code can be used to set selected values in textboxes, combos, grids,...

6.1.2 Add options to a selection control

To add an option to a combo, radio or checkbox control, add this code:

```
DialogsEngine.addOption(<controlName>,<newOptionDisplay>,<newOptionValue>);
```

Where:

- **controlName** is the value of *ValueVariable* parameter in the question.
- **newOptionDisplay** is the display value of the new option.
- **newOptionValue** is the value of the new option.

6.1.3 Remove options from a selection control

To remove an option from a combo, radio or checkbox control, add this code:

```
DialogsEngine.removeOption(<controlName>,<optionValue>);
```

Where:

- **controlName** is the value of *ValueVariable* parameter in the question.
- **optionValue** is the value of the option to remove.

6.1.4 Hide/Show controls

To hide or show a control (with the caption of the question), add this code:

```
DialogsEngine.setVisible(<controlName>,<isVisible>);
```

Where:

- **controlName** is the value of *ValueVariable* parameter in the question.
- **isVisible** Boolean value, true to show control, false to Hide control.

6.1.5 Add validation extensions

You can add code to validate control values. To do that subscribe to form validation event and the event handler will be called when the validation is performed by AgileDialogs engine.

If validation is not OK return a string with the message that will be presented to the user.

In order to subscribe to validation events add this code:

```
function myValidationFunction(controlName, controValue){  
  //Add your validation logic  
  if(notValid){  
    return "This value is not valid";  
  }  
}  
validationHandler.addHandler(myValidationFunction);
```

`addHandler` method will add a handler to be called when controls are validated by AgileDialogs Runtime Engine. The method added (in this sample `myValidationFunction`) will receive 2 parameters:

- **controlName**: Name of the control to be validated.

- **controValue:** Current value of the control.

To show a validation message the validation function should return a string message.

6.1.6 Subscribe to control value changes

In order to manage value changes in controls his code must be added:

```
function myChangeHandler (controlName,controlValue){
//add your logic here
}
fieldChangeHandler.addHandler(myChangeHandler);
```

myChangeHandler method will add a handler to be called when a control value is changed. The method added (in this sample myChangeHandler) will receive 2 parameters:

- **controlName:** Name of the control to be validated.
- **controValue:** Current value of the control.

6.1.7 Collapse or expand Comments control.

The control used to get comments from user can be expanded or collapsed using this code:

```
DialogsEngine.toggleNotes();
```

6.1.8 Get Dialog Context

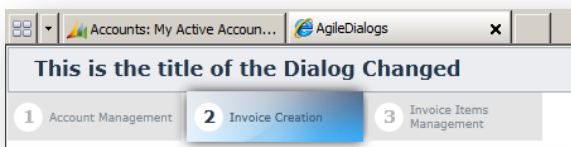
The values selected in other pages by the user can be accessed using this code:

```
var dialogContext = getDialogContext();
var contextValue = arrContext["attrName"];
```

dialogContext returns a collection of name-values with all Dialog context variables.

6.2 Breadcrumbs

AgileDialogs has a feature that allows including Breadcrumbs while the dialog is executed by the user. These bread crumbs are presented in the top of the AgileDialogs interface:

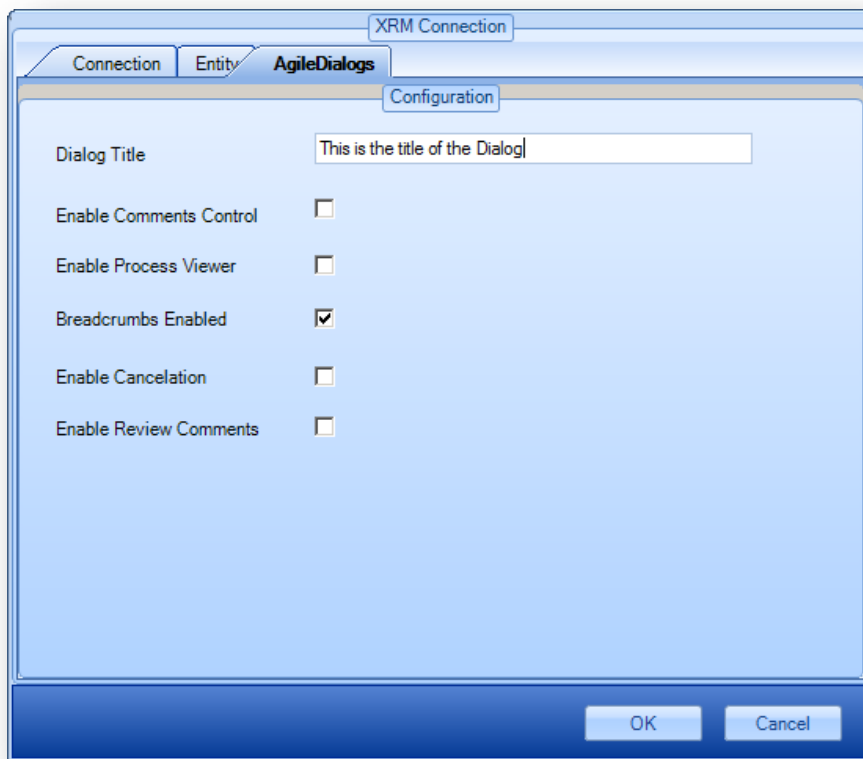


In this sample the user is in the second phase of the dialog (*Invoice Creation*).

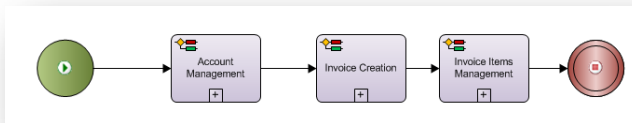
This feature is used to show the user where he/she is in the Dialog.

To configure a Dialog with this behavior:

- Create a parent process that will contain the phases of the dialog.
- In XRM Connection window check *Breadcrumbs Enabled*.



- Each phase will be a SubDialog step and the name of the SubDialog shape will be the name of the phase. This feature can be used only in processes where all phases must be fulfilled, there cannot be decisions between phases, and parent process only allows SubDialogs shapes.



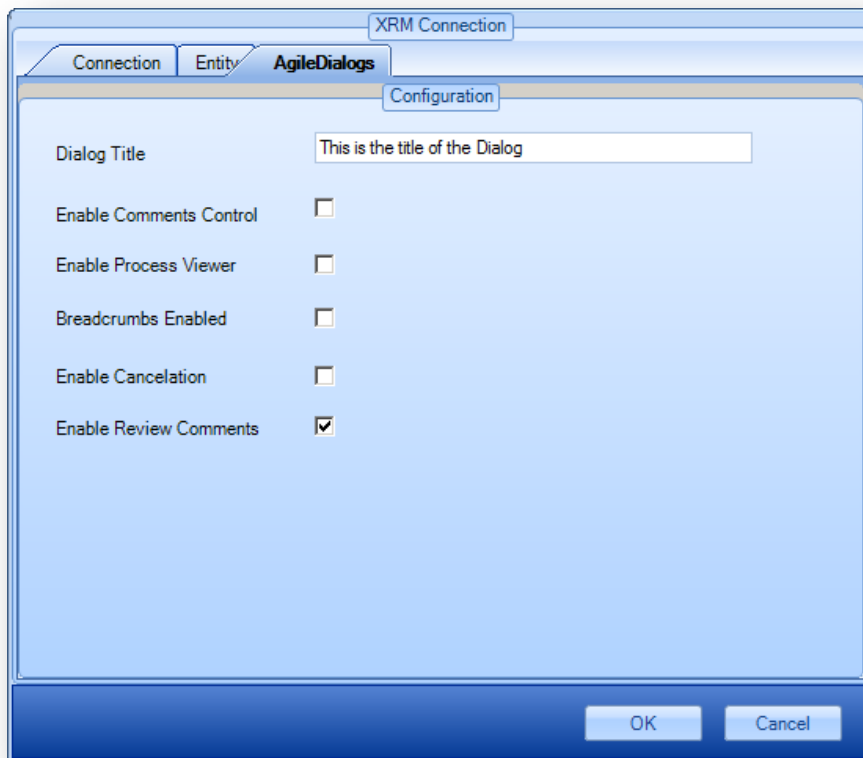
6.3 Custom Controls SDK

See AgileDialogs SDK documentation.

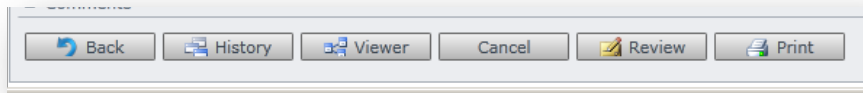
6.4 Dialogs Review

AgileDialogs infrastructure allows involving users in the design of the dialogs.

To allow users add suggestions or comments enable review comments in *XRM Connection* window:



Doing that, a button to add comments (*Review*) is added to Dialogs Runtime:



When a user clicks this button this window is presented to allow the user adding comments:

Remove	Step	Comment	Created
☒	address (AgileScriptsDialogs.80)	Missing option in check box list: "Shipping"	6/14/2011

Comments

Wrong control size

Add Comment Cancel

In the top of the window the user can see the comments included by him/her for the current dialog instance and the step that the comment is linked to.

All comments are saved in XRM in an entity called *AgileDialogs Review Comments*.

Each comment has a set of information associated to help the designer to improve the Dialog: the step (AgileDialog Page) where the comment was included, the name of the Dialog, the version, the date and the user that added the comment.

activityID	Created On	Dialog Template	Dialog Version	Step Name	Username	Status	Status Reason
FF7308A307B846...	6/30/2011 7:59 PM	AgileDialogs	1.31	address (AgileScriptsDialogs.80)	AGILEPOINT\Administra...	Active	New
FF7308A307B846...	6/30/2011 8:08 PM	AgileDialogs	1.31	address (AgileScriptsDialogs.80)	AGILEPOINT\Administra...	Active	New

Using XRM views the designer can manage the comments fixing the issues or suggestions that the reviewers have created.

6.5 Preview Configuration

While configuring a Dialog the user that is modeling can preview the pages as the end user will see them (see bottom right in this image):

The screenshot shows the 'AgileDialogs Preview Configuration' window. On the left, there's a 'Form Properties' panel with options like 'Disable Back (Clear History)' and 'Print Enabled'. The main area displays the configuration for a form element named 'confirmpassword'. It includes fields for 'Display Name' (confirmpassword), 'Type' (PASSWORD), and 'Question Caption' (confirm). Below this, there's a preview of the dialog form, which includes fields for 'Address Type' (Billing), 'Street name', 'Address 2', 'n', 'Zip Code', 'City' (Albacete), 'password', and 'confirm'. The preview also shows a 'Lorem Ipsum' text block.

In order to make this preview work some configuration must be set.

To set preview configuration click in Configuration menu and set the right values in this window:

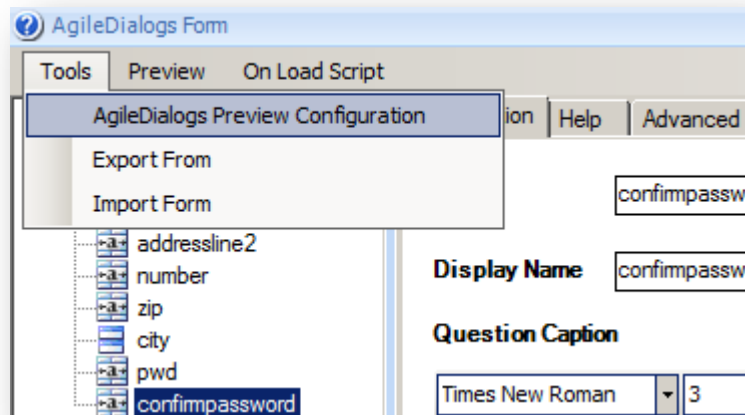
The screenshot shows the 'AgileDialogs Preview Configuration' window. It features a text field for the URL: `http://localhost:8888/AgileDialogsweb/AgileScriptsSLRunTime.aspx?orgname=agilexmorg`. Below the URL field, there's a 'Window Size' section with input fields for 'Width' (800) and 'Height' (800). An 'OK' button is located at the bottom right.

The first parameter is the URL of the AgileDialogs page and must include the name of the organization in XRM to get data from.

Window size parameters are useful to see the preview with the expected resolution of the users. Both parameters are shared for all previews for the user.

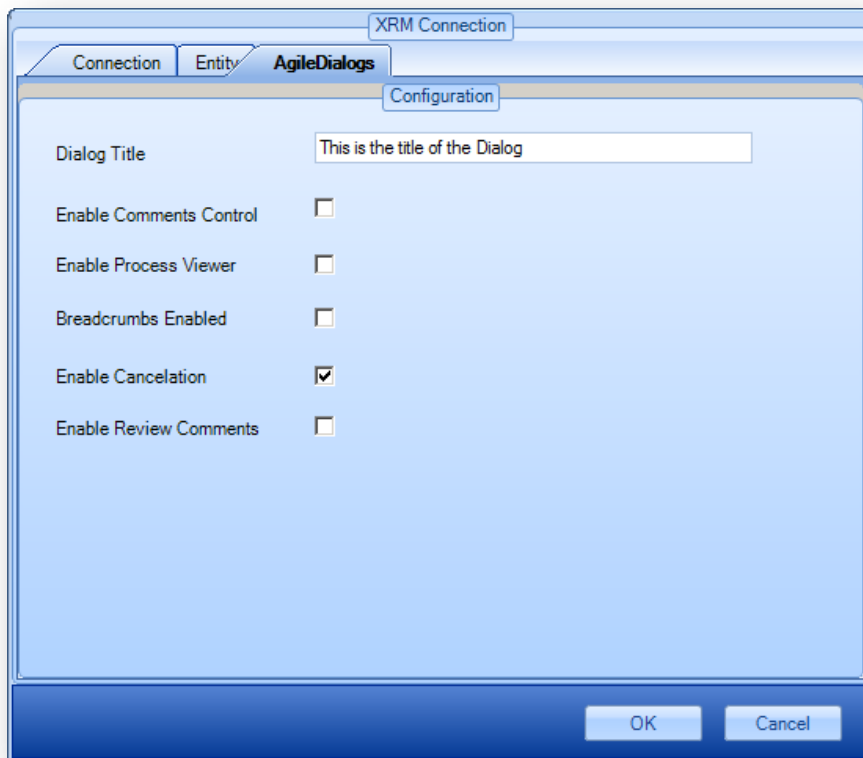
6.6 Exporting and Importing Form configuration

The form definition can be exported to a XML file or imported from a XML file using the menu options available in Tools menu:



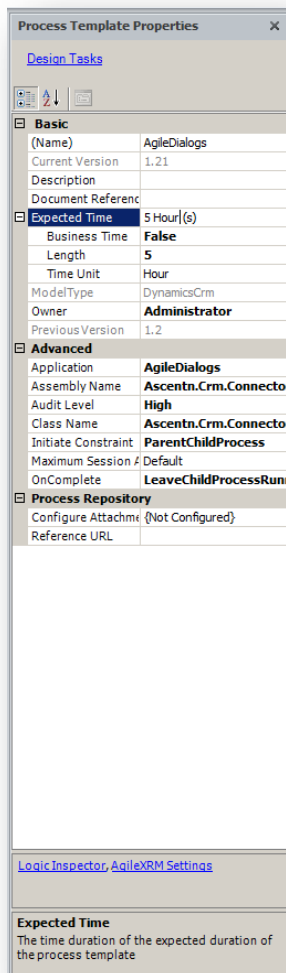
6.7 AgileDialogs cancelation

AgileDialogs that are associated to a task (either intranet task or XRM task) can be cancelled by the user if the modeler allows it setting the Enable Cancelation parameter to true:



When the dialog is canceled the corresponding task is canceled and the process continues. This dialogs can be automatically canceled when a timer is configured on the corresponding tasks and the task is overdue.

While the dialog is not canceled the user can suspend and open it in the last page whenever he wants. Dialogs that are run on demand (that is, they are not associated with a task) are automatically canceled when Expected Time parameter of the dialog is exceeded:



The screenshot shows the 'Process Template Properties' dialog box with the 'Design Tasks' tab selected. The dialog is divided into four main sections: Basic, Expected Time, Advanced, and Process Repository. The 'Expected Time' section is currently selected and highlighted.

Basic	
(Name)	AgileDialogs
Current Version	1.21
Description	
Document Reference	

Expected Time	
Expected Time	5 Hour(s)
Business Time	False
Length	5
Time Unit	Hour
ModelType	DynamicsCrm
Owner	Administrator
Previous Version	1.2

Advanced	
Application	AgileDialogs
Assembly Name	Ascentn.Crm.Connecto
Audit Level	High
Class Name	Ascentn.Crm.Connecto
Initiate Constraint	ParentChildProcess
Maximum Session /	Default
OnComplete	LeaveChildProcessRun

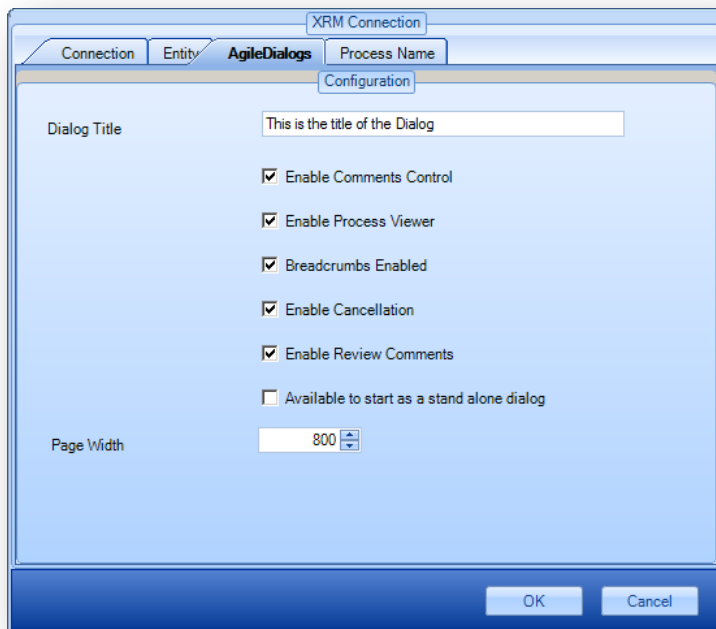
Process Repository	
Configure Attachme	{Not Configured}
Reference URL	

[Logic Inspector, AgileXRM Settings](#)

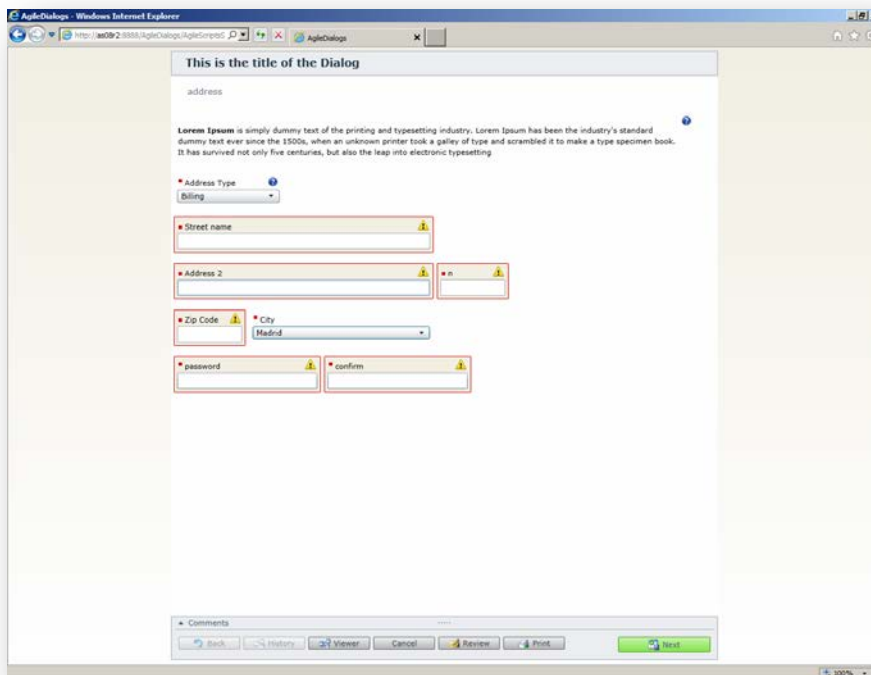
Expected Time
The time duration of the expected duration of the process template

6.8 Setting a fixed width for the dialog at runtime

Using the Width setting in AgileDialogs tab:



The width of the pages of the dialog can be fixed at runtime so that the content of the dialog will be centered in the screen using the configured width (in pixels):



6.9 Launching AgileDialogs on Demand

AgileDialogs can be launched using a URL, there is no need to have either an associated XRM record nor an AgileXRM task.

An AgileDialog can be used, for instance, to create a record in XRM helping the user to fill the data avoiding the use of complex forms or to get data to call other system using a web service.

The format of the URL to launch a dialog is:

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>
```

This URL's can be used in portals to allow the users starting dialogs on demand.

In order to make this work the user must have permission to start instances of the dialog template.

When the dialog is localized and the link must open a specific language, the URL can contain the Locale ID of the language, the format of the URL in this case is:

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>&lcid=<USER_LOCALE_ID>
```

For a list of Locale ID's view [http://msdn.microsoft.com/en-us/library/0h88fahh\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/0h88fahh(v=vs.85).aspx).

The value to pass is in the column *Decimal Value*, for instance for *Spanish 3082*, for *French 1036*, ...

When creating a link to open a dialog it is also possible to specify what to do when the dialog is finished. There are several options based on the use of the query string parameter *AfterSubmit*:

- Close the window when dialog finishes. This is useful when opening the dialog in a new window.

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>&AfterSubmit=Close
```

- *Refresh the dialogs page when dialog finishes. This is useful to refresh the Dialogs Engine page to run another dialog.*

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>&AfterSubmit=self
```

- Redirect to another URL. This approach is used to return to the calling page instead of closing the window.

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>&AfterSubmit=<THE_NEW_URL>
```

6.10 Creating links to complete tasks associated to a Dialog

Sometimes the process modeler needs to send an email to a user with a link to complete a task that consists of a dialog.

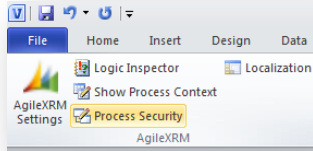
The format of this URL is:

```
http://<SERVER_URL>/AgileDialogs/AgileScriptsSLRunTime.aspx?orgname=<ORGANIZATION_NAME>&wid=<AGILEXRM_WORKITEM_ID>&DefaultProcessTemplate=<PROCESS_TEMPLATE_NAME>
```

When the user opens this URL the dialog is either started (the first time that the user opens this URL) or resumed if the user has not completed the Dialog yet.

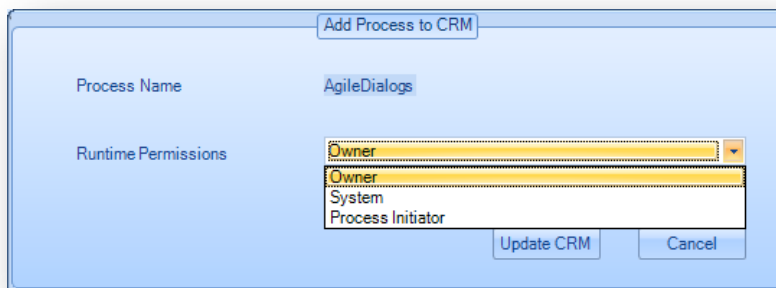
6.11 AgileDialogs Security

Configuring permissions for AgileDialogs works like configuring permissions for AgileXRM process, when a Dialog is deployed to server the modeler can configure permissions using the same mechanism that is used for AgileXRM processes.

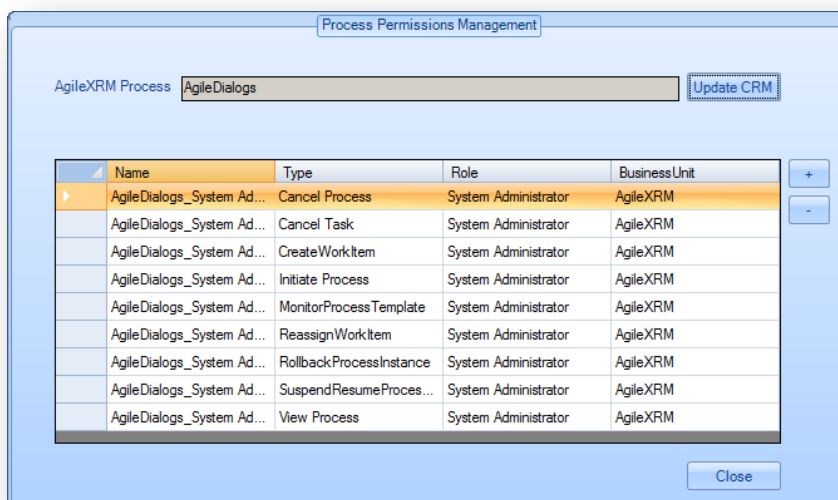


Process permission can be managed using Process Security button in AgileXRM Ribbon.

Runtime Permissions have the same configuration options that AgileXRM processes



And permission can be assigned based on XRM roles:



In order to launch a Dialog the user must belong to a role that has *Initiate Process* permission. AgileMonitor can be used to monitor the state of the instances of a dialog template. The permission *MonitorProcessTemplate* is assigned to allow the users to monitor a specific AgileDialog template.