AGILELIGHTFORMS DESIGN GUIDE

DOC ID: 12111-ALD-1.0

Table of Contents

1. Introduction 4
   - 1.1 Disclaimer of warranty 4

2. Managing Forms in CRM 5

3. Creating new Forms 9
   - 3.1 Form to Complete a Manual CRM Activity 9
   - 3.2 Form to Complete an External Manual Task 10
   - 3.3 Form to Create Child Entities 11
   - 3.4 Form to View or Edit Child Entities 12
   - 3.5 Form To Create Entities and Launch a Process 13

4. AgileLightForms Designer Tool Bar 13
   - 4.1 Language Combo 14
   - 4.2 Export Form… Button 15
   - 4.3 Import Form… Button 15
   - 4.4 Upload Handler… Button 16
   - 4.5 Remove Handler Button 16
   - 4.6 Full-screen Button 16
   - 4.7 Clear Cache Button 16

5. Connection Editor 17
   - 5.1 Entity Editor 17
   - 5.2 Entity Validation 18
   - 5.3 AgilePoint Connection 18
   - 5.4 CRM Connection 19

6. Schema Designer 20
   - 6.1 Result Metadata 21
     - 6.1.1 Result Metadata Attributes 21
   - 6.2 Data Binding 25
   - 6.3 Localizable Texts and Concatenations 27
   - 6.4 Entity Bar 27
     - 6.4.1 Parameters Entity 28
     - 6.4.2 Work Item Entity 28
     - 6.4.3 Process Instance Entity 28
     - 6.4.4 Custom Attributes Entity 28
     - 6.4.5 AgileXRM User Entity 29
     - 6.4.6 AgileXRM External User Entity 29
     - 6.4.7 New CRM Entity 30
     - 6.4.8 Existing CRM Entity 30

7. Submit Action Editor 31
   - 7.1 Complete Form Submit Action 32
   - 7.2 Complete Work Item Submit Action 33

7.3 Launch Process 1 Submit Action 33
7.4 Launch Process 2 Submit Action 34
7.5 Complete Activity Submit Action 34

8. Form Editor 35
8.1 Entity Bar 35
8.2 Design Surface 37
8.3 Actions Pane 39
8.3.1 Movement Pane 39
8.3.2 Command Pane 39
8.3.3 Form Properties 39
8.3.4 Tab Properties 40
8.3.5 Section Properties 40
8.3.6 Attribute Properties 41
8.3.7 Control Properties 41

AgileLightForms Design Guide

1. Introduction

This document will describe how to design AgileLightForms (ALF).

Previous knowledge of SharePoint, CRM, and AgilePoint are required.

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.

2. Managing Forms in CRM

AgileLightForms are CRM records (ascentn_agilelightforms) that can be managed like any other CRM record. Forms can be added, deleted, opened and searched for in the usual way.

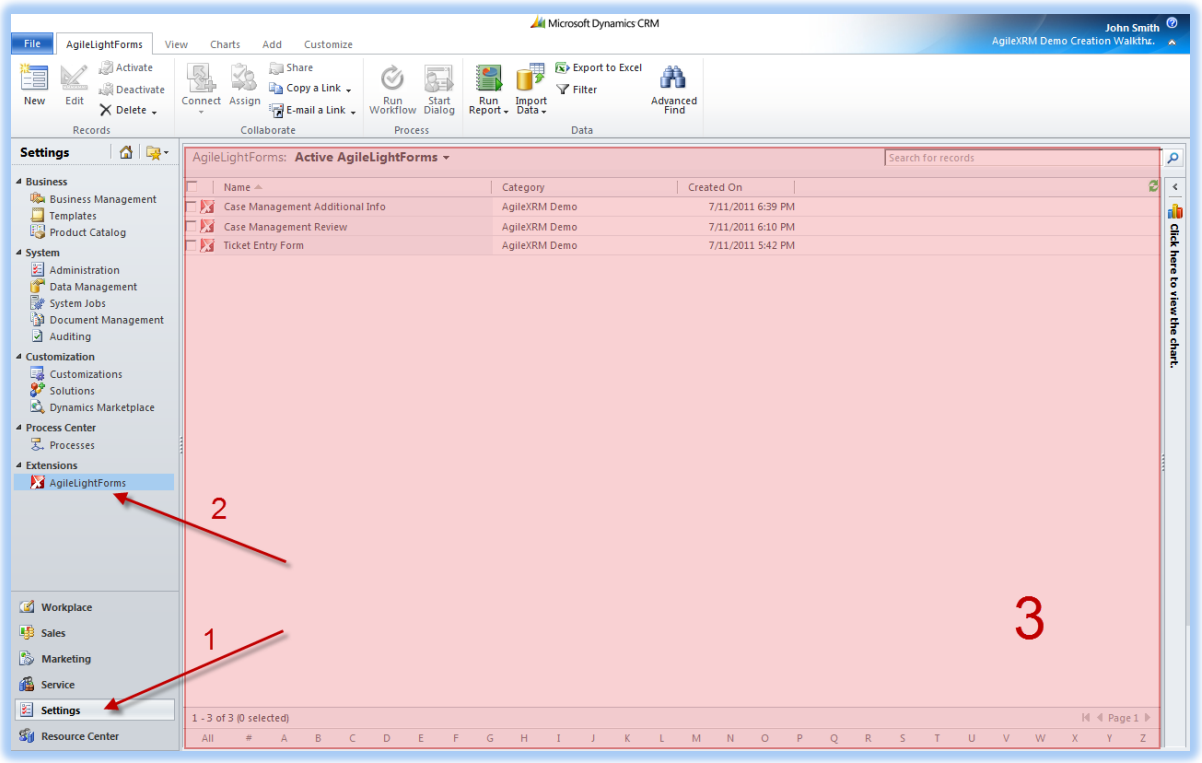


Figure 1 - Managing AgileLightForms in CRM

Form Designers must be CRM users with, at least, permissions to manage AgileLightForms and access metadata. Permissions to manage other entities can be required to create forms to access those entities. For a complete guide on required permissions, see the *AgileXRM Reference Architecture* document.

AgileLightForms can be accessed from the Settings Area in the main CRM page, after AgileXRM installation. This can be changed using standard CRM customization.

NOTE: Do not rename forms once they're referenced by a process template (a form can be referenced to be a start-up form, a task form, or a child view form). If you rename a referenced form, AgileXRM won't be able to find it, and process execution will fail.

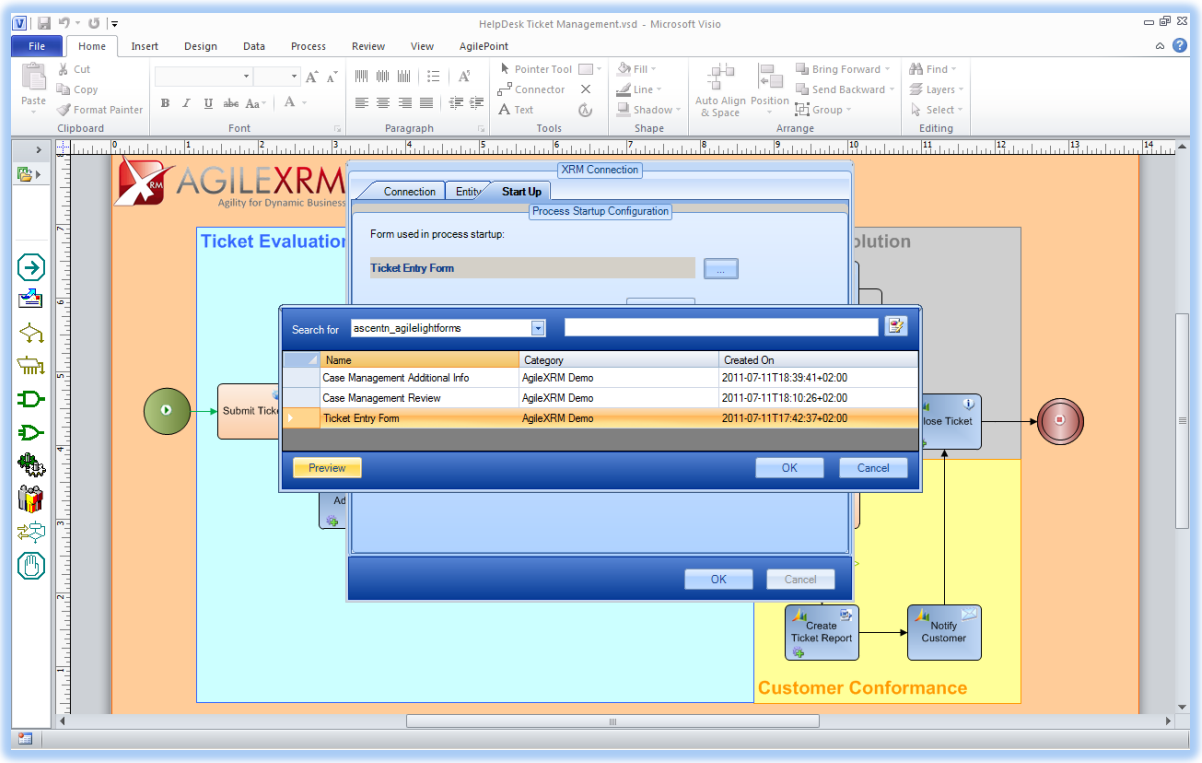


Figure 2 – AgileLightForm referenced as AgileXRM Process Start-up Form

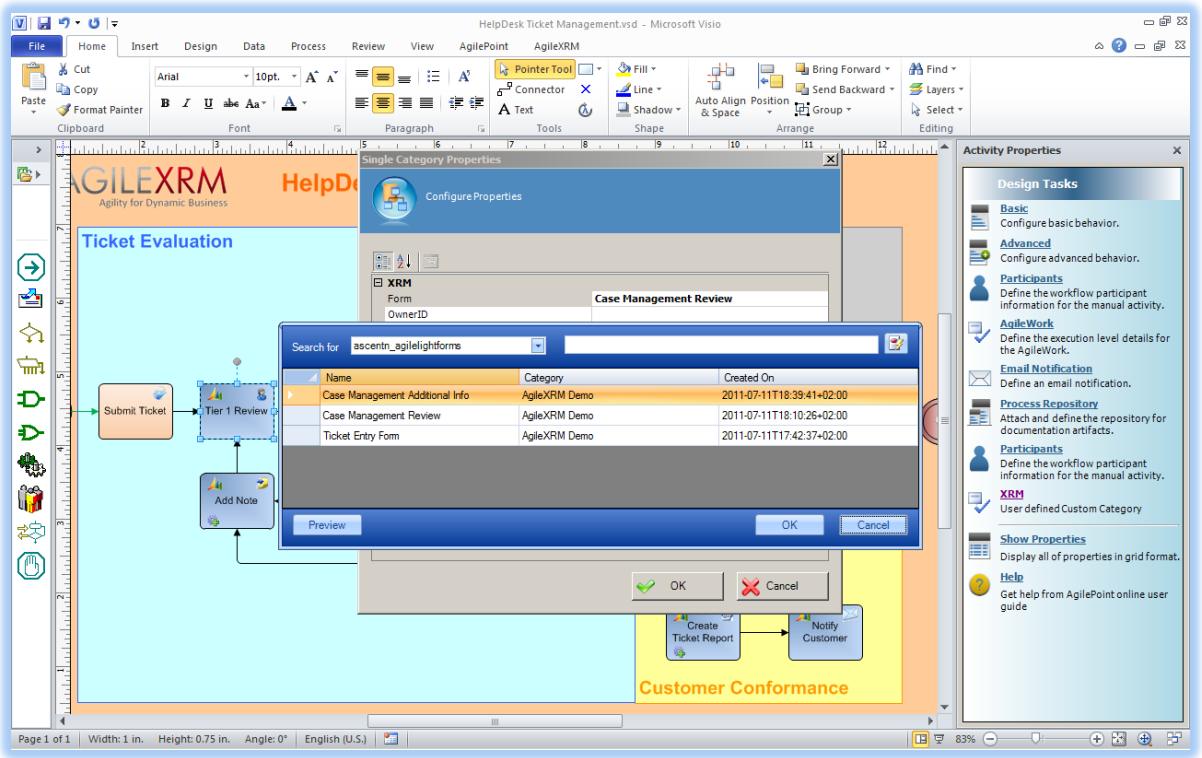


Figure 3 - AgileLightForm referenced as CRM Manual Activity Form

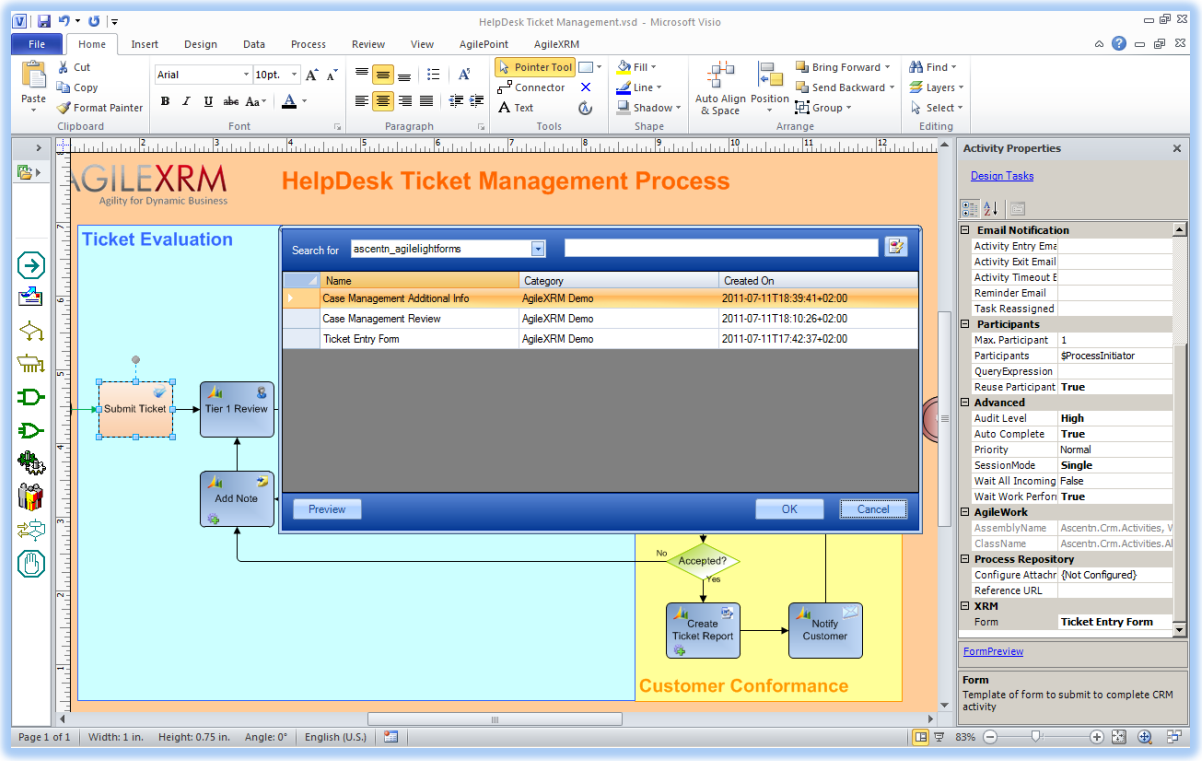


Figure 4 - AgileLightForm referenced as External Task Form

You can further customize the entity to add other attributes, and edit the CRM form, but you should never delete any of the standard attributes.

Customizing the default public view of AgileLightForms not only changes the way in which forms are displayed in CRM by default, but also how they are displayed in AgilePoint Envision and inside AgileLightForms Designer.

AgileLightForms Designer is embedded in the CRM form for AgileLightForms. It is in an IFrame. If you customize this form, take care not to remove this IFrame. Keep in mind too that reducing the size of the IFrame could adversely affect the usability of AgileLightForms Designer.

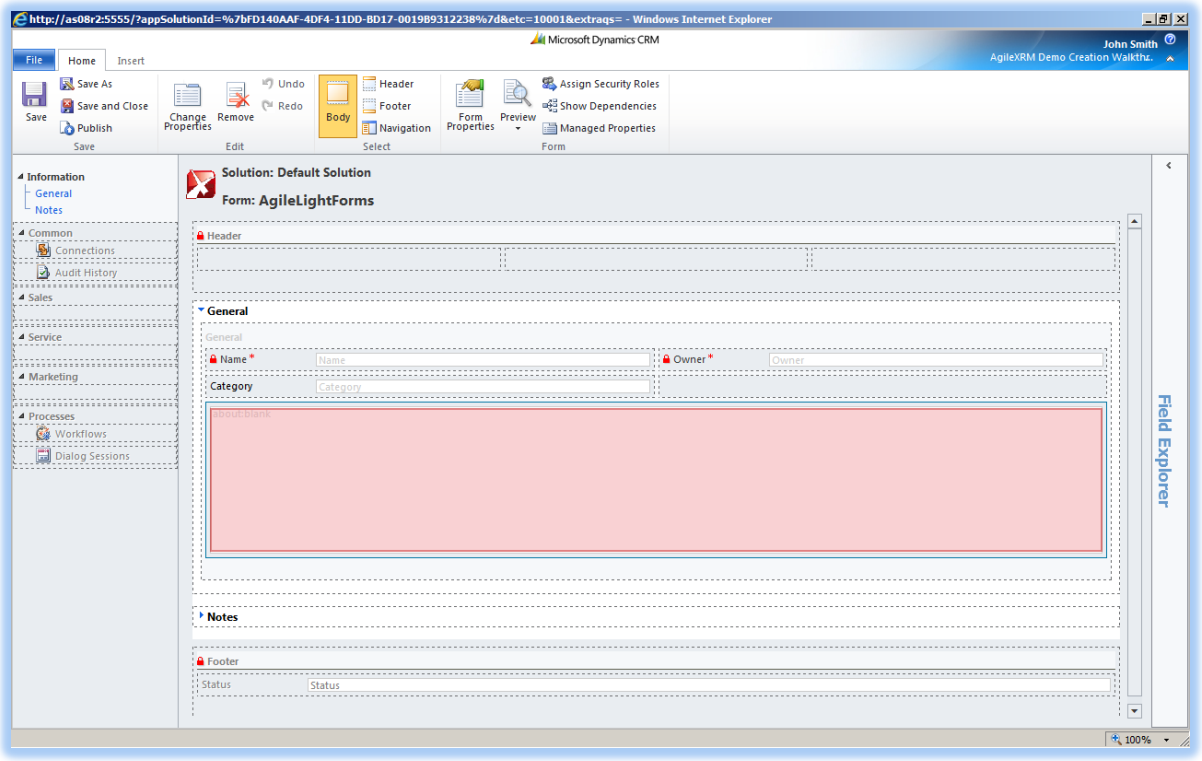


Figure 5 - Design Area in CRM Form for AgileLightForms

Actual form design is stored in a note called AgileLightForm.zip. If you delete this note, you'll lose your form design. Nevertheless, this can be useful if your form is corrupt for whatever the reason.

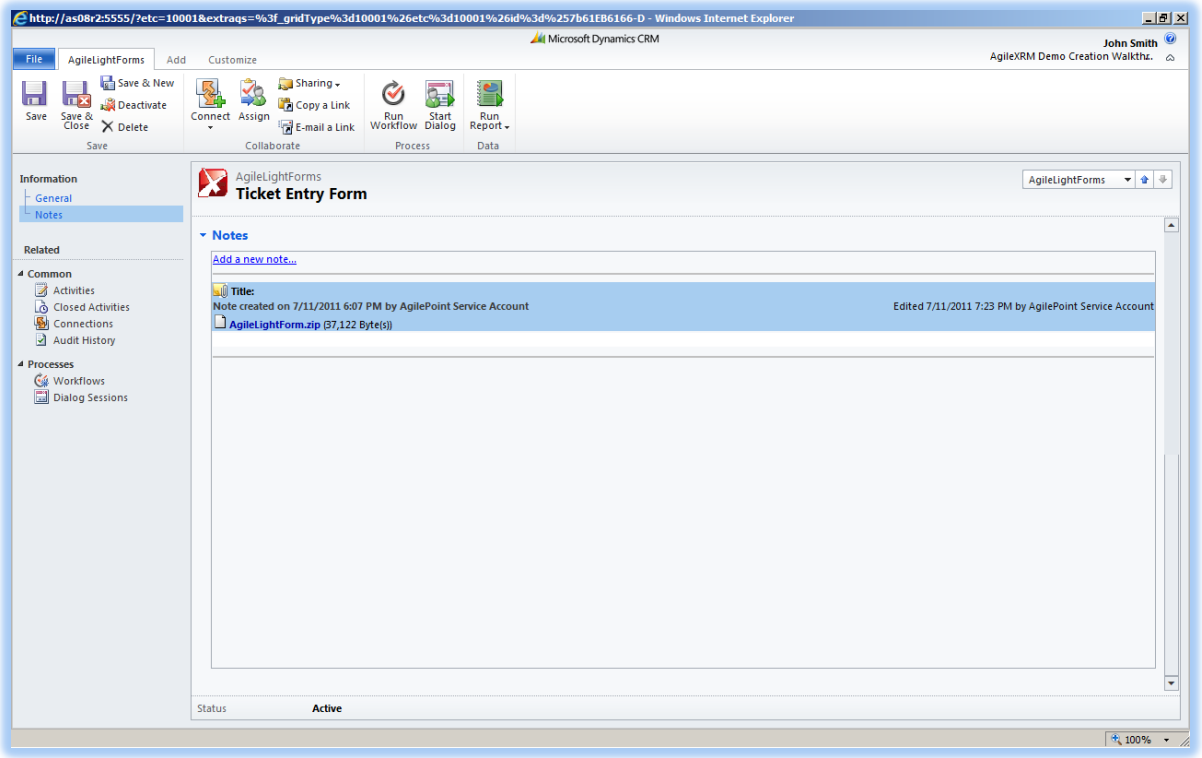


Figure 6 - AgileLightForm design Note

3. Creating new Forms

After creating a new form, you must give it a name and save it before the designer appears. If ALF designer can't find the form design (the AgileLightForm.zip note), a window appears asking which template to use to create the new form.
This only affects to the initial layout of the new form. You can freely edit them to create mixed forms, or any other variety not listed here.

Figure 7 - New Form Templates

3.1 Form to Complete a Manual CRM Activity

This template creates a form that is designed to be embedded inside a CRM Activity Form (Task, Phone Call, Fax…). It connects to the default CRM and AgilePoint servers, gets the WorkItemId as a form parameter, and uses it retrieve the Work Item, the Process Instance, and the Custom Attributes from AgilePoint Server. Then, it uses the Custom Attributes to retrieve the process main entity from CRM Server.

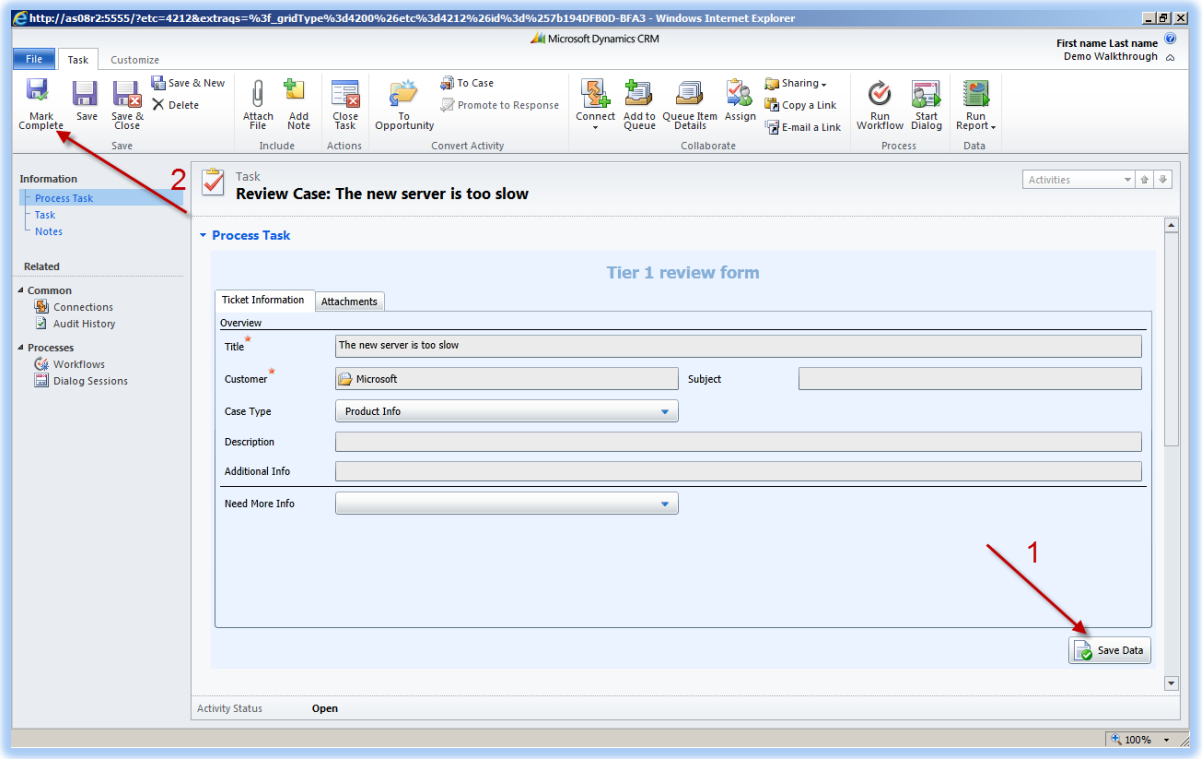


Figure 8 - Form to Complete a Manual CRM Task

This form template doesn't do anything on submit, apart from completing the form. So, the user must complete the CRM Activity in the usual way for the process to continue.

3.2 Form to Complete an External Manual Task

This template creates a form that is designed to fulfill an external task. It will be opened from the Task List AgileXRM WebPart in SharePoint. It connects to the default CRM and AgilePoint servers. It gets the WorkItemId as a form parameter, and uses it to retrieve the Work Item, the Process Instance, and the Custom Attributes from AgilePoint Server. Then, it uses the Custom Attributes to retrieve the process main entity from CRM Server.

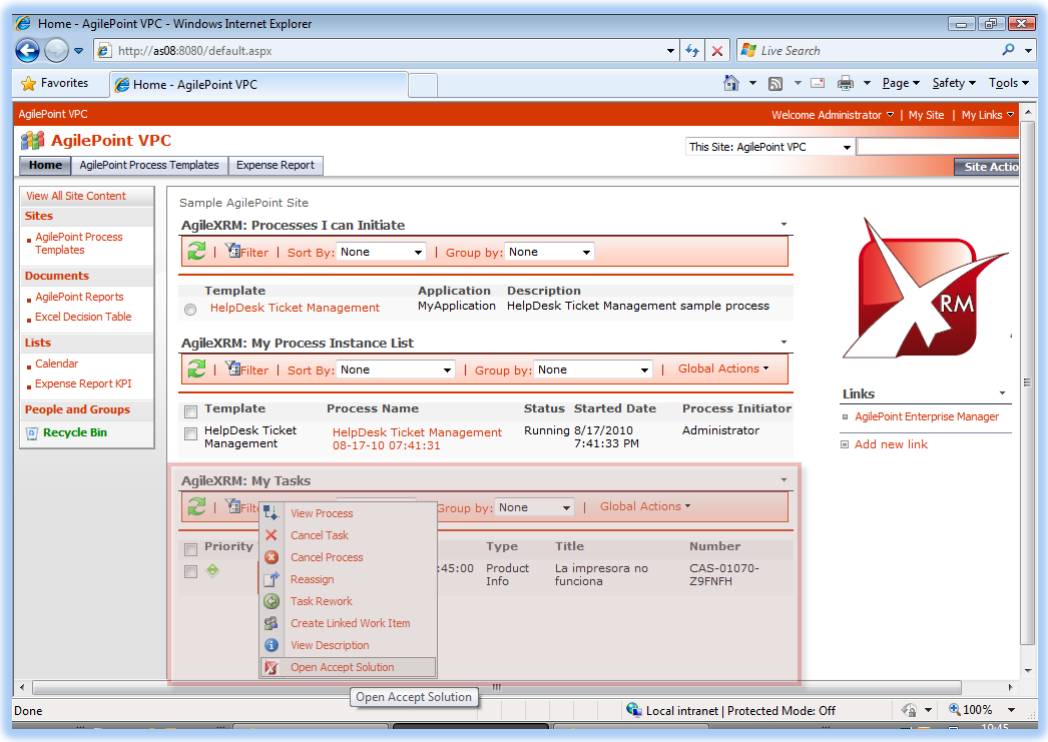


Figure 9 - AgileXRM Task List WebPart in a SharePoint site

On submit, this form completes both the form and the AgilePoint Work Item.

3.3 Form to Create Child Entities

This template creates a form that is designed to create child entities as a child form of the Child View control. It appears whenever a user clicks the Add button.

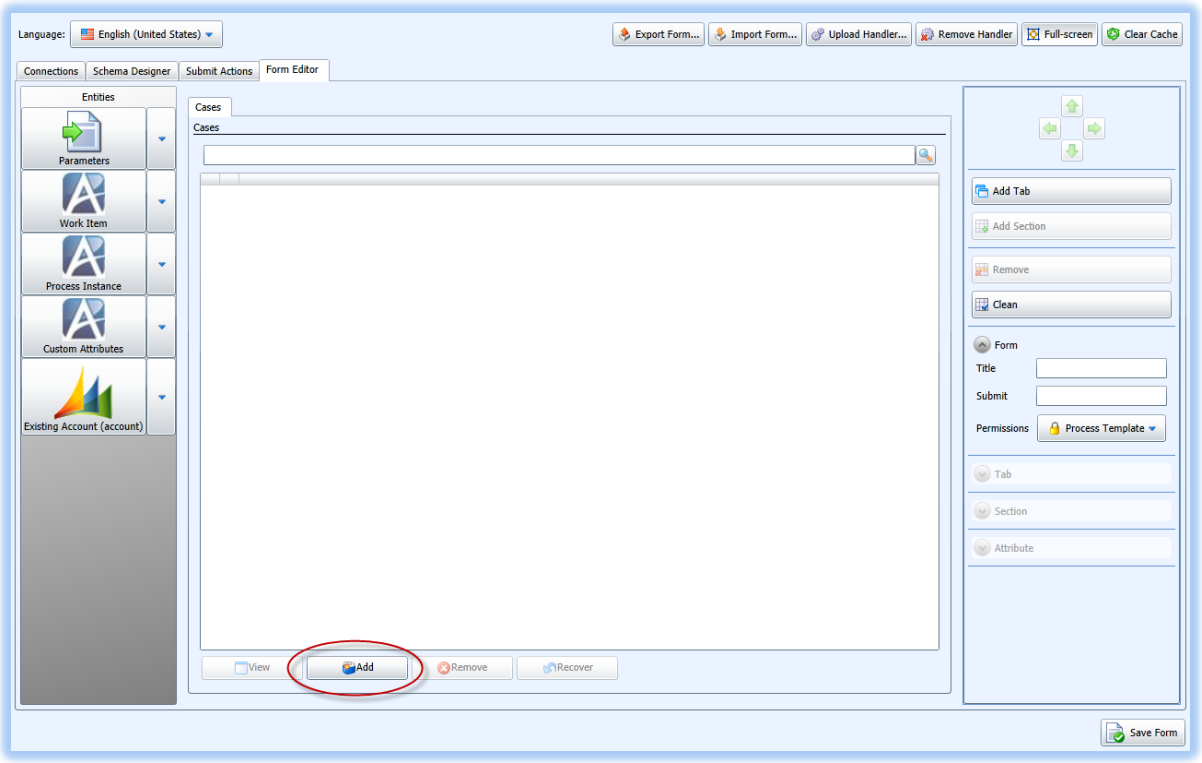


Figure 10 - Add Child Button in Child View Control

On submit, the new entity is added to the parent form. It is not really created in CRM until the main form is submitted.

3.4 Form to View or Edit Child Entities

This template creates a form that is designed to view or edit child entities as a child form of the Child View control. It appears whenever a user clicks the View or Update buttons.

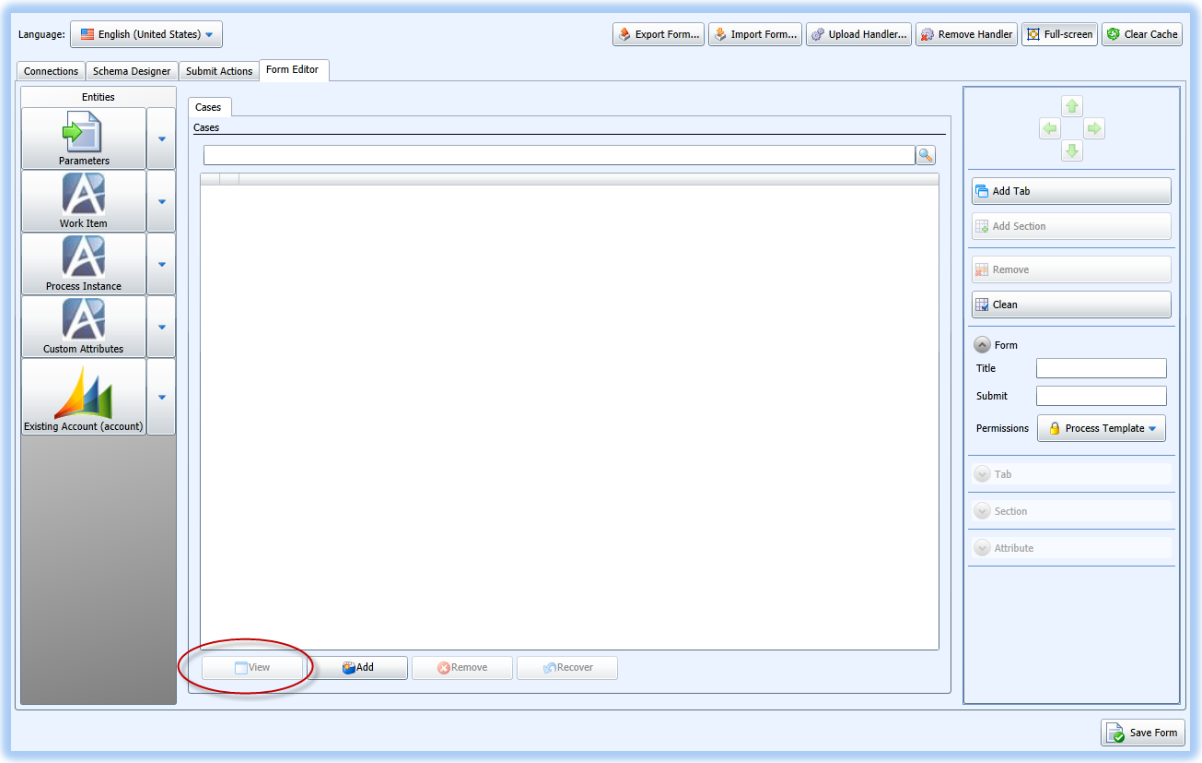


Figure 11 - View Child Button in Child View Control

If editing is allowed, on submit, the entity is updated in the parent form. It is not really updated in CRM until the main form is submitted.

3.5 Form To Create Entities and Launch a Process

This template creates a form that is designed to create a new CRM entity and launch an AgilePoint process associated to it. This form has to be configured as start-up form for a process in Envision. Then, the Launch Process AgileXRM WebPart in Envision will display the process template to users with permissions to open it. When they click on the process template name, the WebPart will open the form.

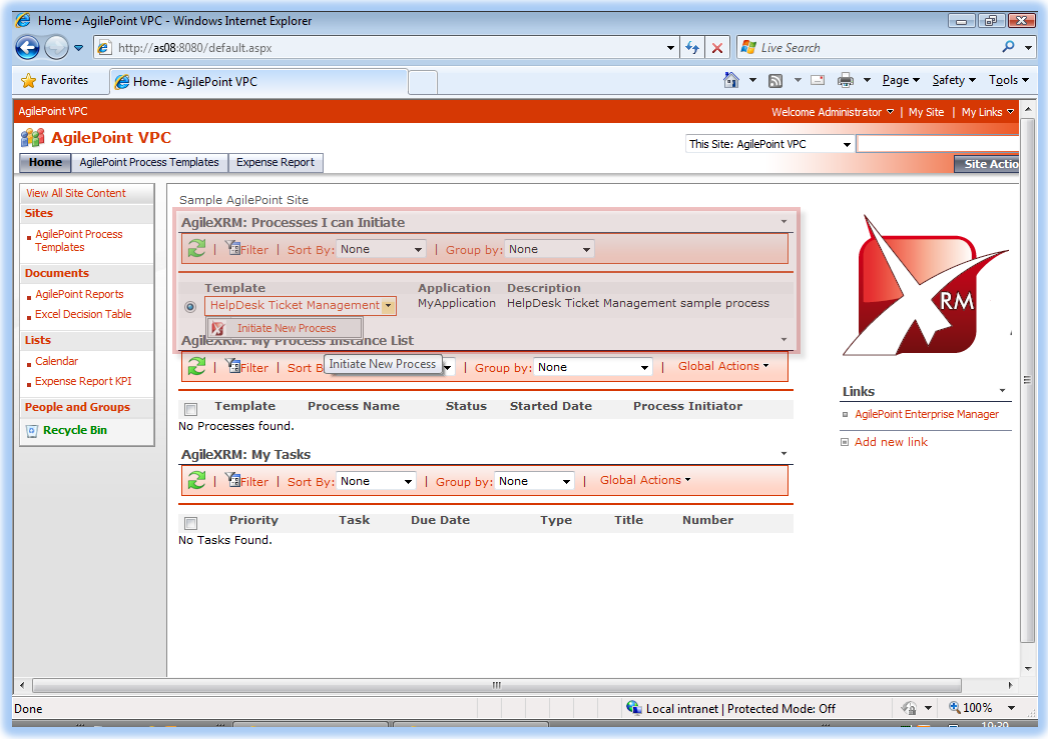


Figure 12 - AgileXRM Launch Process WebPart in a SharePoint site

This form connects to the default CRM and AgilePoint servers. On submit, it launches a new instance of the specified AgileXRM Process, associating it to the new CRM entity created.

4. AgileLightForms Designer Tool Bar

The top part of the AgileLightForms Designer is the tool bar. It contains some controls useful for form designing process:

Figure 13 - AgileLightForms Designer Tool Bar

4.1 Language Combo

Forms created by AgileLightForms designer are multilingual forms, i.e., they can be used by users with different languages. For this to be possible, all form texts have to be entered once for every language of target users. For every text you type (label, or literal used to initialize an attribute at form start-up), this control tells in which language is that text written.

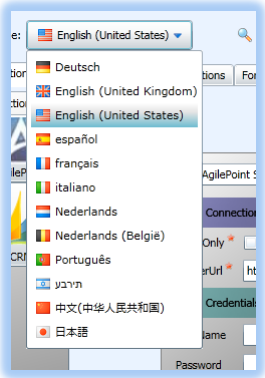


Figure 14 - Language Combo

To translate a form just change the language in the Language Combo and write again all labels and literals.

During run-time, when a user opens a form, AgileLightForms selects the most appropriate language for that user, depending on the available languages in form template.

For example, let's suppose that we have the following labels for a control in our form:

- English: Center
- UK English: Centre
- Spanish: Centro

Users from the United Kingdom will see "Centre", while users from the rest of English Speaking countries will see "Center", and Spanish users (from any Spanish speaking country) will see "Centro".

ALF will first try to find a label for the language and culture of the form user. If it fails to find it, it will search for a generic label for the language of the form user (ignoring the culture). If it doesn't find it, then it will use the first label it finds for the same language, even if it is for a different culture. If it is not able to find a label in the language of the form user, the first label found will be used.

For example, if a user from Australia opens a form, ALF will first look for labels for Australian English. If it doesn't find them, it will look for labels for Generic English. If it doesn't find them, it will use any label for English in other countries (American, British…). If it still doesn't find any, it will use the first label it finds, that won't be English at all.

The list of languages available on the list is configured by an administrator in the ApFormsServerConfig.xml file in the App_Data folder in AgileLightForms Server site. Every language option in this file has the following format:

```
<DataOption>
 <Value xsi:type="xsd:string">en-US</Value>
 <ImageUrl>/Images/16x16/flag_usa.png</ImageUrl>
 <Labels>
  <Labels>
   <item key="en-US">
    <string>English (United States)</string>
   </item>
  </Labels>
 </Labels>
</DataOption>
```

The text inside Value element is the .NET code for the language and / or culture. For a complete list of available languages visit http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx.
The text inside the ImageUrl element is the path for the language icon. Allowed formats are .png, .gif, and .jpg. If the image is outside of AgileLightForms Server site, take into account that AgileLightForms Server will navigate to it using the credentials of the account of the AgileLightForms Server Application Pool.
The Labels element is optional. If omitted, ALF will display the native name for the language and culture specified in the Value element.
If specified, it contains a list of labels for the language, each one in a different Language and or culture. Each label must be in its own Labels element and, inside it, there must be an item element whose key attributes indicates the language and culture of the label. The label must be in a string element inside that one.
Hint: When importing entity forms or attribute controls from CRM, AgileLightForms imports labels in all languages available in the CRM Server from which forms or controls are being imported. So, CRM Server in development environment should have all target user Language Packs to avoid translation efforts.

4.2 Export Form… Button

This button allows you to export the form design as a Visual Studio project, which can be opened with Visual Studio, Expression Blend, or other XAML editors.
Refer to “AgileXRM AgileLightForms Programmer's Guide” for more information about exporting forms.

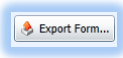


Figure 15 – Export Form… Button

4.3 Import Form… Button

This button allows you it import the form design from a previously exported Visual Studio project, after having edited it with Visual Studio, Expression Blend, or other XAML editors.
Refer to “AgileXRM AgileLightForms Programmer's Guide” for more information about importing forms.

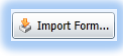


Figure 16 – Import Form… Button

4.4 Upload Handler… Button

This button allows you to upload a form handler, an assembly with code that will be executed on certain form events, such as loading, submitting or attribute changing.
Refer to "AgileXRM AgileLightForms Programmer's Guide" for more information about form handlers.

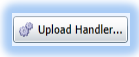


Figure 17 – Upload Handler… Button

4.5 Remove Handler Button

This button allows you to remove the previously uploaded handler from a form.
Refer to "AgileXRM AgileLightForms Programmer's Guide" for more information about form handlers.

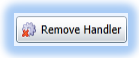


Figure 18 – Remove Handler Button

4.6 Full-screen Button

This button allows you to enter or leave the full-screen mode. This mode is useful to maximize the design area.

Figure 19 - Full-screen Button

Pressing Ctrl + F11 is similar to clicking this button. Esc allows you to exit full-screen mode.
Note: Due to security reasons, Silverlight applications aren't allowed to receive keyboard input when in full-screen mode. So, you can only use the mouse in this mode.

4.7 Clear Cache Button

For performance reasons, AgileLightForms Server does not access data sources (CRM, AgilePoint…) whenever it needs some data from them. Instead, it caches the response it receives for one request and uses it for similar requests.

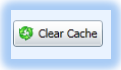


Figure 20 - Clear Cache Button

Cached values are disposed of after they have not been used for a certain amount of time, called cache time. An administrator can configure the cache time in the ApFormsServerConfig.xml file in the App_Data folder in AgileLightForms Server site. The default cache time (<CacheTime>**01:00:00**</CacheTime>) is one hour.
This behavior dramatically improves performance when, for example, accessing metadata, but has some drawbacks if the cached data has been changed in its data source and we still see the old (stale) data. In cases in which we have recently made a change to a data source (like, for instance, creating a new CRM entity, or adding a new attribute to an existing entity) and we don't see that changes in ALF Designer, clearing the cache can solve the problem.
Keep in mind that clearing the cache will not only affect the performance of the user that clears the cache, but of all users of ALF server. Also, it will not only affect the performance of ALF, but any other applications that might share the ALF application pool would be affected too. That is why it is recommended to host ALF Server in its own Application Pool.

5. Connection Editor

Connection Editor is the first tab of ALF Designer, and allows you to manage connections to data sources.

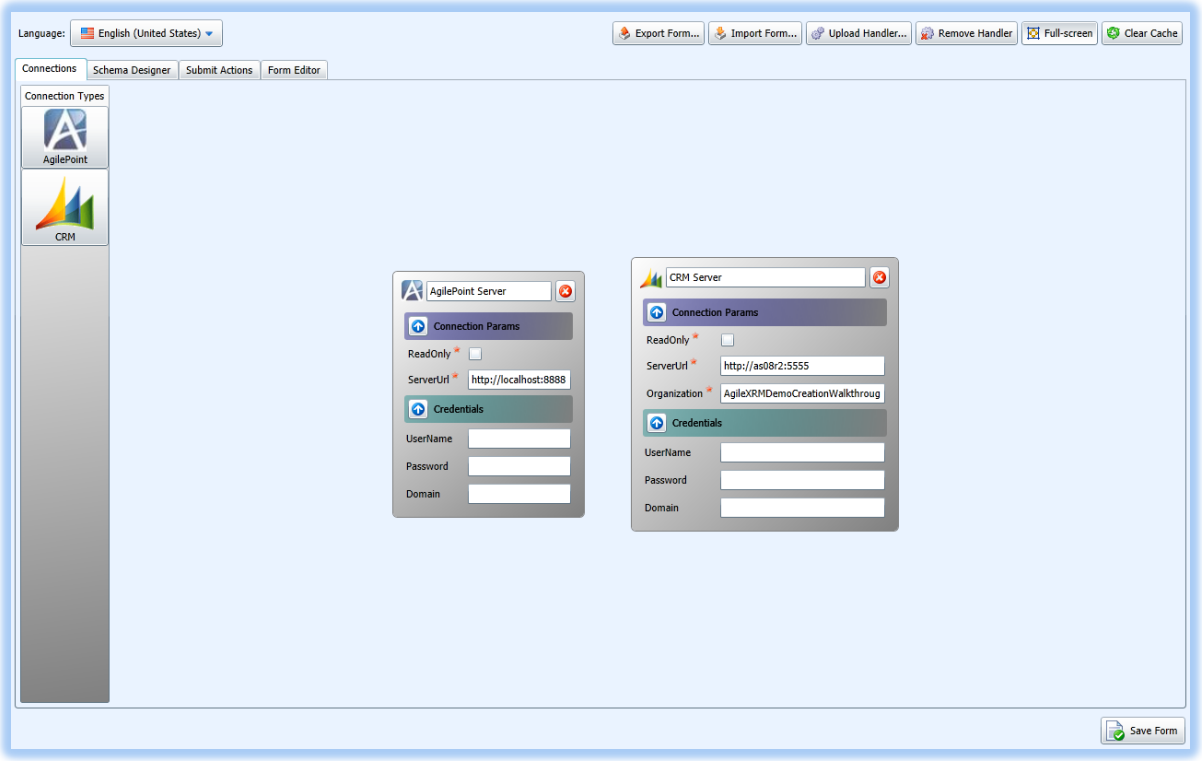


Figure 21 - Connection Editor

Connection Editor allows you to define which connections will be made from this form. Every connection points to a Data Source. There are two types of connections available, AgilePoint Server connections and CRM Organization connections:

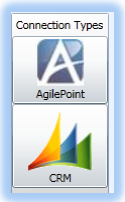


Figure 22 - Connection Types

Usually, all needed connections are added on form creation after selecting the appropriate template. Connections to default AgilePoint and CRM servers without credentials are usually added.
Adding a connection doesn't mean it will be opened when loading the form. Connections are opened on demand, only when they are needed.
To add a new connection, just click on the connection type on the toolbar on the left.

5.1 Entity Editor

Every Connection is depicted on the surface area using a control called Entity Editor.
To delete a connection, just click on the delete ( ) button on the top right corner of the control.
When you add a connection, it is given a default name based on its type. You can rename it using the name box between the connection icon and the delete button. Connection names must be unique. An

error will be thrown if you try to give the same name to more than one connection.
The rest of the Entity Editor control consists of Attribute Editors. Each attribute editor has an attribute name on the left, and an attribute value control on the right.
Attribute names might be followed by a requirement indication icon:

- **Required ( ):** Means you must provide a value for this attribute
- **Recommended ( ):** Means you should provide a value for this attribute, but it is not mandatory
- **Read-only ( ):** Means you can't change the value of this attribute
- The absence of icon means that the attribute is optional

Some attributes are grouped. You can click on the Hide ( ) button on the left side of the group header to hide them all. To view them again, just click on the Show ( ) button on the left side of the group header.

5.2 Entity Validation

Whenever you leave an attribute, an attribute group, or an entity, it is validated. That could result in issues being displayed on the right side of the attribute, the attribute group header, or the whole entity. There are two kinds of issues:

- **Warning ( ):** Means that something is wrong, but the form will work anyway
- **Error ( ):** Means that something must be corrected for the form to work

Hovering the mouse cursor over the issue icon will display a message to clarify what is wrong:

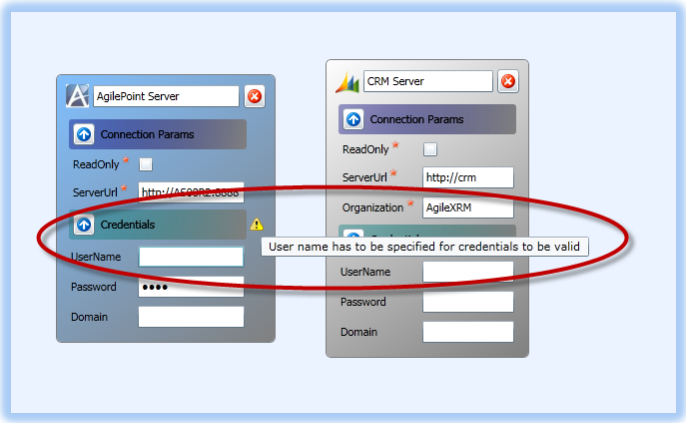


Figure 23 - Validation Issue (Warning) on AgilePoint Server Credentials

5.3 AgilePoint Connection

Indicates a connection to an AgilePoint server. Usually, the default one.

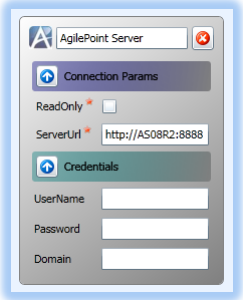


Figure 24 - AgilePoint Connection

You must provide the AgilePoint Server URL. Usually, only the Server address is specified but, if the service hasn't been installed in the default folder (agilepointserver/workflow.asmx), you must specify the full URL to the service.

If you mark the connection as read-only, the form won't be able to change the process instance data (custom attributes).

Credentials are optional. For a full explanation of credentials usage, see document "AgileXRM Reference Architecture". In any case, for credentials to be used, the UserName must be specified.

5.4 CRM Connection

Indicates a connection to a single organization in a CRM server. Usually, the default one.

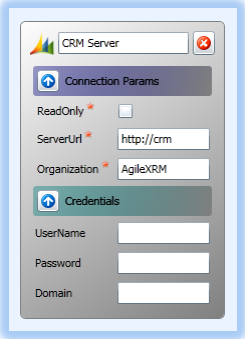


Figure 25 - CRM Connection

You must provide the CRM Server URL and the Organization name.

If you mark the connection as read-only, the form won't be able to create new entities or edit existing ones.

Credentials are optional. For a full explanation of credentials usage, see document "AgileXRM Reference Architecture". In any case, for credentials to be used, the UserName must be specified.

6. Schema Designer

Schema Designer is the second tab of ALF Designer, and allows you to manage the entities that will be retrieved from / updated in / inserted into data sources.
It uses the same paradigm of entity types on the tool bar on the left and entity editors on the design area.

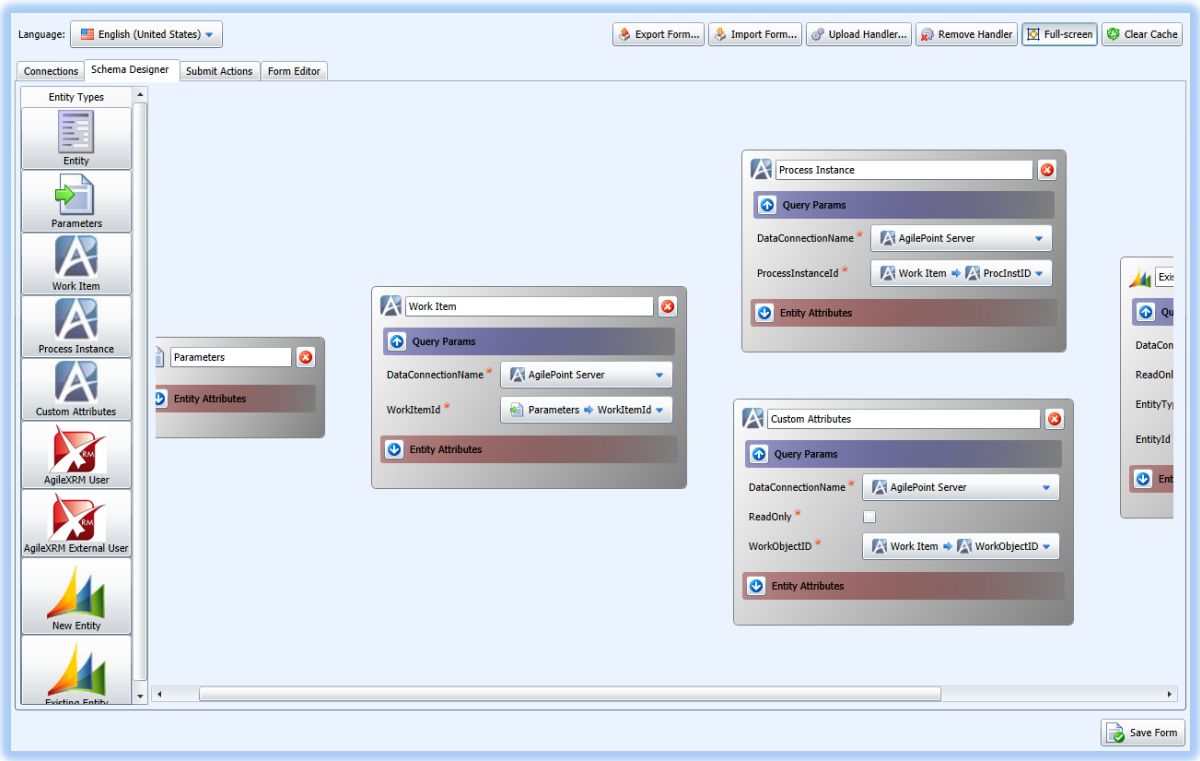


Figure 26 - Schema Designer

6.1 Result Metadata

Unlike connections, entities in Entity Editor use the attributes you provide here to retrieve data (other attributes) from existing entities, or provide data to create new entities.
These attributes depend on the metadata retrieved from the connection the entity is associated to.
Once the entity attributes are valid, you can see the resulting attributes by opening the "Entity Attributes" attribute group, by clicking the Show ( ) button on the left side of the group header.

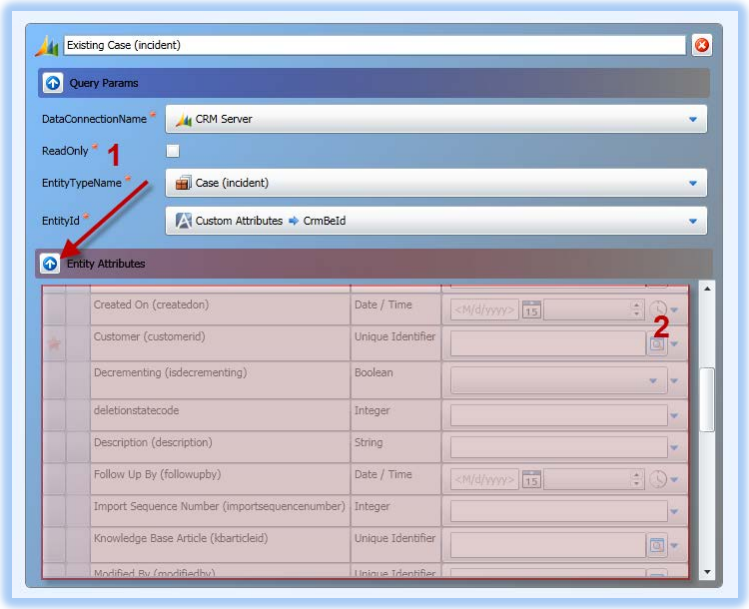


Figure 27 - Result Metadata for an Existing CRM Case

Some entities allow you to edit the name and type of some attributes, while others let you create new attributes and delete them if you think they are not needed.

6.1.1 Result Metadata Attributes

Every attribute in the result metadata portion of Entity Editor is composed of the following controls:

Delete Button ( )

This button appears only in attributes that can be deleted. If you click on it, the attribute will be deleted, and won't be available for the user interface.

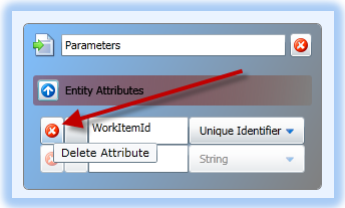


Figure 28 - Delete Attribute Button

Required Button

This button contains an icon that indicates the requirement constraints for the attribute. You can only change the requirement constraint of some attributes.

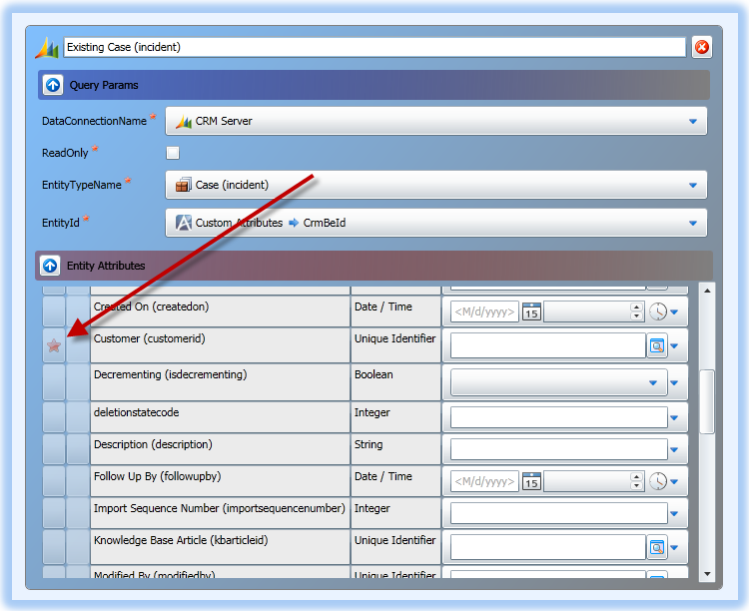


Figure 29 - Read-only Required Button in non-editable attribute

This is a toggle button that toggles between the following values:

- **Required ( ):** Means the final user must provide a value for this attribute. This option is available for non-read-only entities only
- **Recommended ( ):** Means the final user should provide a value for this attribute, but it is not mandatory. This option is available for non-read-only entities only
- **Read-only ( ):** Means the final user can't change the value of this attribute. This option is available for non-read-only entities only
- **Hidden ( ):** Means the final user can't see the value of this attribute
- The absence of icon means that the attribute is visible and optional for the final user

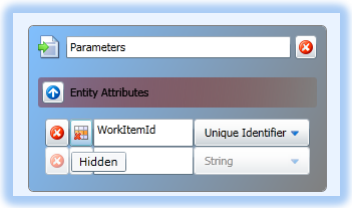


Figure 30 - Required Button in editable attribute

Password Button

Password Button is only available in String attributes in some entities. It allows you to mark an attribute as a password, so it will appear on a password box (with a repeating symbol instead of real chars) in the user interface.

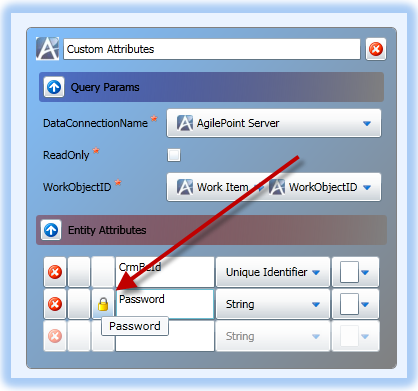


Figure 31 - Password Button

This is a toggle button that toggles between the following values:

- **Password ( ):** Means the attribute is a password and its characters will be displayed as a repeating symbol
- The absence of icon means that the attribute value is not a password

Attribute Name

It indicates the attribute name. You can only edit the name of some attributes in some entities. Attribute names must be unique in their entities.

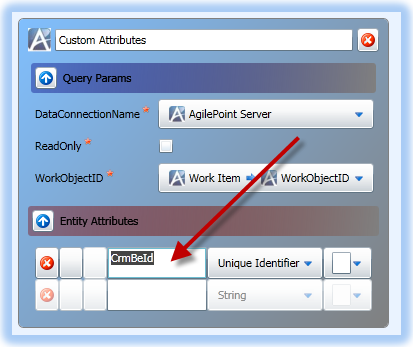


Figure 32 - Editable Attribute Name Box

Attribute Type

It indicates the type of the value of the attribute. You can only edit the type of some attributes in some entities.

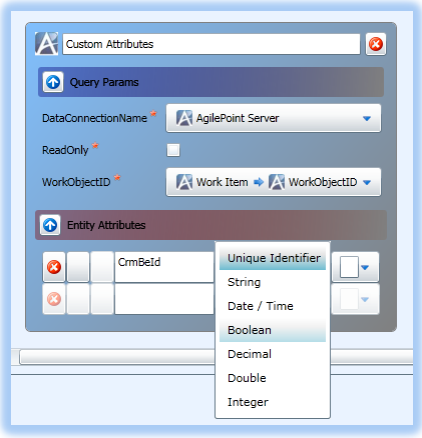


Figure 33 - Editable Attribute Type Box with dropped-down list of available types

Default Value

You can provide a default value for non-read-only attributes. This is very useful to provide some fields to your end user, like his name, his company, etc.
You can either provide a fixed value or bind the default value of the attribute to the value of other attribute in the schema.
For now, we'll see only how to provide constant default values. The other option will be described later (data binding).
To provide a constant default value, just forget about the drop icon ( ) on the right and enter the value in the control on the left. The type of control depends on the type of the attribute.

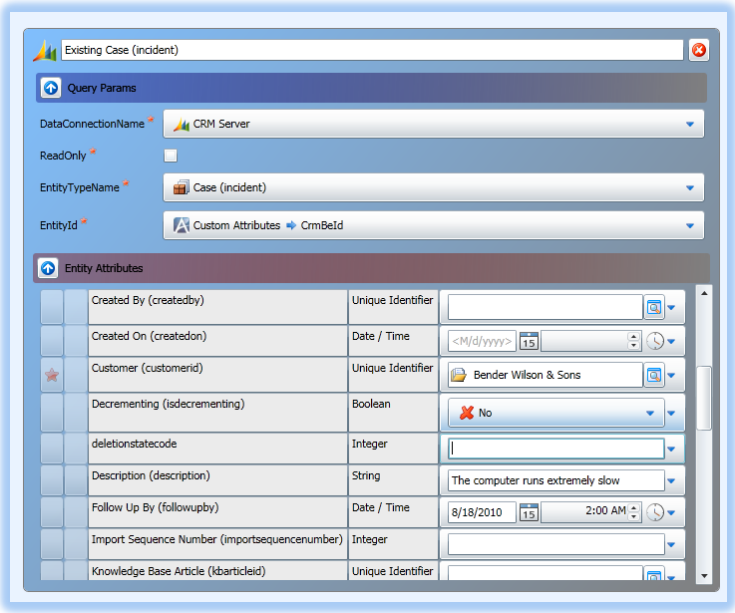


Figure 34 - Constant Default Values of several types

6.2 Data Binding

Data binding is a very powerful tool. It consists in using the value of a result attribute retrieved by an entity as a parameter attribute to retrieve other. For example, when you retrieve a Work Item, you get the result attribute ProcInstId. You can see it if you open the Entity Attributes group of a work Item entity. Then, you can bind the parameter ProcessInstanceId of a Process Instance entity to it, by selecting it in the combo box. The effect of this data binding is using a value retrieved in one entity (in this case, the value ProcInstId of Work Item) to retrieve a different entity (in this case, Process Instance data).

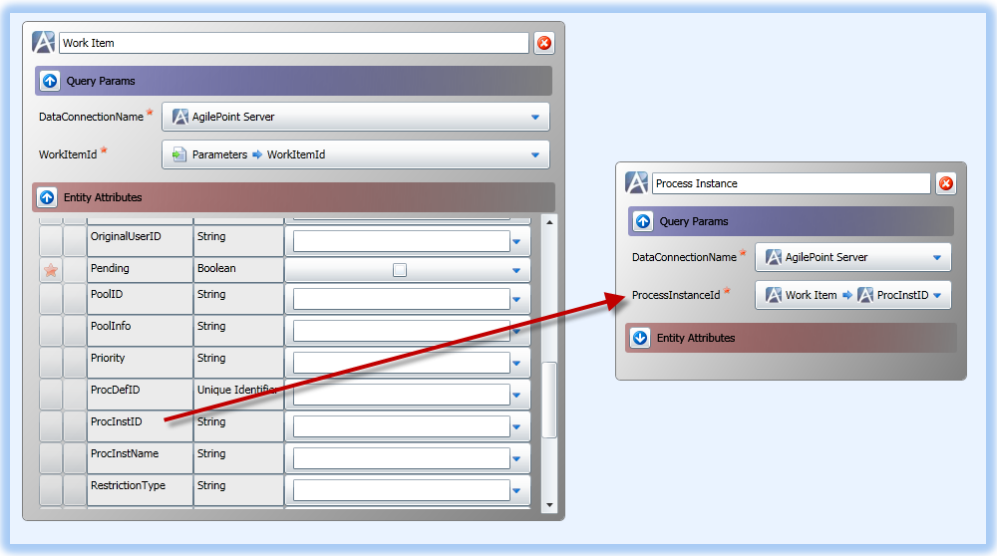


Figure 35 - Data Binding

This makes the latter entity dependent on the former one. While in Connection Editor the layout of connections on the design surface is meaningless, in Schema Editor entities are laid out in a way that makes dependent entities appear on the right side of the entities they depend on. This avoids circular references, i.e., entity A depending of an attribute of entity B and entity B depending on an attribute of entity A. That is not possible because then entity A would have to be retrieved to retrieve entity B, but entity B would require entity A to be retrieved first.

Other use for data binding is using the value of an attribute as the default value of other attribute. With this type of binding, take into account that it is the initial value of the source attribute the one that is being copied to the target attribute. If the user changes the value of the source attribute, that change won't be copied to the target attribute.

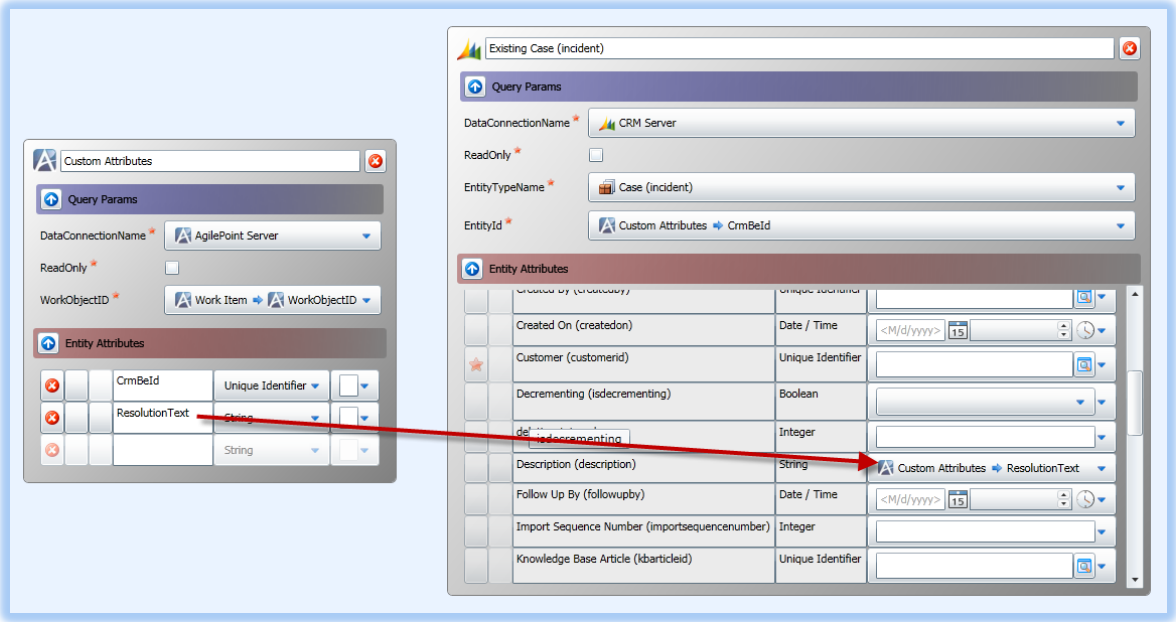


Figure 36 - Data Bound Default Value

To bind an entity parameter or an attribute default value to the value of another attribute, click on the drop-down icon ( ) on the right side. It will drop a tree with all entities with attributes to bind to. Only entities that won't cause a circular reference will appear. If you open an entity, you will only see attributes you can bind to. These are the attributes that can be bound:

- Attributes of the same type
- A string attribute and any other attribute, as long as its value can be cast to/from string
- An array of byte attribute and any other attribute, as long as it is serializable

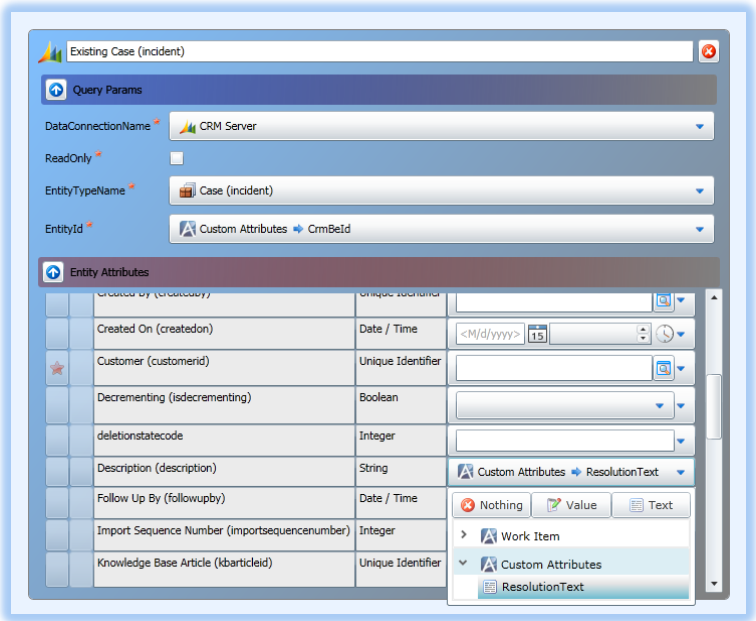
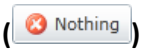
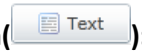
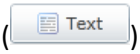


Figure 37 - Data Binding Drop-down Tree

There are three buttons on the top of the drop-down:

- **Nothing Button (**  **):** Removes the current value of the attribute
- **Value Button (**  **):** Removes the current binding of the attribute and allows you to enter a constant value
- **Text Button(**  **):** Allows you to enter a localizable text, as described in next section

6.3 Localizable Texts and Concatenations

If you provide a constant string in the text box for the default value of an attribute, that value will be used for any end-user, no matter what language he speaks. To provide localizable texts, click the Text Button() on the top right corner of the data binding drop down. This mechanism also allows you to concatenate values from several attributes. When you click the button, the Text Editor Window pops up:

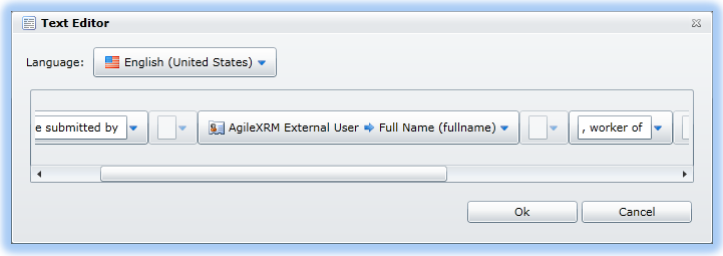


Figure 38 - Text Editor Window

The Text Editor Window has a Language combo similar to that on the ALF designer tool bar, which allows you to introduce different texts for different languages and / or cultures without having to open the same window many times.
Apart from that, a text can be composed of many different string literals and / or data bindings.

6.4 Entity Bar

The following entities can be added to form schema:

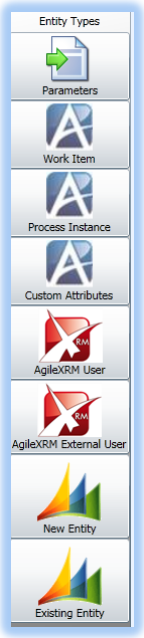


Figure 39 – Entity Bar in Schema Designer

6.4.1 Parameters Entity

This entity represents the parameters passed to the form by the ASP form that hosts it. Each Attribute represents a different parameter.

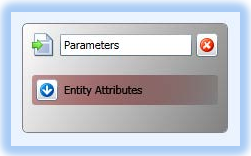


Figure 40 - Parameters Entity

ALF Parameters can be any query string parameter in the URL of the container page, or any parameter passed by container page to Silverlight object.
This entity needs no attributes to be retrieved, and is a read-only entity.

6.4.2 Work Item Entity

Represents a manual activity in an AgilePoint process. It needs two parameters to be retrieved: the connection to AgilePoint (usually the default one) and the Work Item Id (usually data bound to Parameter WorkItemId).

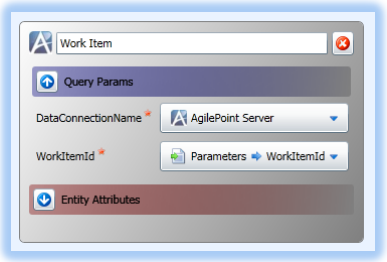


Figure 41 - Work Item Entity

This is a read-only entity.

6.4.3 Process Instance Entity

Represents an instance of an AgilePoint process. It needs two parameters to be retrieved: the connection to AgilePoint (usually the default one) and the Process Instance Id (usually data bound to attribute ProcInstID in a Work Item entity).

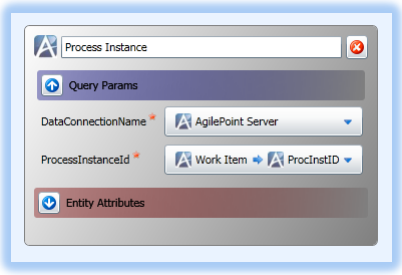


Figure 42 - Process Instance Entity

This is a read-only entity.

6.4.4 Custom Attributes Entity

Represents the custom attributes (context data) of an instance of an AgilePoint process. It needs two parameters to be retrieved: the connection to AgilePoint (usually the default one) and the Work Object

ID (usually data bound to attribute WorkObjectID in a Work Item or Process Instance entity).

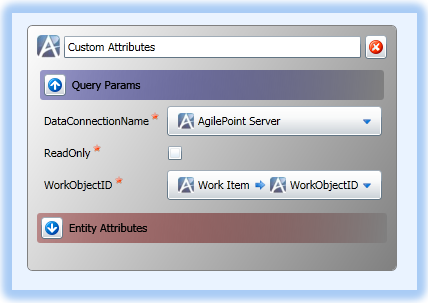


Figure 43 - Custom Attributes Entity

You can add attributes to the metadata. If you give a default value to custom attribute, or let the user edit it, those changes will be made on process instance data when form is submitted by end user.

6.4.5 AgileXRM User Entity

Represents an AgileXRM User. This has to be a named user, not a user that connects through the external connector.

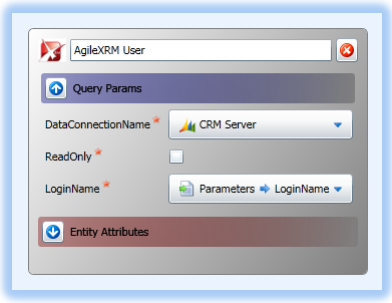


Figure 44 - AgileXRM User Entity

AgileXRM stores this kind of users in CRM's entity User (systemuser). So, this entity is similar to using Existing CRM Entity to retrieve an entity of type User, except that AgileXRM User queries using the Login Name instead of the User Id.

You have to provide the CRM Connection (usually the default one) and the Login Name (usually data bound to parameter LoginName).

6.4.6 AgileXRM External User Entity

Represents an AgileXRM External User. This is a user that connects through the external connector.

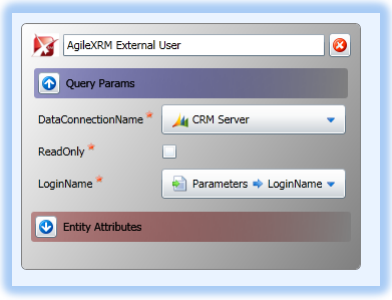


Figure 45 - AgileXRM External User Entity

AgileXRM stores this kind of users in CRM's entity Contact (contact). So, this entity is similar to using Existing CRM Entity to retrieve an entity of type Contact, except that AgileXRM External User queries using the Login Name instead of the Contact Id.

You have to provide the CRM Connection (usually the default one) and the Login Name (usually data bound to parameter LoginName).

6.4.7 New CRM Entity

This represents a New Entity that will be created in CRM when form is submitted by end user.

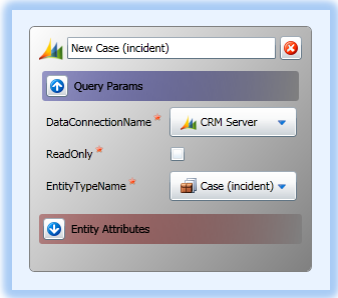


Figure 46 - New CRM Entity

You have to provide the CRM Connection to the organization in which you want the entity to be created (usually the default one) and the type of entity to create. The list of entity types is retrieved from selected connection, and the result attributes are those for that type of entity in the selected CRM Organization. If selected organization is not the production one, it should be an exact copy of it.
If you don't see an entity you have just created, or any other change just made to metadata, see the description of Clean Cache Button on the Designer Toolbar.
The new entity will be initialized with CRM's default values for attributes of that entity.

6.4.8 Existing CRM Entity

Represents an entity that currently exists in CRM. If you modify any attributes, those changes will be updated in CRM when the form is submitted by end user.

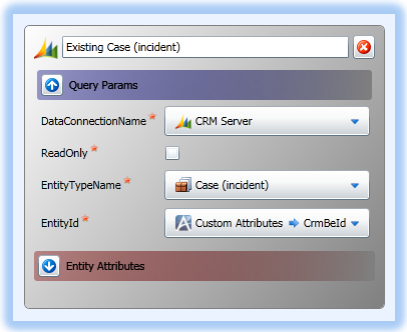


Figure 47 - Existing CRM Entity

You have to provide the CRM Connection to the organization from which you want the entity to be retrieved (usually the default one) and the type of entity to retrieve. The list of entity types is retrieved from selected connection, and the result attributes are those for that type of entity in the selected CRM Organization. If selected organization is not the production one, it should be an exact copy of it.
If you don't see an entity you have just created, or any other change just made to metadata, see the description of Clean Cache Button on the Designer Toolbar.

7. Submit Action Editor

Submit Action Editor is the third tab of ALF Designer, and allows you to manage the actions that will be taken when the form is submitted.

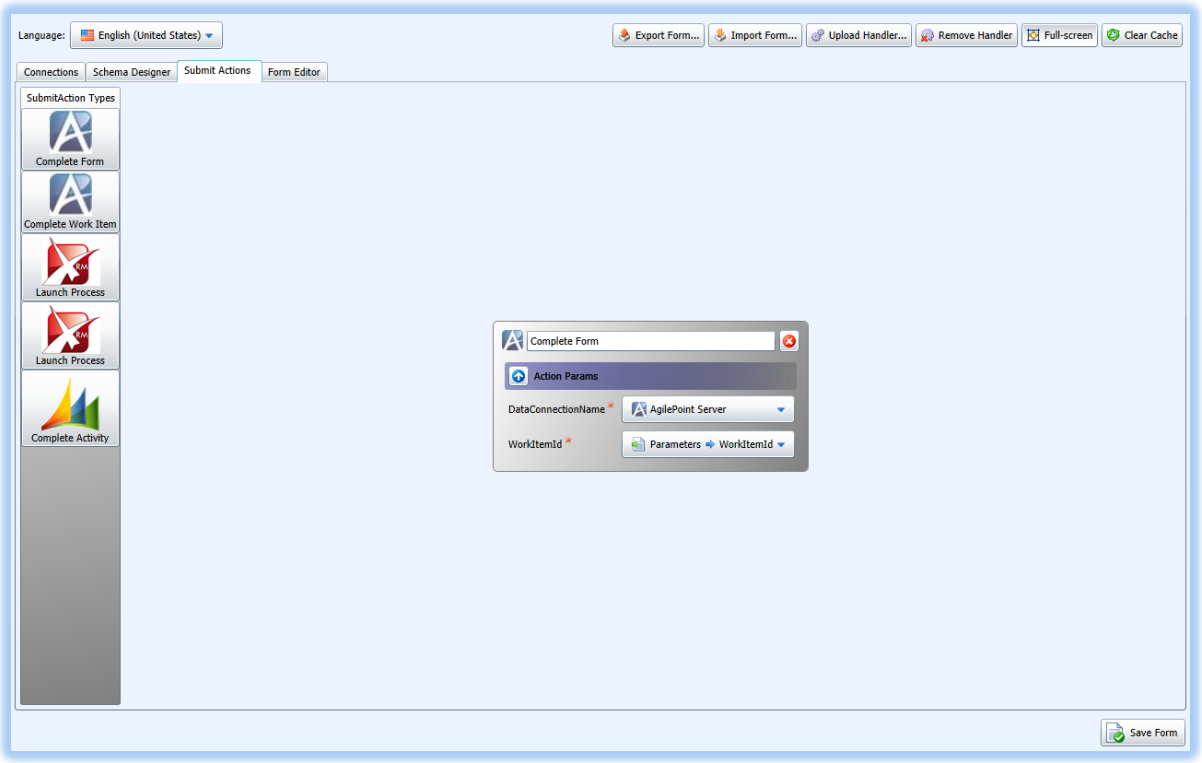


Figure 48 - Submit Action Editor

The first action that is always taken when a form is submitted is to create new entities and / or update existing ones. Then, submit actions are executed in the order they appear on the designer.

Note: Every submit action is executed independently. If a submit action fails, subsequent actions won't be executed, but previous actions have been. So, some actions may have been executed even if an error is thrown.

These are the available Submit Actions

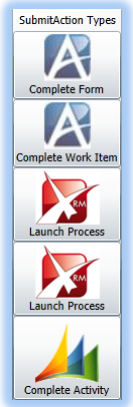


Figure 49 - Available Submit Actions

7.1 Complete Form Submit Action

This action marks the form as completed. Completing form is not the same as completing the manual task (a.k.a. work item).

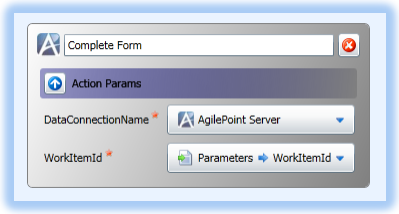


Figure 50 - Complete Form Submit Action

Work Items have a property called Wait Work Performed that doesn't allow the process to advance if the form hasn't been completed.

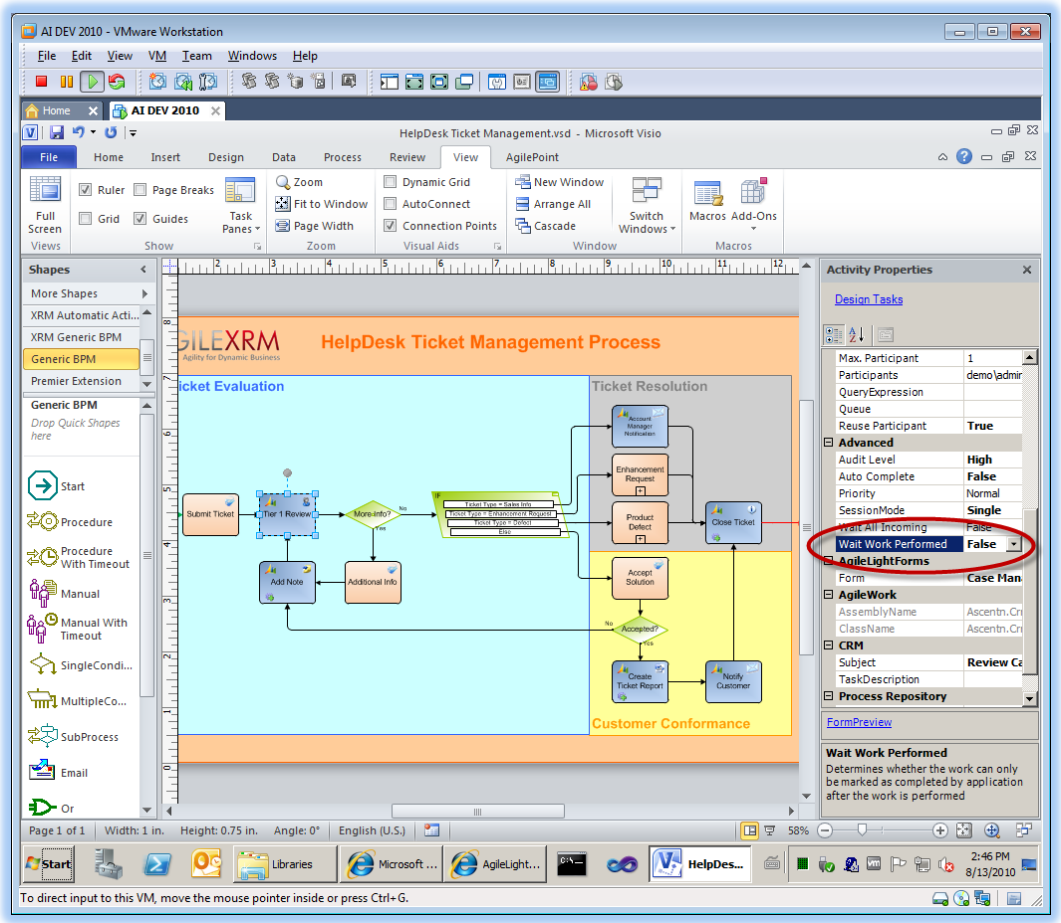


Figure 51 - Wait Work Performed Property for Manual Work Items

For example, CRM Activities can't be completed (by clicking in the Save As Completed button, for example) if they represent a manual task in an AgileXRM process in which Wait Work Performed is True and the form hasn't been completed.

It requires two parameters to run: the connection to AgilePoint (usually the default one) and the Work Item Id (usually data bound to Parameter WorkItemId).

7.2 Complete Work Item Submit Action

This completes an AgilePoint Work Item. If the Work Item is associated to a CRM activity, that activity is completed too.

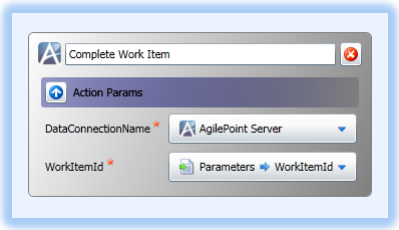


Figure 52 - Complete Work Item Submit Action

It requires two parameters to run: the connection to AgilePoint (usually the default one) and the Work Item Id (usually data bound to Parameter WorkItemId).

7.3 Launch Process 1 Submit Action

This action launches a new instance of an AgileXRM process, and associates it with the first "New CRM Entity" it finds in Form Schema.

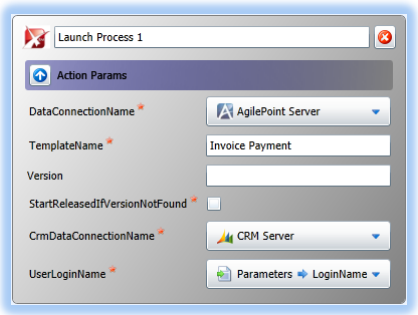


Figure 53 - Launch Process 1 Submit Action

This submit action requires six parameters: The connection to the AgilePoint Server in which the process instance is going to be created (usually the default one), the name of the process template from which to create a new instance, the CRM Organization that hosts the process main entity (usually the default one), the LoginName of the user we want to become the process initiator (usually data bound to LoginName Parameter), and two version-related parameters.

If version is left blank, or *.* is entered, released version of process will be used. If a string with a major version only (like 5.*) is entered, the latest version 5 of process will be used. If a string with both major and minor versions is entered, that specific version will be launched.

If specified version is not found and StartReleasedIfVersionNotFound is checked, Released version of process will be launched.

7.4 Launch Process 2 Submit Action

This action launches a new instance of an AgileXRM process, and associates it with the provided entity.

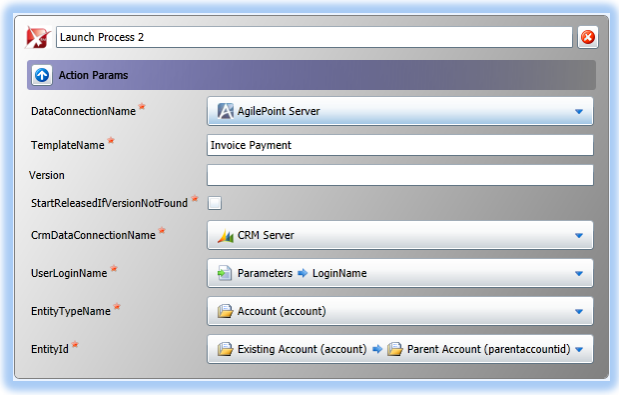


Figure 54 - Launch Process 2 Submit Action

This submit action requires eight parameters: The connection to the AgilePoint Server in which the process instance is going to be created (usually the default one), the name of the process template from which to create a new instance, the CRM Organization that hosts the process main entity (usually the default one), the LoginName of the user we want to become the process initiator (usually data bound to LoginName Parameter), the type and Id of the entity that will become the main entity of the new process, and two version-related parameters.

If version is left blank, or *.* is entered, released version of process will be used. If a string with a major version only (like 5.*) is entered, the latest version 5 of process will be used. If a string with both major and minor versions is entered, that specific version will be launched.

If specified version is not found and StartReleasedIfVersionNotFound is checked, Released version of process will be launched.

7.5 Complete Activity Submit Action

This completes a CRM Activity (Fax, Phone Call, …). If the activity is associated to an AgilePoint Work Item, that Work Item is completed too.

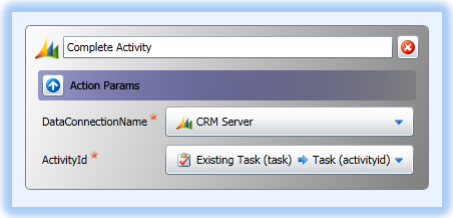


Figure 55 - Complete Activity Submit Action

It requires two parameters to run: the CRM Organization (usually the default one) and the Activity Id (usually data bound to another attribute).

8. Form Editor

Form Editor is the fourth (and last) tab of ALF Designer, and possibly the most important. It allows you to define the interface that will be used to show data to the end user and get data from him / her.

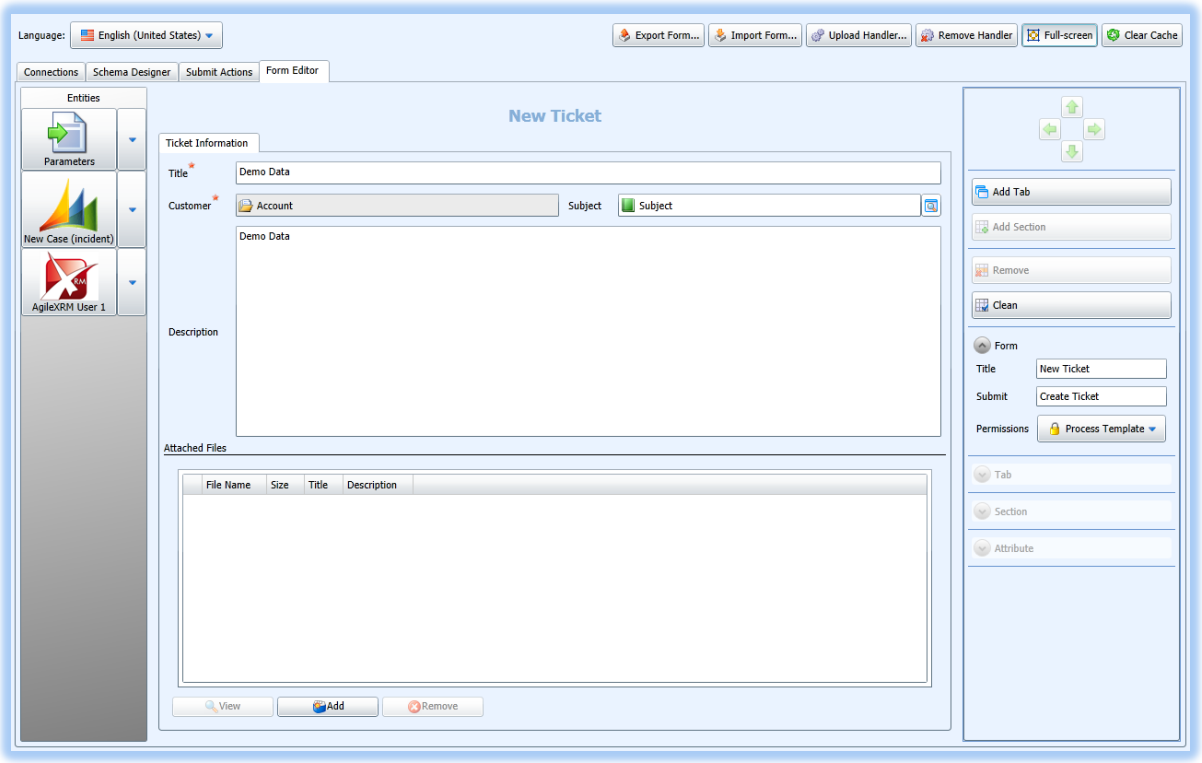


Figure 56 - Form Editor

Form Editor is divided in three zones: Entity Bar on the left, Actions Pane on the right, and design surface in the middle.

8.1 Entity Bar

Entity bar contains all the entities in form schema. The drop-down button on the right of every entity contains all the result attributes from that entity.

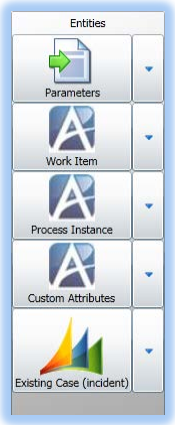


Figure 57 - Form Editor Entity Bar

Clicking on an entity adds the default form for that entity to current form. For CRM entities (either new or existing), the CRM Form is copied from CRM. For other entities, a default form is created.

Clicking on an attribute in the drop-down list on the right of an entity adds a control for that attribute only. If that attribute belongs to a CRM Entity (either new or existing), and the attribute is on the CRM Form, the control is copied from CRM. For other attributes, a default control is created.

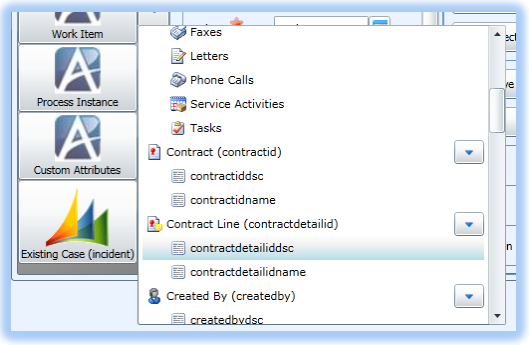


Figure 58 - Drop-down list of Entity Attributes

Apart from single attributes, the drop-down list can add a control to manage collections of child entities.

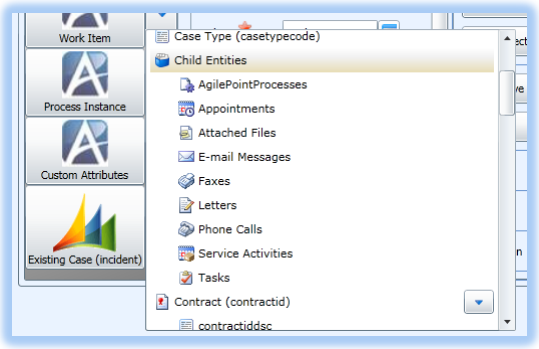


Figure 59 - Child Entities in Entity Drop-down

Clicking on a child entity adds a Children View control. This control allows end user to add, delete, view, update and search for child entities of the clicked type. Form designer can decide which of these options will be available for the end user. Children View control properties are described later.

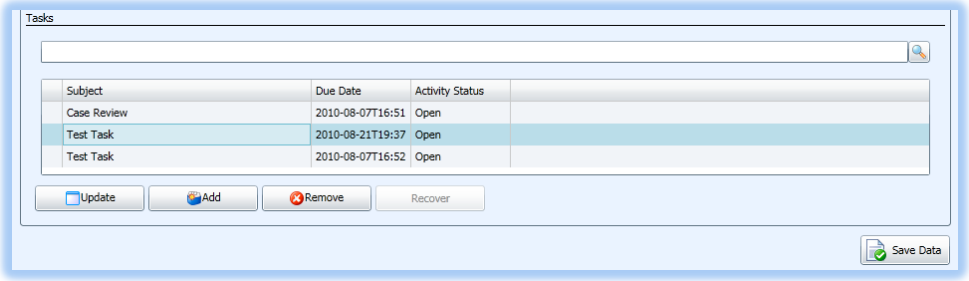


Figure 60 - Children View Control

A special case is when you click on “Attached Files” child entity. In this case, the Attached Files control is added. Attached Files control properties are described later.

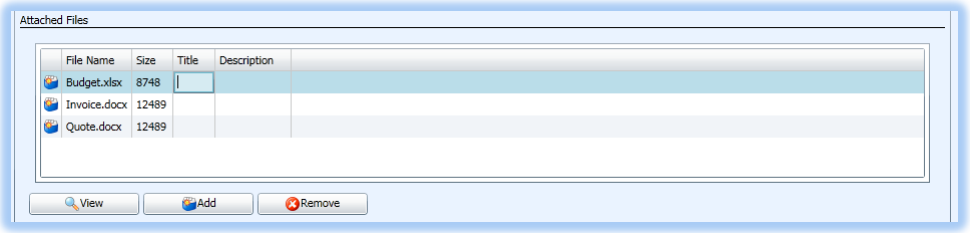


Figure 61 - Attached Files Control

8.2 Design Surface

AgileLightForms Designer Design Surface is where you see a WYSIWYG layout of the form the end user will see.

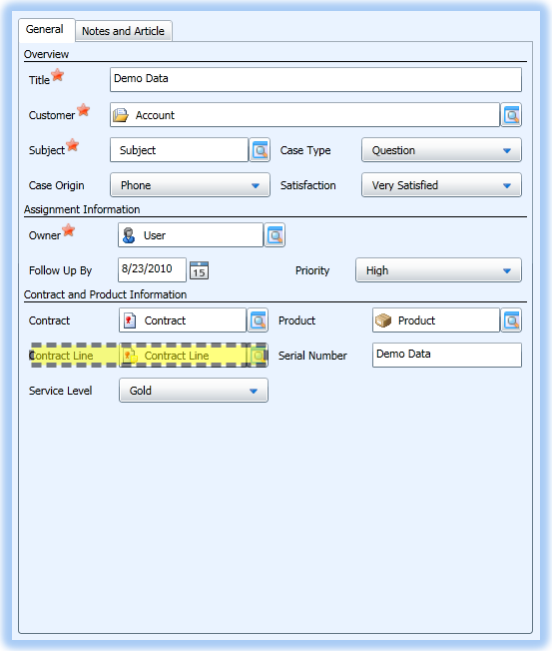


Figure 62 - Form Editor Design Surface with a cell selected

AgileLightForms have a layout similar to CRM Forms. Every Form is composed of one or more tabs, each one containing one or more sections. Every section has a number of columns in which cells are laid out in rows. Every cell is composed of a control that can be preceded by a label.

A full reference of properties of form, tabs, sections and controls can be found later.

Apart from letting you see what the end user will see, the design surface also allows you to select tabs, sections and cells.

To select a cell, just click on either the cell control or label. A yellow highlight box indicates the currently selected cell. Selecting a cell automatically selects its container section and tab, but no visual indication of that is displayed.

To select a section, just click on the section space between or around controls. A blue highlight box indicates the currently selected section. Selecting a section automatically selects its container tab, but no visual indication of that is displayed.

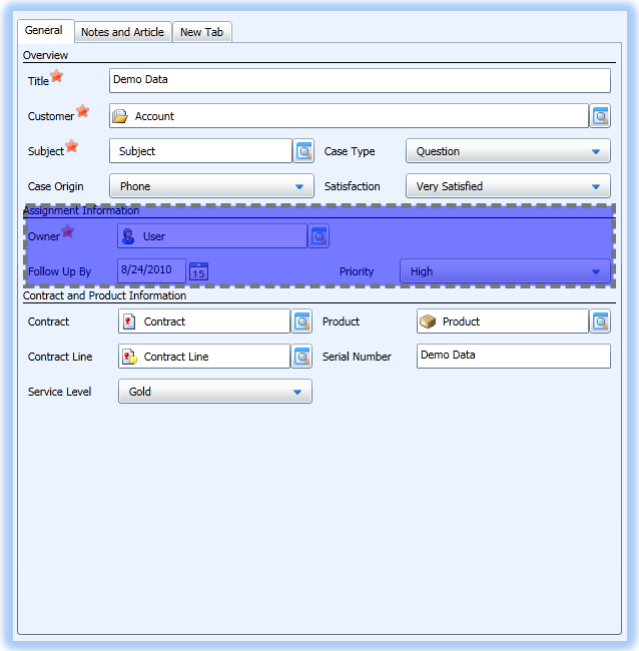


Figure 63 - Design Surface with a section selected

To select a tab, just click on the tab header on the top of the design surface. A red highlight box indicates the currently selected tab.

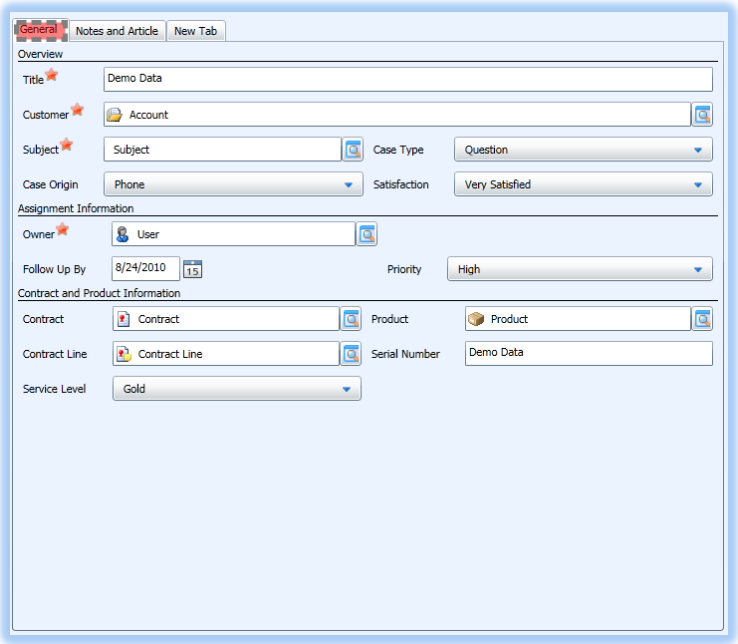


Figure 64 - Design Surface with a tab selected

If you want to unselect everything just click on the empty space at the right of tab labels.

8.3 Actions Pane

This pane on the right side of Form Editor contains many useful tools.

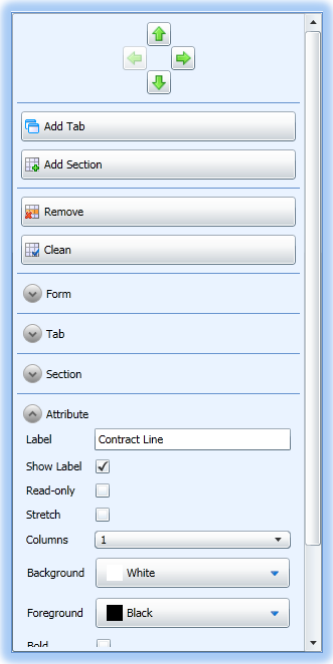


Figure 65 - Form Editor Actions Pane

8.3.1 Movement Pane

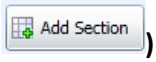
The movement pane allows you to move the Tab, Section, or Cell selected on the design surface.

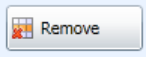
Figure 66 - Movement Pane

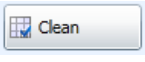
8.3.2 Command Pane

The command Pane contains some useful buttons:

- **Add Tab Button (** 

): Adds a new tab to the form
- **Add Section Button (** 

): Adds a new section to the selected tab
- **Remove Button (** 

): Removes the selected tab, section or cell
- **Clean Button (** 

): Cleans empty space on the form

8.3.3 Form Properties

These are the properties that you can configure for your form:

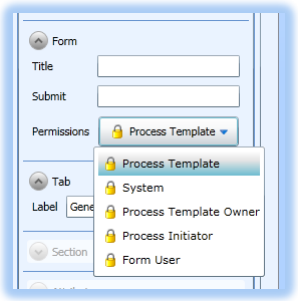


Figure 67 - Form Properties

- **Title:** This is a localizable text that will be used as Form Title.
  Tip: Use the language combo on the tool bar to enter the label in different languages and / or cultures.
- **Submit:** This is a localizable text that will be used as the caption for the Submit button. If you leave it empty, “Save Data” will be used.
  Tip: Use the language combo on the tool bar to enter the label in different languages and / or cultures.
- **Permissions:** Allows you to specify under which permissions will the form run. For a complete explanation, see the document “AgileXRM Reference Architecture”.

8.3.4 Tab Properties

There’s only one property that can be configured for a tab: its label.

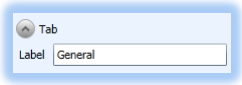


Figure 68 - Tab Properties

Tip: Use the language combo on the tool bar to enter the label in different languages and / or cultures.

8.3.5 Section Properties

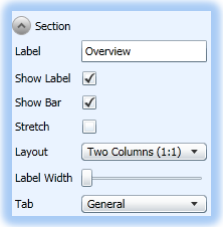


Figure 69 - Section Properties

- **Label:** Allows you to introduce the section label.

  Tip: Use the language combo on the tool bar to enter the label in different languages and / or cultures.
- **Show Label:** Indicates whether section label will appear on the form.
- **Show Bar:** Indicates whether a bar will appear below the section label on the form.
- **Stretch:** Indicates whether the section will stretch vertically to occupy as much space as possible. All spare vertical space will be shared between all sections with attribute Stretch set.
- **Layout:** Lets you choose between some layouts. The numbers in brackets indicate the relative widths of the columns.

- **Label Width:** Lets you choose a fixed with for the labels in the section or, if you leave the slider at its left most side, ALF will choose the minimum to accommodate the labels in every language.
- **Tab:** Allows you to select a different tab to move the section to.

8.3.6 Attribute Properties

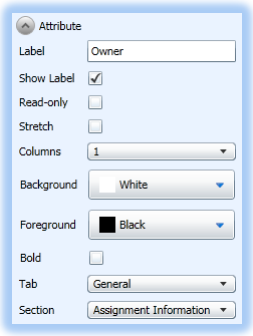


Figure 70 - Attribute Properties

- **Label:** Allows you to introduce the attribute label.

  Tip: Use the language combo on the tool bar to enter the label in different languages and / or cultures.
- **Show Label:** Indicates whether attribute label will be visible.
- **Read-only:** Indicates whether attribute will be read-only, and so editing will not be permitted on the control.
- **Stretch:** Indicates whether the control will stretch vertically to occupy as much space as possible. All spare vertical space in the section will be shared between all controls with attribute Stretch set. When you set Stretch on a control, it is automatically set in the container section as well.
- **Columns:** Indicates how many columns will be used by the cell. The possible number depends on the chosen section layout.
- **Background:** Indicates the background color of the control. Not all controls can adjust their background
- **Foreground:** Indicates the foreground color of the control. Not all controls can adjust their foreground
- **Bold:** Indicates whether control text will be bold. Not all controls can make their text bold
- **Tab:** Allows you to select a different tab to move the attribute to. No control is really moved until you select a target section.
- **Section:** Selects which section in the previous tab will be the container for the cell.

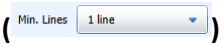
8.3.7 Control Properties

Some types of controls have additional properties, as shown below.

TextBox Properties

Textbox controls have two additional properties:

- **Mode (** Mode Text **):** to specify whether you want a Button that will navigate to the URL specified by attribute value instead of a text box to edit the attribute value.
- **Multiline (** Multiline **):** to specify whether the control will accept several lines. If so, both the control and the section will stretch.

- **Min. Lines (**  **):** Indicates the minimum number of lines the control will stretch

Lookup Properties

Lookup controls have some additional properties:

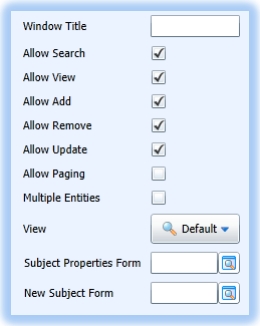


Figure 71 - Lookup Properties

- **Window Title (**  **):** to specify the title of the lookup search window. If you leave it blank, the name of the lookup attribute will be used.

  Tip: Remember you can localize this text using the Language combo.
- **Allow Search:** Check it to allow the end-user to search for child entities containing a text.
- **Allow View:** Indicates whether View button will be available. If control is not read-only and Allow Update is also set, Update button will appear instead of View button. View button will never show if Existing Entities Form hasn't been specified.
- **Allow Add:** Indicates whether Add button will be available. Add button won't appear on read-only entities, or if New Form Template hasn't been specified.
- **Allow Remove:** Indicates whether the Remove / Recover buttons will appear. Recover button appears when you select an already removed entity. None of these buttons will appear on read-only entities.
- **Allow Update:** Indicates whether the Update button will be available. If either control or entity are read-only, View button will appear instead. Update button will never show if Existing Entities Form hasn't been specified.
- **Allow Paging:** Indicates whether paging will be enabled, and so entities will be retrieved on a page-by-page basis.
- **Multiple Entities:** For party lists, indicates whether user will be allowed to select more than one entity. Some party lists, such as sender fields, can point to a single party.
- **View (**  **):** To specify the view to use to let user choose entities from. If no view is specified, the Default Lookup View is used.
- **<EntityType> Properties Form:** Lets you choose the form that will appear when the user tries to view or update a child entity of that type. This form should be created using the template "Form to View or Update Existing Child Entities". A different form must be specified for every possible entity type.

  If you don't specify a value, the View / Update button won't appear on the control when an entity of that type is selected.

- **New <EntityType> Form:** Lets you choose the form that will appear when the user tries to create a new child entity of that type. This form should be created using the template "Form to Create New Child Entities". A different form must be specified for every possible entity type.
  If you don't specify a value, the New button won't appear when you select the entity type with the missing form.

Children View Properties

Children View control provides the following extra properties you can configure.

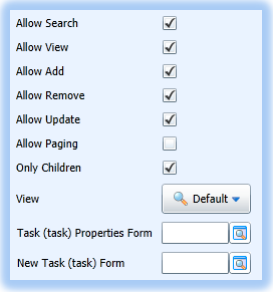


Figure 72 - Children View Properties

- **Allow Search:** Check it to allow the end-user to search for child entities containing a text.
- **Allow View:** Indicates whether View button will be available. If control is not read-only and Allow Update is also set, Update button will appear instead of View button. View button will never show if Existing Entities Form hasn't been specified.
- **Allow Add:** Indicates whether Add button will be available. Add button won't appear on read-only entities, or if New Form Template hasn't been specified.
- **Allow Remove:** Indicates whether the Remove / Recover buttons will appear. Recover button appears when you select an already removed entity. None of these buttons will appear on read-only entities.
- **Allow Update:** Indicates whether the Update button will be available. If either control or entity are read-only, View button will appear instead. Update button will never show if Existing Entities Form hasn't been specified.
- **Allow Paging:** Indicates whether paging will be enabled, and so entities will be retrieved on a page-by-page basis.
- **Multiple Entities:** For party lists, indicates whether user will be allowed to select more than one entity. Some party lists, such as sender fields, can point to a single party.
- **<EntityType> Properties Form:** Lets you choose the form that will appear when the user tries to view or update a child entity of that type. This form should be created using the template "Form to View or Update Existing Child Entities". A different form must be specified for every possible entity type.

  If you don't specify a value, the View / Update button won't appear on the control when an entity of that type is selected.
- **New <EntityType> Form:** Lets you choose the form that will appear when the user tries to create a new child entity of that type. This form should be created using the template "Form to Create New Child Entities". A different form must be specified for every possible entity type.
  If you don't specify a value, the New button won't appear when you select the entity type with the missing form.

Attachments View Properties

Attachments View control provides the following extra properties you can configure.

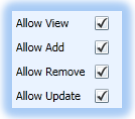


Figure 73 - Attachments View Properties

- **Allow View:** Indicates whether View button will appear and let end-user view the contents of attached files already existing on entity.
- **Allow Add:** Indicates whether Add button will appear and let end-user add new attached files. Of course, neither control nor entity can be read-only.
- **Allow Remove:** Indicates whether Remove button will appear and let end-user remove already existing attached files. Of course, neither control nor entity can be read-only.
- **Allow Update:** Indicates whether Update button will appear and let end-user add attached files with the name of an already existing one, updating them. Of course, neither control nor entity can be read-only.