



# **FlexiLayout Studio 9.0**

## Tutorial Sample 2

## Contents

<b>Introduction.....</b>	<b>3</b>
<b>Step 1. Creating a new project.....</b>	<b>3</b>
<b>Step 2. Adding images to the batch .....</b>	<b>4</b>
<b>Step 3. Setting the FlexiLayout properties.....</b>	<b>7</b>
<b>Step 4. Pre–recognition.....</b>	<b>8</b>
<b>Step 5. Viewing images and pre–recognition results.....</b>	<b>8</b>
<b>Step 6. Creating a form identifier .....</b>	<b>9</b>
<b>Step 7. Testing the identifier element.....</b>	<b>10</b>
<b>Step 8. Specifying the order in which the field Recipe # and the name for the recipe must be detected.....</b>	<b>11</b>
<b>Step 9. Describing the Recipe # field .....</b>	<b>12</b>
<b>Step 10. Creating the Recipe element.....</b>	<b>12</b>
<b>Step 11. Creating the RecipeNumber element.....</b>	<b>13</b>
<b>Step 12. Creating the RecipeNumber block.....</b>	<b>16</b>
<b>Step 13. Describing the field which contains the name of the recipe.....</b>	<b>18</b>
<b>Step 14. Describing the Ingredients field .....</b>	<b>19</b>
<b>Step 15. Describing the field which contains the cooking instructions and the field which contains the cooking time.....</b>	<b>20</b>
<b>Step 16. Creating the CookingTimeHeader element .....</b>	<b>21</b>
<b>Step 17. Creating the CookingTime element .....</b>	<b>21</b>
<b>Step 18. Creating the CookingTime block .....</b>	<b>21</b>
<b>Step 19. Creating the InvertedHeader element .....</b>	<b>22</b>
<b>Step 20. Describing the Cooking field.....</b>	<b>23</b>
<b>Step 21. Creating the Serves element.....</b>	<b>23</b>
<b>Step 22. Creating the Servings element .....</b>	<b>24</b>
<b>Step 23. Describing the Cooking Instructions field .....</b>	<b>24</b>
<b>Step 24. Creating the CookingInstructions block.....</b>	<b>24</b>
<b>Step 25. The FlexiLayout is ready.....</b>	<b>25</b>

# Introduction

**⚠️Important!** For the sake of simplicity, a one-page document is used in this sample.

This sample illustrates the creation of a FlexiLayout for a document which has more different types of fields compared with the Halloween Registration Form. This sample will not only help you to go through all the steps of FlexiLayout creation, but will also teach you some tricks that can be used to describe the position of objects on the document image.

The FlexiLayout Studio project with the test images and a ready-made FlexiLayout can be found in **<disk name>:\Documents and Settings\All Users\Application Data\ABBYY\FlexiCapture\9.0\Samples\FlexiLayoutStudio\Recipe.**

The FlexiLayout must reliably detect the following fields on all the test pages:

- Receipt Name
- Recipe #
- Ingredients
- Cooking Time
- Cooking Description

## Easiest Recipes

---

### Barbecued Beefy Beans

*Baked barbecued beans are a delicious side dish for a cookout, potluck, or family meal.*

**Recipe #:** 878WN233

---

**INGREDIENTS:**

- 1 tablespoon vegetable oil
- 1/2 pound lean ground beef
- 1/4 cup chopped green pepper
- 1/2 cup chopped onions
- 1/2 cup chopped celery 1 can (8oz) tomato sauce
- 1/2 cup water
- 1 large clove garlic, minced
- 2 tablespoons vinegar
- 1 tablespoon dry mustard
- 1/2 teaspoon thyme
- 1 tablespoon brown sugar salt & pepper to taste
- 1 large can (approximately 28 ounces) pork and beans

---

**Cooking:**

Heat vegetable oil in a large skillet over medium heat; saute ground beef, onions, green bell pepper, and celery until beef is browned and vegetables are tender. Add tomato sauce, water, garlic, vinegar, mustard, thyme, and brown sugar. Blend well; simmer for 5 minutes. Taste and add salt and pepper as needed.

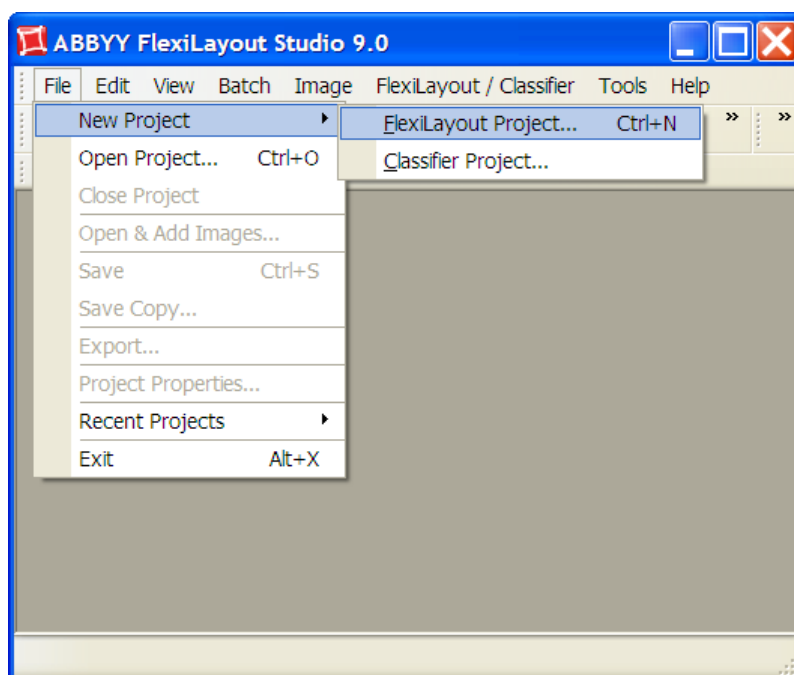
Pour canned pork and beans into a 2-quart casserole; gently stir in meat mixture. Bake at 375° for about 30 minutes.

**Barbecued beans serves 4 to 6.**

1 hour

## Step 1. Creating a new project

1. Create a new folder and name it **Recipes**.
2. Run ABBYY FlexiLayout Studio.
3. Create a new project (**File>New Project...**).



4. In the dialog box that opens, type the name for the project: **Sample2**.

Once the project is created, the **Recipes** folder will contain:

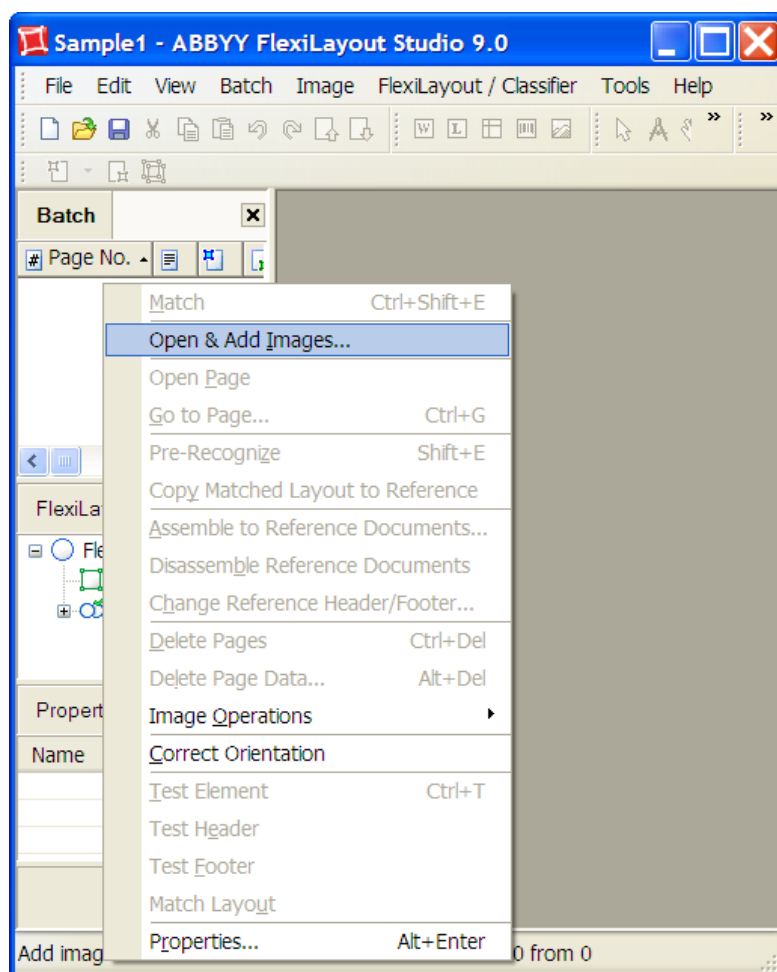
- **Sample2.fsp** – this is the file of the project
- **Sample2Batch** – this is the default folder for storing any images that are added to the batch
- **Sample2Templates** – this is the default folder for storing the FlexiLayout

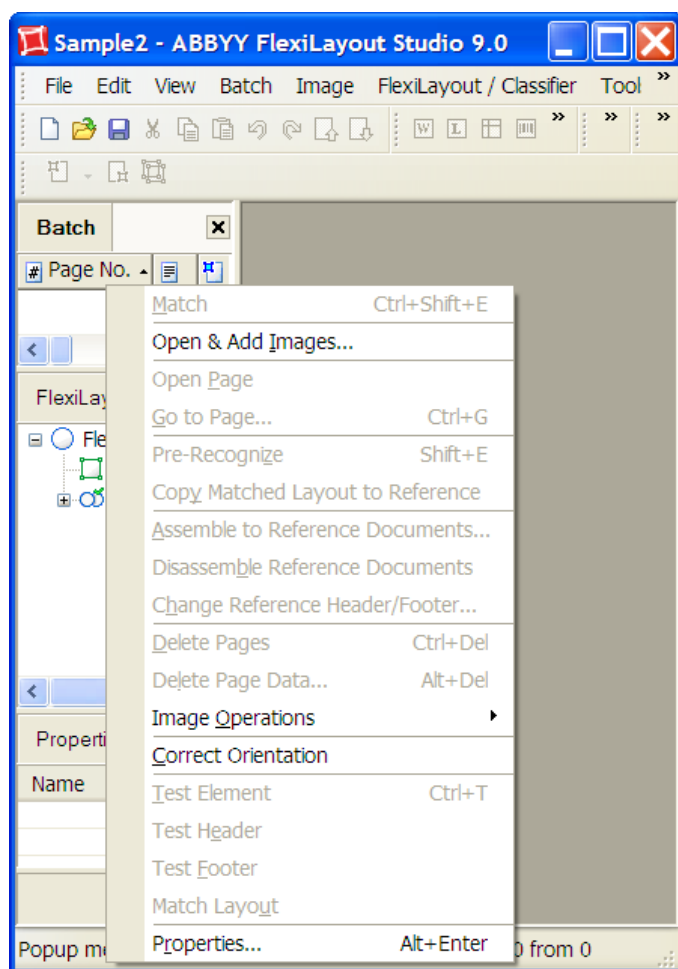
## Step 2. Adding images to the batch

Once the new project has been created, you must add all the test images to the batch. The test images will be used to test and adjust the FlexiLayout.

1. Click on the **Batch** tab in the main program window.
2. Select the **Open & Add Images** command in the **File** menu or in the local menu.



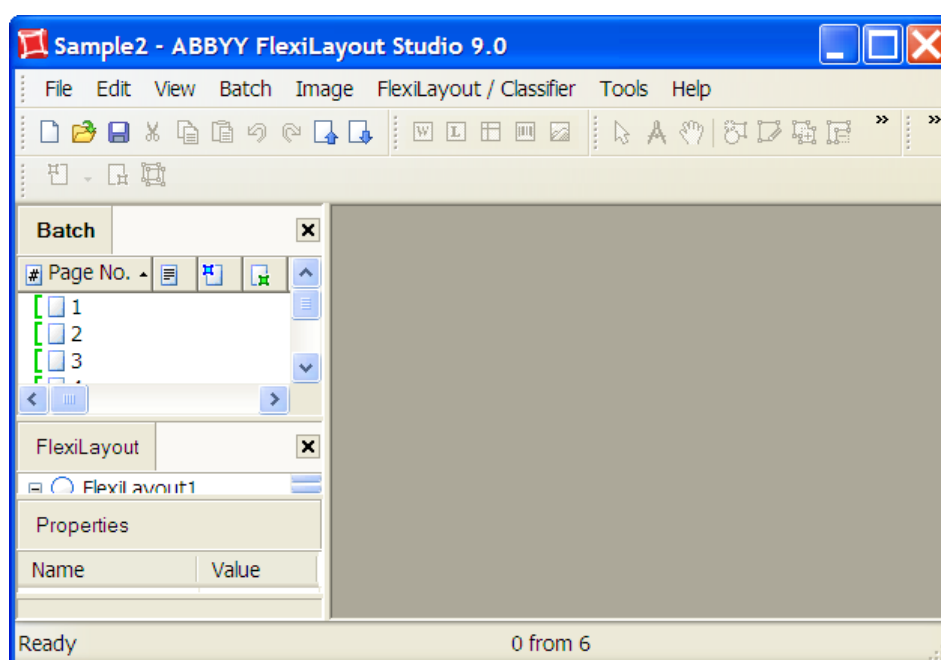





3. In the **Open Images** dialog box, select **Document per file** and specify the test image files.

(The test images for this sample can be found in <disk name>:\Documents and Settings\All Users\Application Data\ABBYY\FlexiCapture\9.0\Samples\FlexiLayoutStudio\Recipe.)

The test images you add to the batch will be displayed in the **Batch** window.



 **Note:** When adding images, you can immediately specify how the reference document assembly should be performed. Selecting **Document per file**, tells the program that each image is a separate document. For more about reference document assembly, see Help\Testing and Adjusting the FlexiLayout\Reference document assembly.

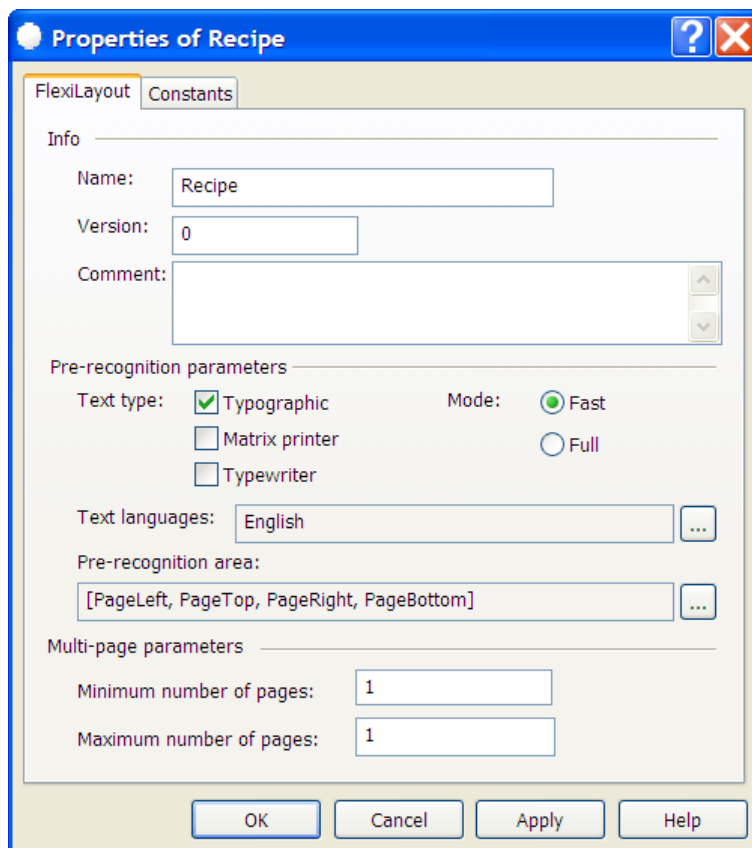
## Step 3. Setting the FlexiLayout properties

By default, the program has named the new FlexiLayout **FlexiLayout1** (see the **FlexiLayout** tab in the FlexiLayout Properties dialog box).

We recommend renaming the FlexiLayout and giving it a meaningful name for the sake of convenience.


To set the FlexiLayout properties (these also include the name of the FlexiLayout):

1. Double-click on the name of the **FlexiLayout** (i.e. FlexiLayout1) or right-click on the **FlexiLayout** and select **Properties** in the local menu.
2. In the **Name** field, type a new name for the FlexiLayout, e.g. **Recipe**.
3. Specify the pre-recognition parameters:
  - From the **Text language** list, select English as the pre-recognition language (the documents contain text in English).
  - Select **Typographic** in the **Text type** group (this is the default setting).
  - Select the **Fast** pre-recognition mode (this is the default setting).
  - **Multi-page parameters** – Our document consists of one page. Therefore set **Minimal number of pages** and **Maximal number of pages** to 1.



 **Note:** For more about selecting the right parameters, see Sample 1, Step 3.

For one-page documents, you do not need to use predefined compound **Header** and **Footer** elements to indicate the beginning and end of the documents. Therefore, you can remove them from the list of FlexiLayout elements.

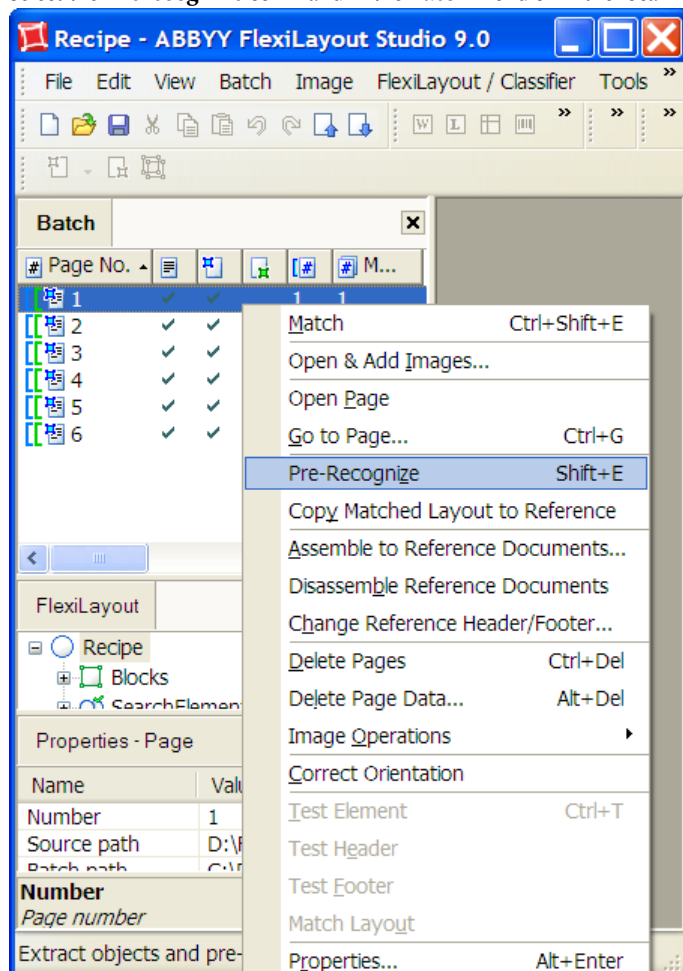
 **Note:** You can add **Header** and **Footer** elements to your FlexiLayout to indicate the beginning and end of the documents. If neither the Header nor the Footer element is found when matching the FlexiLayout, the program will use the maximum number of pages specified in the FlexiLayout as the number of pages in the document. For this particular document we specified 1 as the maximum number of pages.


## Step 4. Pre-recognition

Before you start creating form elements, you need to know which objects on the document can be used as "signposts" when looking for fields (blocks). These are usually pictures and/or text fragments that are consistently detected on all the images during pre-recognition.

To start pre-recognition:

1. Select all the images in the batch.
2. Select the **Prerecognize** command in the **Batch** menu or in the local menu of each image.




 **Note:** If you replace images in the batch or add new images, you will need to restart the pre-recognition procedure. If an image has not been pre-recognized, it will be pre-recognized when matching it with its FlexiLayout.

## Step 5. Viewing images and pre-recognition results

Pre-recognition shows that all the text objects corresponding to the titles of the fields, the inverted text, and the horizontal separators can be reliably detected. The program can use these objects as "signposts" or reference elements when looking for other objects.

Now you can create reference elements. Click on the **FlexiLayout** tab in the main program window.

 **Note:** For more about viewing images and pre-recognition results, see Sample 1, Steps 5 and 6.

 **Note:** For each image in the batch, you can create a reference layout which reflects the desired positions of the blocks on the image. A reference layout can be created manually or based on the FlexiLayout matching results. Reference layouts are used when testing and adjusting FlexiLayouts, allowing you to compare the matching results with the expected positions of the blocks on each of the images. See **Reference layout** for details.

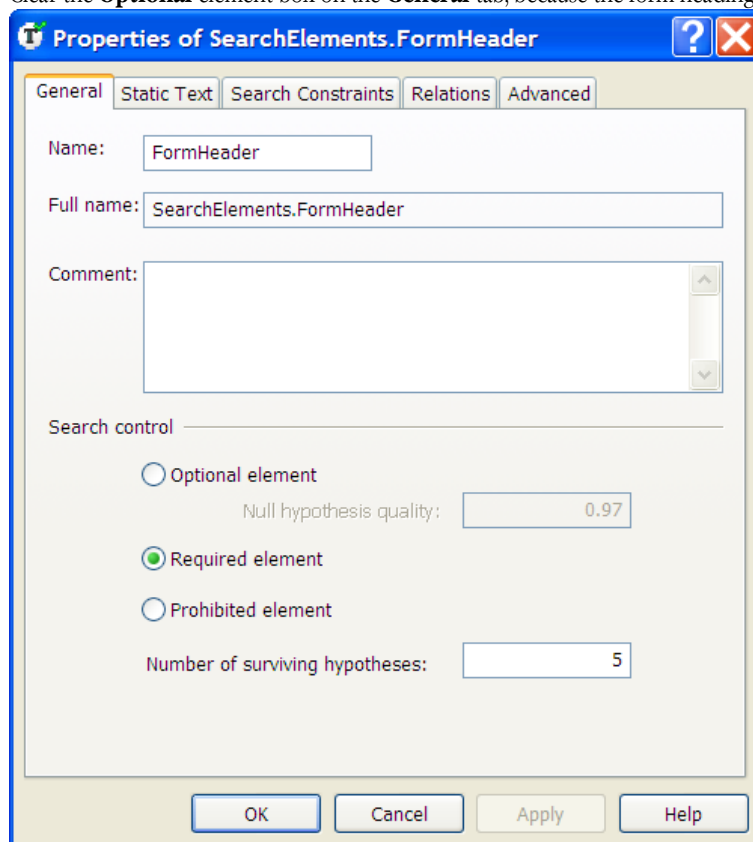
## Step 6. Creating a form identifier

When processing semi-structured documents in ABBYY FormReader, one would normally wish to exclude forms not belonging to the current type. One way to identify a document is to mark at least one element as required. In this particular case, the document heading will make a good identifier element, since it contains distinct text that can be easily read by the OCR engine.

 **Note:** An identifier element or set of elements can be described in a predefined **Header** element (not used in this sample).

The document heading will be used solely to identify the document as belonging to the given type and will not be recognized in ABBYY FormReader. In the FlexiLayout, describe the document heading as an element of the Static Text type:

1. Click on the **FlexiLayout** tab in the program main window.
2. Select **SearchElements** in the FlexiLayout tree.
3. Select the **Static Text** command in **FlexiLayout>Add Elements>Static Text** or in the local menu of the element.
4. In the **Name** field, type a name for the element, e.g. **FormHeader**.
5. Clear the **Optional** element box on the **General** tab, because the form heading is a required element.



**Properties of SearchElements.FormHeader**

General | Static Text | Search Constraints | Relations | Advanced

Name: FormHeader

Full name: SearchElements.FormHeader

Comment:

Search control

☐ Optional element  
Null hypothesis quality: 0.97

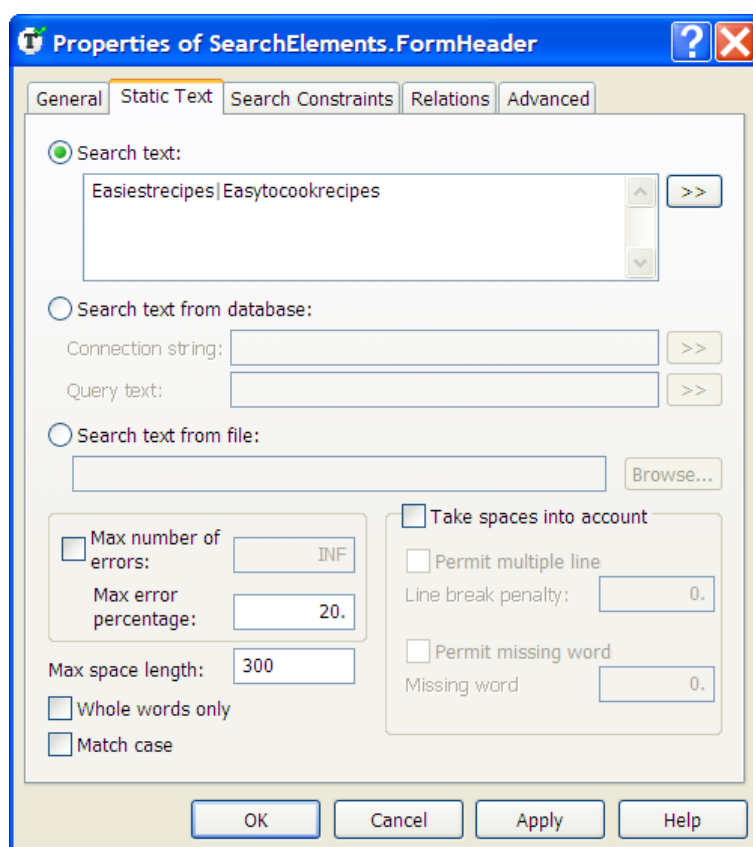
☒ Required element

☐ Prohibited element

Number of surviving hypotheses: 5

OK Cancel Apply Help

6. Click on the **Static Text** tab.



7. In the **Search text** field, type the text to find.  
The batch contains test forms that have different headings: **Easiest Recipes** or **Easy to Cook Recipes**. Enter both headings.  
The headings are written in one line on all the test images. Therefore, you can type the headings without spaces to speed up looking for single-line static text. Separate the two alternative headings by "|".
8. Set the maximum number of errors that the detected text may contain (either in percentage points or as a number). In this particular case we recommend setting the **Max error percentage** to 20, allowing 5 errors among the 25 characters of the form heading.

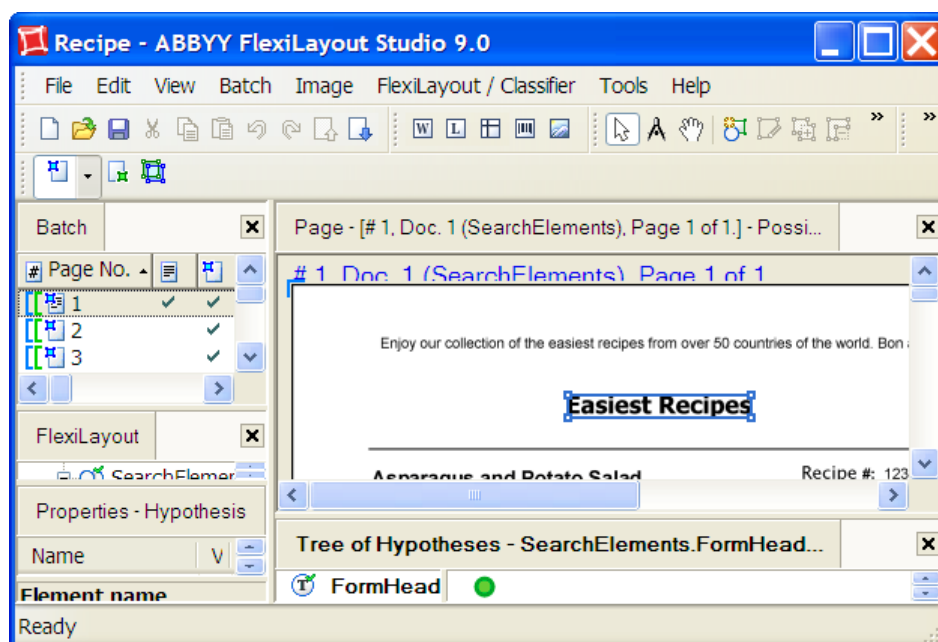
**Note:** The maximum number of errors is selected by method of trial and error.

## Step 7. Testing the identifier element

To check that the program can reliably detect the identifier element **FormHeader**, try matching the FlexiLayout with each image in the batch:

1. Open each image.
2. Select the **Match FlexiLayout & Show Hypotheses** command in the **Page** menu or in the local menu of each image.

If the image and its FlexiLayout have been matched successfully, the hypothesis which the program formulates about the FormHeader element will be marked with in the **Tree of Hypotheses**. If you click on the hypothesis, the program will draw a blue frame around the detected element. You will see the properties of the hypothesis in the **Properties** window.



When you try matching the FlexiLayout with the images, you will see that on some of the images the program has found a part of the phrase above the form heading instead of the heading itself. The phrase "Enjoy our collection of easiest recipes from over 50 countries of the world" occurs on almost all the images in the batch. To prevent the program from mistaking this phrase for the form heading, you need to adjust the properties of the identifier element. Since the phrase "Enjoy our collection..." is always located above the form heading, the program must look for an object that has all the properties specified by the element and is located closest to the bottom edge of the image.

To specify additional search constraints:

1. Open the **Properties** dialog box for the **FormHeader** element.
2. Click on the **Advanced** tab.
3. In the **Advanced pre-search relations** field, specify an additional search constraint: *Look for an object that is closest to the bottom edge of the image*. In FlexiLayout Language this must be written as follows: **NearestY: PageRect.Bottom;**

**Note:** The same constraint can also be specified via the program's graphical user interface. Click the **Relations** tab, select **Nearest**, and in the **To:** drop-down list select **Page bottom edge**.

**Note:** The Nearest function tells the program that out of the several hypotheses for the element it should look for the one nearest to a particular other element or point on the image (the distance between element centers is measured). Once the function is executed, only one hypothesis remains. This function does not take into account the quality of the hypotheses, as the choice is made at the stage when hypotheses are still being generated.

Once have matched the FlexiLayout with the images again, you will see that the program has reliably detected the form headings, which are described by the FormHeader element, on all the test images.

## Step 8. Specifying the order in which the field Recipe # and the name for the recipe must be detected

Usually fields are detected through their titles, but not all the fields on our documents have titles. For example, the field containing the name of the recipe has no title. Therefore, a different approach must be used to detect this field.

The most obvious solution which first comes to mind is to tell the program that the recipe name is the object that is closest to the form's heading. But if you look at test image 6, you will see that on this image the closest field is **Recipe #:**. Sometimes it is possible to specify additional search constraints which help the program to distinguish between the two fields, but in this particular case it is very difficult to differentiate between the two fields — the recipe name and the **Recipe #** field are located very close to each other and have similar structures:

1. The title **Recipe #:** and the field itself are located on the same level. The field may contain the same characters that may be used in the recipe name.
2. One cannot be sure that the text in the recipe name will always be longer than the title **Recipe #:** and its field.

Note, however, that pre-recognition consistently detects the title **Recipe #:** on all the test images. This means that you can tell the program to look for **Recipe #:** first. Then the program will look for the recipe name: the program must look for an object that is closest to the form heading, but which is NOT **Recipe #** (which will have been detected by then).

## Step 9. Describing the Recipe # field

The **Recipe #:** title is detected reliably on all the test images and does not repeat on each given form. This means that for **Recipe #:** you require:

1. a **Recipe** element of the **Static Text** type which will correspond to the title **Recipe #**.
2. a **RecipeNumber** element of the **Character String** type which will correspond to the contents of the field.
3. a **RecipeNumber** block which will correspond to the **Recipe #** field.

## Step 10. Creating the Recipe element

To create the **Recipe** element:

1. Create an element of the **Static Text** type and name it **Recipe**.  
Do not clear the **Optional element** box which is selected by default. Unlike the form heading, which is used as a form identifier and for this reason was specified as a required element, all the other form elements are assumed to be optional. This will allow the program to formulate hypotheses even for those elements for which it fails to find corresponding objects on the images.

**Properties of SearchElements.Recipe**

General Static Text Search Constraints Relations Advanced

Name: Recipe

Full name: SearchElements.Recipe

Comment:

Search control

☒ Optional element  
Null hypothesis quality: 0.97

☐ Required element

☐ Prohibited element

Number of surviving hypotheses: 5

OK Cancel Apply Help

2. Click on the **Static Text** tab.
3. In the **Search text** field, type the text to search. Since the field title has only one line on all the test images, you can type it without spaces: **Recipe#:**.
4. Set the **Max error percentage** to 10 (since the title of the field consists of only one word, it is unlikely to contain a very large number of errors).

Try matching the FlexiLayout with the test images and make sure that the program successfully finds the field title on all the images.



## Step 11. Creating the RecipeNumber element

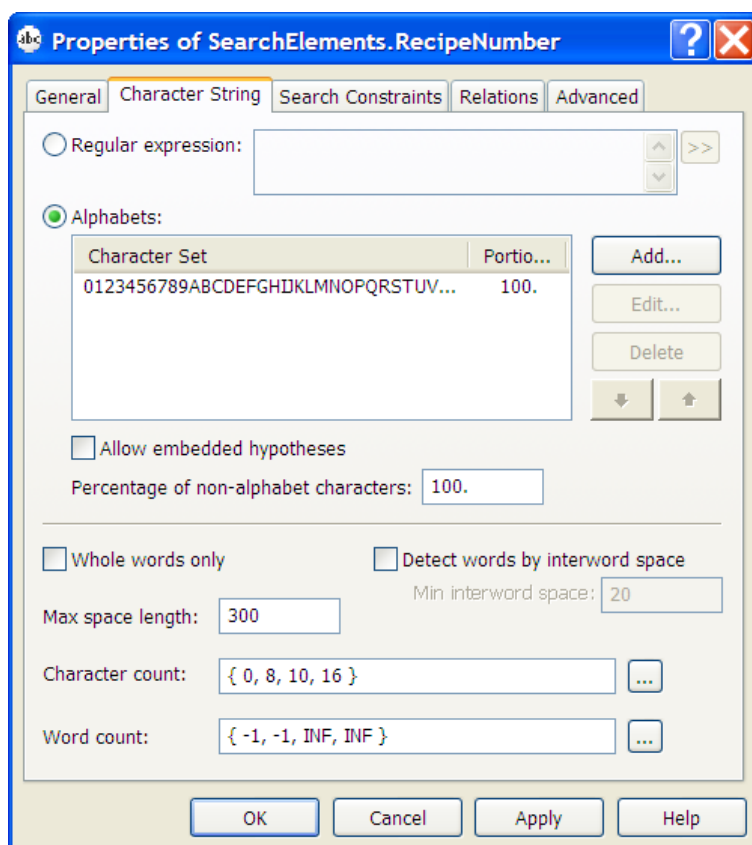
We assume that the **Recipe #:** field always contains only one line. Since, unlike the field title, the contents of this field is not fixed, you must describe it as an element of the **Character String** type.

To create the **RecipeNumber** element:

1. Create an element of the Character String type and name it **RecipeNumber**.

The screenshot shows a dialog box titled "Properties of SearchElements.RecipeNumber". It has five tabs: "General", "Character String", "Search Constraints", "Relations", and "Advanced". The "General" tab is active. Inside the dialog, there are three text input fields: "Name" (containing "RecipeNumber"), "Full name" (containing "SearchElements.RecipeNumber"), and "Comment" (empty). Below these is a "Search control" section with three radio buttons: "Optional element" (selected), "Required element", and "Prohibited element". To the right of the "Optional element" radio button is a text box for "Null hypothesis quality" containing "0.97". Below the radio buttons is a text box for "Number of surviving hypotheses" containing "5". At the bottom of the dialog are four buttons: "OK", "Cancel", "Apply", and "Help".

2. Click on the **Character String** tab.

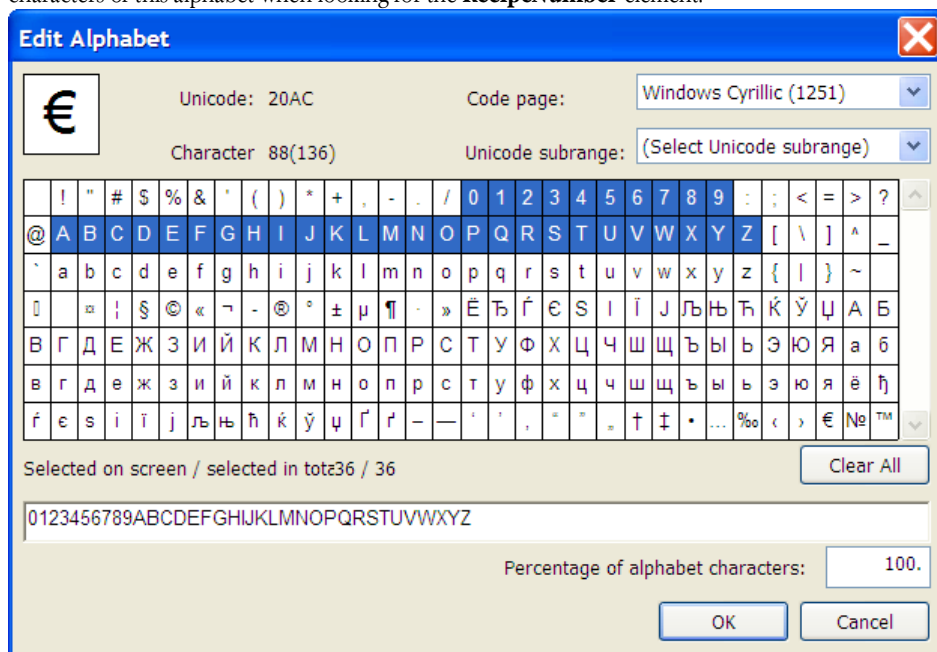


The dialog box 'Properties of SearchElements.RecipeNumber' has tabs: General, Character String, Search Constraints, Relations, and Advanced. The 'Character String' tab is active. It contains a 'Regular expression' field with a search icon and a 'Portio...' button. Below is the 'Alphabets' section with a table:

Character Set	Portio...
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ...	100.

Buttons: Add..., Edit..., Delete, and arrow buttons. Below the table are checkboxes for 'Allow embedded hypotheses' and 'Whole words only'. A text field for 'Percentage of non-alphabet characters' is set to 100. Further down are checkboxes for 'Detect words by interword space' and a text field for 'Min interword space' set to 20. Text fields for 'Max space length' (300), 'Character count' ({ 0, 8, 10, 16 }), and 'Word count' ({ -1, -1, INF, INF }) are also present. At the bottom are OK, Cancel, Apply, and Help buttons.

- Set the alphabet, i.e. all the characters that may be encountered in the recipe numbers. Judging by the test images, the alphabet includes digits and capital letters of the English alphabet. The order of letters and digits is not known beforehand. To set the alphabet, click the **Add** button and select the required characters in the **Add New Alphabet** dialog box that opens.
- Note:** The selected characters will be displayed in the **Character set** column (**Character String** tab, **Alphabet** field).
- Set the **Percentage of alphabet characters** to 100. This means that the program will take into account only the characters of this alphabet when looking for the **RecipeNumber** element.




The 'Edit Alphabet' dialog box shows the character '€' (Unicode: 20AC) and 'Code page: Windows Cyrillic (1251)'. It includes a 'Character 88(136)' field and a 'Unicode subrange' dropdown. Below is a large character grid with various symbols and letters. At the bottom, a text field shows '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ' and a 'Percentage of alphabet characters' field is set to 100. Buttons for OK, Cancel, and Clear All are at the bottom right.


- Note:** You can specify several alphabets for one element. In this case, set the **Percentage of alphabet characters** for each alphabet used in the element.
- Clear the **Allow embedded hypotheses** box. This will allow the program to formulate hypotheses which have the maximum length and meet all the search criteria. Otherwise the program may formulate several embedded hypotheses, each

consisting of portions of one of the selected alphabets (taking into account the allowed percentage of non-alphabet characters).

6. Set the Percentage of non-alphabet characters to 20%.

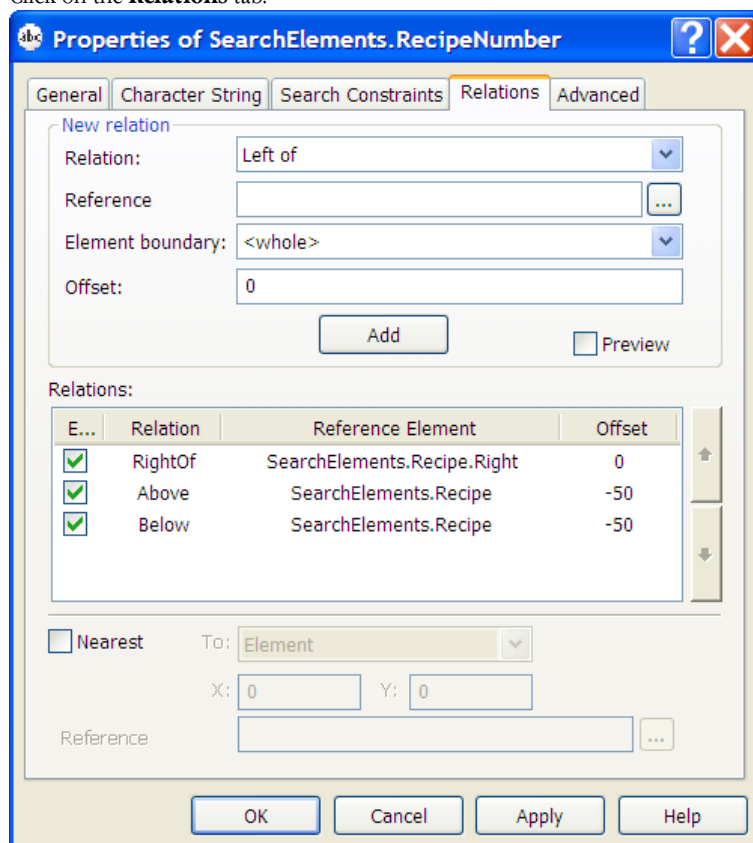
 **Note:** This value can only be selected by trial and error and can be changed when adjusting the FlexiLayout.

7. In the **String length** field, specify this fuzzy range: {0, 8, 10, 16}. This is an estimation of the length of the string of characters. We assume that the number is always 8 digits long. To be on the safe side, tell the program that the number may have from 8 to 10 digits. Any hypotheses outside this range will be penalized.

 **Note:** This value can only be selected by trial and error and can be changed when adjusting the FlexiLayout.

8. In the **total length of spaces** field, specify the fuzzy range that estimates the total length of all the spaces between the characters in the hypothesis (by default measured in dots, 1 dot = 1/300 inch). Do not change the default range of {-1, -1, INF, INF}. This means that there are no constraints on the number of spaces in the hypothesis.

9. Click on the **Relations** tab.



The screenshot shows the 'Properties of SearchElements.RecipeNumber' dialog box with the 'Relations' tab selected. The 'New relation' section has 'Relation' set to 'Left of', 'Reference' is empty, 'Element boundary' is '<whole>', and 'Offset' is '0'. Below this is a table of existing relations:

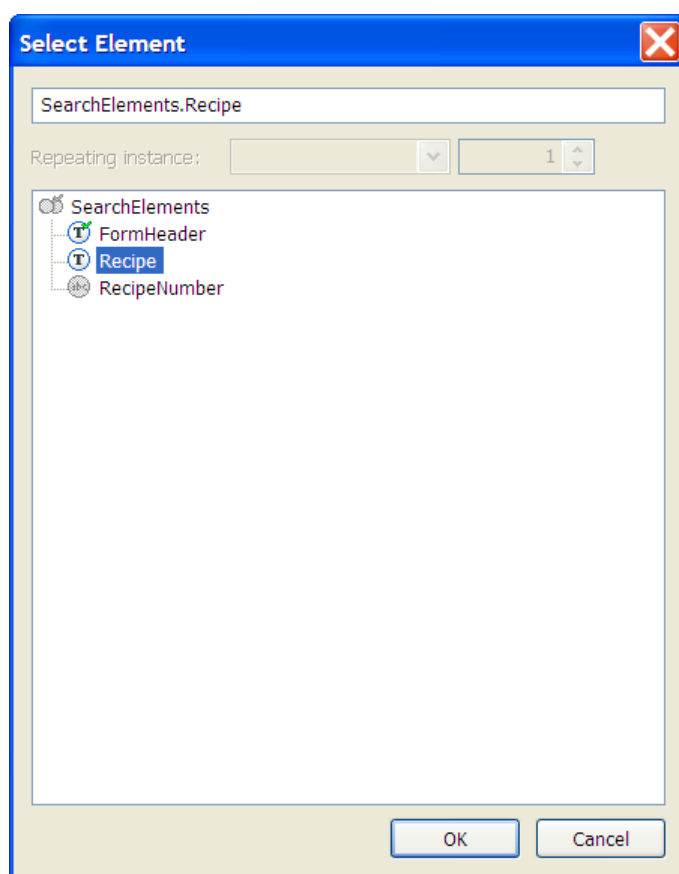
E...	Relation	Reference Element	Offset
<input checked="" type="checkbox"/>	RightOf	SearchElements.Recipe.Right	0
<input checked="" type="checkbox"/>	Above	SearchElements.Recipe	-50
<input checked="" type="checkbox"/>	Below	SearchElements.Recipe	-50

At the bottom, there is a 'Nearest' checkbox (unchecked), a 'To:' dropdown set to 'Element', and 'X:' and 'Y:' fields both set to '0'. There is also a 'Reference' field with an empty text box and a button to select an element. At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.


Use the Recipe element as the reference element for the RecipeNumber element. The number of the recipe will always be located to the right of the field title and will be located on the same level with the title. Therefore, you must specify the location of the element relative to the title and to the title's top and bottom borders. Otherwise, the program will formulate a valid hypothesis for any text fragment that consists of characters of the set alphabet, even if it is located higher or lower than the field title.

to sum up: The program must look for the recipe number described by the **RecipeNumber** element to the right of the field title described by the **Recipe element** and located on the same level with the Recipe heading.

10. Specify that the RecipeNumber element is located to the right of the right border of the Recipe element. Select **Recipe** in the Select Element window. From the **Relation** drop-down list, select **Rightof**, in the **Element border** drop-down list select **right** and leave the value of the **Offset** field unchanged (the default value is 0). Click the **Add** button.



11. Specify that the **RecipeNumber** element is located not lower than the **Recipe** element. Select **Recipe** in the **Select Element** window. In the **Relation** drop-down list, select **Above** and set **Offset** to -50 (this value can only be selected by trial and error). This will give the program some leeway when detecting the position of the element relative to the top border of the field title. Negative offset values allow you to specify that the bottom element border is located below the bottom border of the title. Click the **Add** button.
12. Specify that the **RecipeNumber** element is located not higher than the **Recipe** element. Select **Recipe** in the **Select Element** window. In the **Relation** drop-down list, select **Below** and set **Offset** to -50 (this value can only be selected by trial and error). This will give the program some leeway when detecting the position of the element relative to the top border of the field title. Negative offset values allow you to specify that the top element border is located above the top border of the title. Click the **Add** button.

 **Note:** If you select <whole> in the **Element border** field, positive Offset values allow you to specify only the following locations: to the right of the right-hand border of the element, to the left of the left-hand border of the element, higher than the top border of the element, or lower than the bottom border of the element. If you select <top> and <bottom> in the **Element border** field, positive **Offset** values will limit the search zone by the top and bottom borders of the element and exclude hypotheses which go beyond the left and right borders of the title of the field.

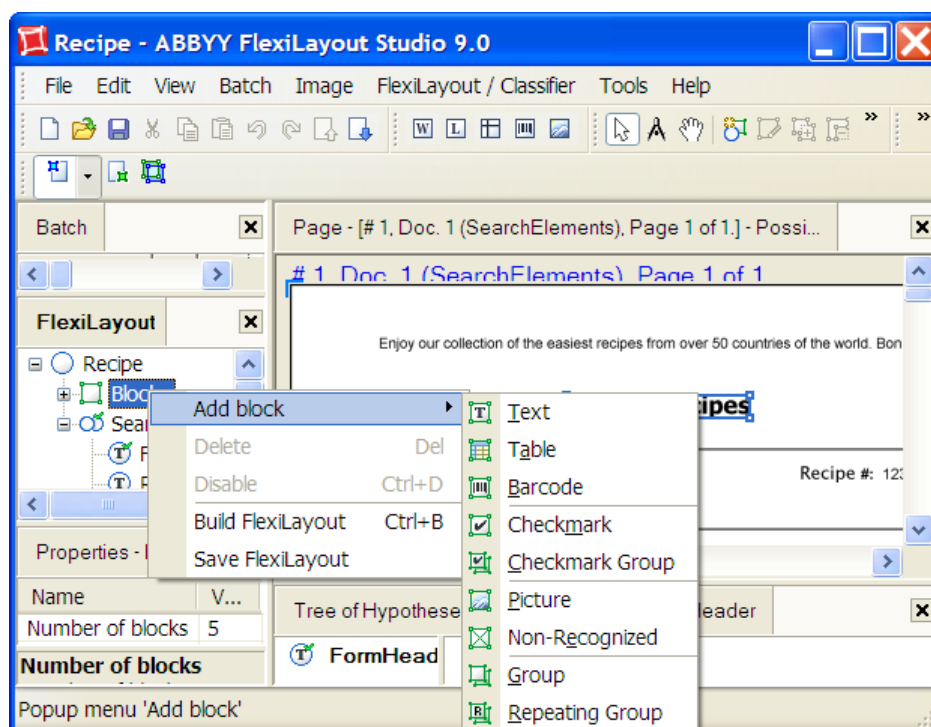
Try matching the FlexiLayout with the test images and make sure that the program successfully finds the field containing the number of the recipe on all the images.


## Step 12. Creating the RecipeNumber block

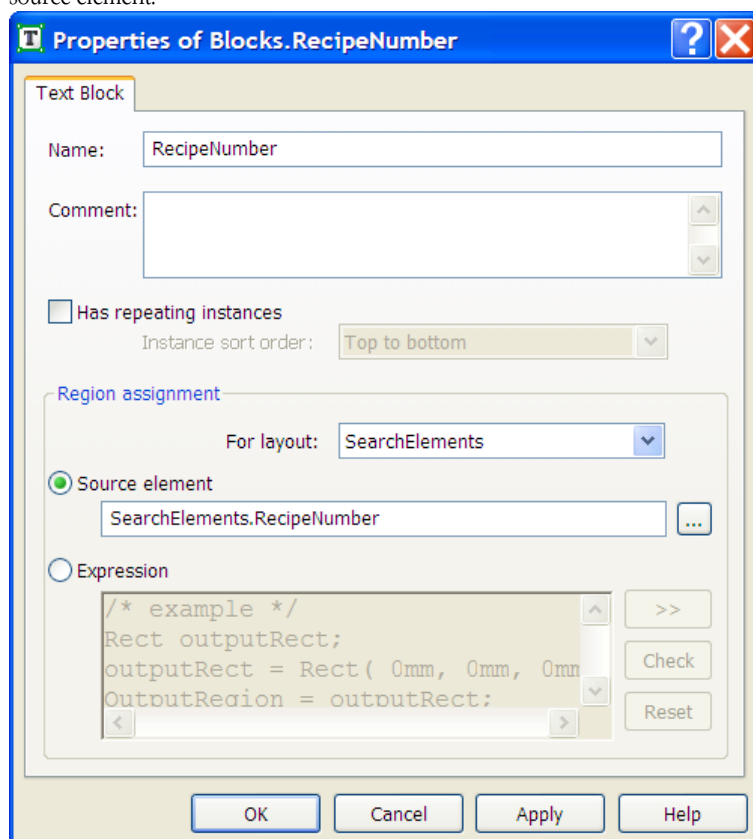
To enable ABBYY FormReader to recognize the number of the recipe, you must create a block corresponding to the Recipe # field and specify its location.

To create a block corresponding to the **Recipe** # field:

1. Click on the **FlexiLayout** tab in the program main window.
2. Select **Blocks** in the FlexiLayout tree.
3. Select **Add Block>Text** in the **FlexiLayout** menu or in the local menu.



4. In the **Properties** dialog box that opens, type a name for the block in the **Name** field, e.g. **RecipeNumber**.
-  **Note:** The name of the block need not coincide with the name of the element corresponding to the **Recipe #** field. We recommend giving the block the same name as the field for the sake of convenience.
5. To describe the location of the block, select **Source element**. Click "... " and specify the **RecipeNumber** element as the source element.



## Step 13. Describing the field which contains the name of the recipe

Describe the recipe name field as the object closest to the bottom border of the form's heading, but exclude those regions which contain the elements **Recipe** and **RecipeNumber** (see Step 8).

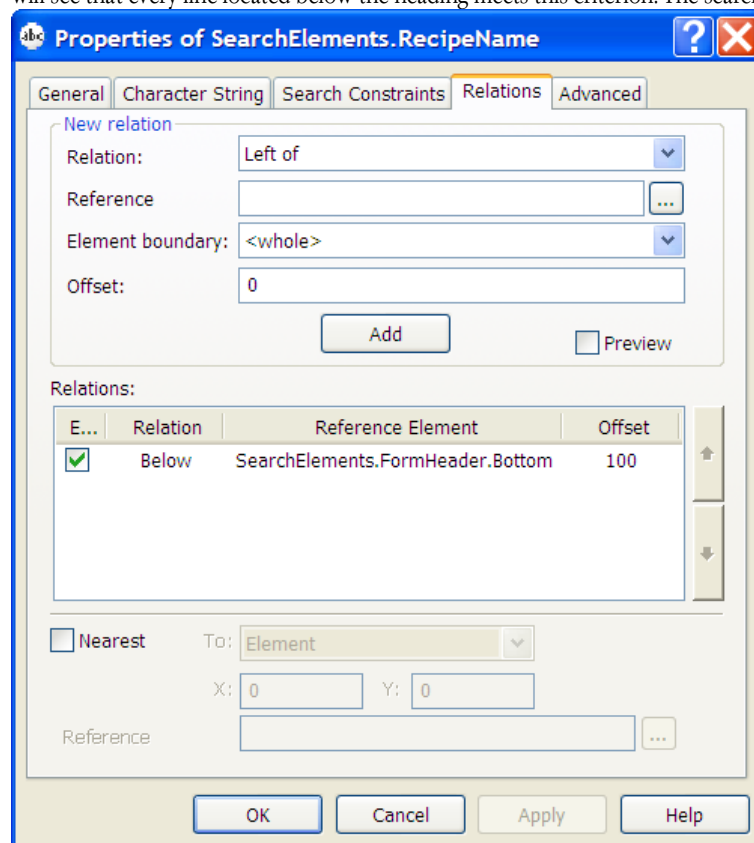
1. an element of the **Character String** type and name it **RecipeName**. This element will correspond to the name of the recipe.

2. a **RecipeName** block corresponding to the field which contains the name of the recipe.

To create the **RecipeName** element:

1. Create an element of the **Character String** type and name it **RecipeName**.
2. Click on the **Character String** tab.
3. Set the required alphabet. Judging by the test images, the alphabet includes all the letters of the English alphabet, digits and the characters "#", "-", "&", and ".". The order and number of characters are not known in advance.
4. Set the Percentage of alphabet characters to 100.
5. Clear the Allow embedded hypotheses box.
6. Set the Percentage of non-alphabet characters to 20%.
7. In the **String length** field, specify the fuzzy range that estimates the total length of all the character strings. Do not change the default range of {-1, -1, INF, INF}. This means that there are no constraints on the length of the character string.
8. In the **total length of spaces** field, specify the fuzzy range that estimates the total length of all the spaces between the characters in the hypothesis (by default measured in dots, 1 dot = 1/300 inch). Do not change the default range of {-1, -1, INF, INF}. This means that there are no constraints on the number of spaces in the hypothesis.
9. Click on the **Relations** tab.

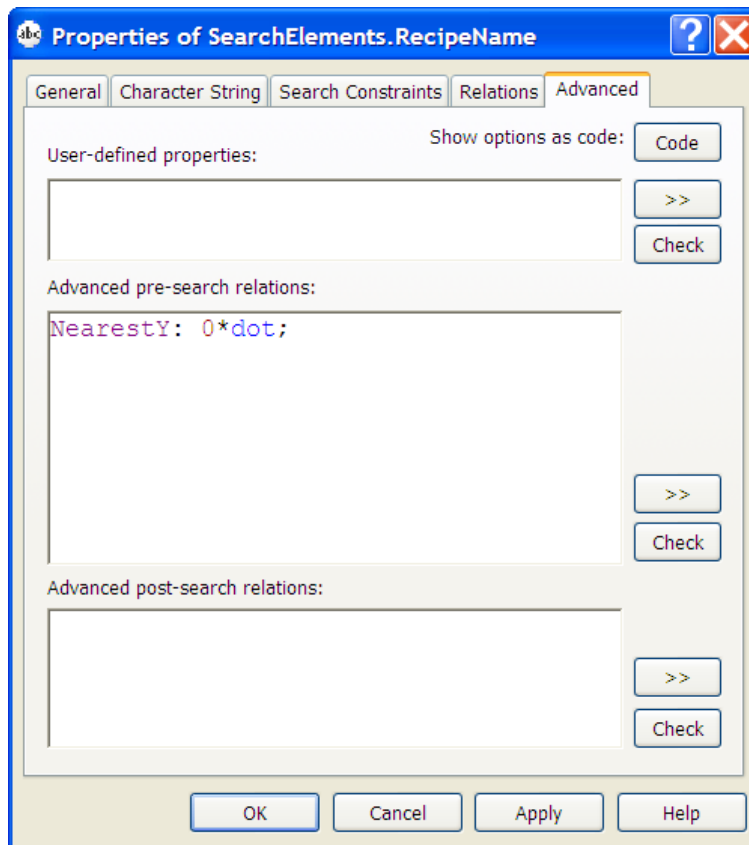
Tell the program that the name of the recipe is located at least 100 dots lower than the bottom border of the form's heading, i.e. there must be a gap of at least 100 dots wide between the heading and the recipe name. If you look at the test images you will see that every line located below the heading meets this criterion. The search area must be narrowed.



10. Click on the **Advanced** tab.

In the **Advanced pre-search relations** field, specify an additional search constraint: *Look for the line closest to the top edge of the form*. In the FlexiLayout Language this must be written as follows: **NearestY: 0\*dot;** together with the search constraint on the **Relations** tab this narrows the search area to one line located closest to the form heading.

**Note:** The same constraint can also be specified via the program's graphical user interface. Click the **Relations** tab, select **Nearest**, and in the **To:** drop-down list select **Page top edge**.



11. Click on the **Search Constraints** tab.
12. As stated in Step 8, the regions of the elements **Recipe** and **RecipeNumber** must be excluded from the search area for **RecipeName**. Click the **Add...** button next to the **Exclude relation** field and in the **Select Element to Exclude** dialog box that opens, select the **Recipe** and **RecipeNumber** elements.

Try matching the FlexiLayout with the test images and make sure that the program successfully finds the field which contains the name of the recipe on all the images.

Finally, create the **RecipeName** block. The **RecipeName** block is created similarly to the **RecipeNumber** block. The only exception is that you must specify the **RecipeName** element as the source element for the **RecipeName** block.

## Step 14. Describing the Ingredients field

Judging by the test images, the list of ingredients is a text fragment which is always located below the title of the **Ingredients** field but above the separator located over the title of the **Cooking** field.


To make the description of the above elements easier and to restrict the number of possible hypotheses, group them into a compound element of the Group type:

1. Create a compound element of the **Group** type and name it **RecipeContents**.
2. Within the **RecipeContents** element, create an element of the **Static Text** type and name it **IngredientsHeader**. This element will correspond to the title **Ingredients**. Specify the properties of the element as in Step 10 for the **Recipe** element.
3. Within the **RecipeContents** element, create an element of the **Separator** type which will correspond to the separator above the title **Cooking**. On the **Separator** tab, specify the orientation of the separator: **Horizontal**. In the element properties dialog box, click on the **Relations** tab and specify the Relations: below the title **Ingredients**: (i.e. **Below the IngredientsHeader** element).

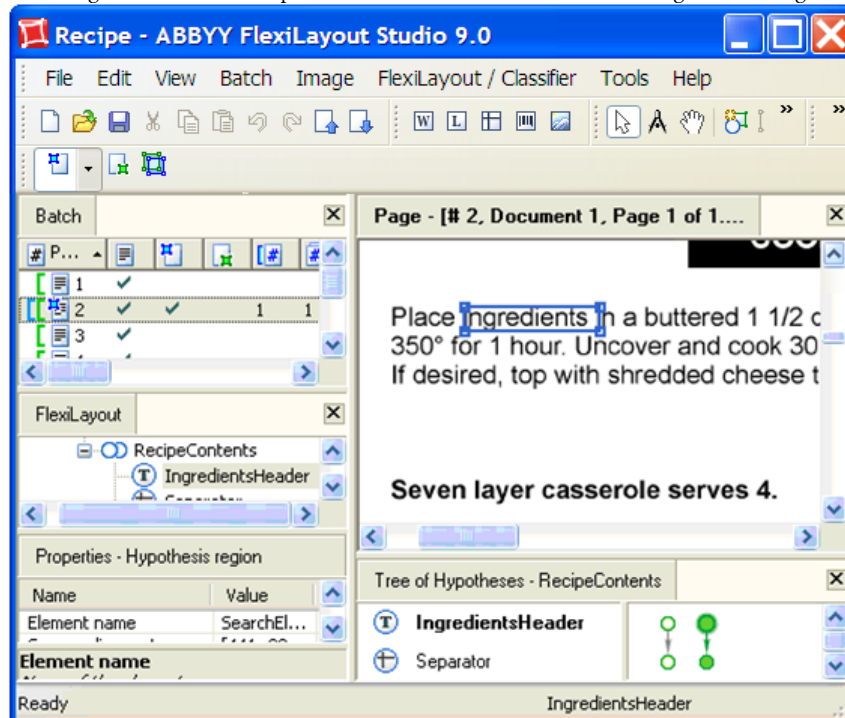
If you try matching the FlexiLayout, you will see that sometimes the program treats the lower edge of an image as a separator.

**Note:** Certain scanner settings may make the edges of the form look like dark lines. These may also be wrongly recognized as separators.


The problem can be solved by adding another search constraint. In the **Advanced pre-search relations** field (**Properties** dialog box, **Advanced** tab), add this search constraint: *Look for the separator closest to the title Ingredients*:. In the FlexiLayout Language this must be written as follows: **Nearest: SearchElements.RecipeContents.IngredientsHeader;**

 **Note:** The same constraint can also be specified via the program's graphical user interface. Click the **Relations**, select **Nearest**, and in the **To:** drop-down list select **Element** and specify **IngredientsHeader** as the **Reference element**.

4. Match the adjusted FlexiLayout once again.
5. Matching the FlexiLayout with image 2 has generated two hypotheses for the **IngredientsHeader** element and two hypotheses for the **Separator** element. Note that the program has selected the best path which mistakenly recognized the word "ingredients" in the recipe as the title of the field and the lower edge of the image as the separator.



There is nothing unusual in this, because the found text fragment is so similar to the text described by the properties of the **IngredientsHeader** element (**Static text** tab, **Search field**), that it cannot be cut off by the error threshold set in the **Max error percentage** field (20). The sole difference between the two text fragments is the colon character ":".

 **Note:** We cannot set **Max error percentage** = 0 (or **Max number of errors** = 0) because recognition errors may occur due to varying image quality. And if the **Ingredients** title is not detected, the list of ingredients will not be detected either.

The problem can be solved by adding another search constraint for the **IngredientsHeader** element. In the **Advanced pre-search relations** field (**Properties** dialog box, **Advanced** tab), add this search constraint: *Look for the field title in the upper part of the image*. In the FlexiLayout Language this must be written as follows: **Above: PageRect.top + PageRect.Height/2;**

6. Within the **RecipeContents** element, create an element of the **Text Fragment** type and name it **Ingredients**. This element will correspond to the **Ingredients** field.  
Set the Relations (element properties dialog box, **Relations** tab: below the **Ingredients** heading (**Below** the **IngredientsHeader**), but above the separator (**Above** the Separator element).  
Try matching the FlexiLayout with the test images and make sure that the program successfully finds the **Ingredients** field on all the images.

Finally, create the **Ingredients** block. The **Ingredients** block is created similarly to the **RecipeNumber** block. The only exception is that you must specify the **Ingredients** element as the source element for the **Ingredients** block.

## Step 15. Describing the field which contains the cooking instructions and the field which contains the cooking time

Create a group element for the cooking time and cooking instructions:

1. Create a compound element of the **Group** type and name it **Cooking**
2. The **Cooking** element must contain the following elements




- 2.1 **CookingTimeHeader** element (of **Static text** type), which corresponds to the time units specified in the **Cooking Time** field (see Step 16 for details).
- 2.2 **CookingTime** element (of **Character String** type), which corresponds to the **Cooking Time** field (see Step 17 for details).
- 2.3 **InvertedHeader** element (of **Object Collection** type), which corresponds to the title of the **Cooking Instructions** field (see Step 19 for details).
- 2.4 Compound **Instructions** element (of **Group** type), which corresponds to the **Cooking Instructions** field (see Step 20 for details).

## Step 16. Creating the CookingTimeHeader element

To create the **CookingTimeHeader** element:

1. Create an element of the **Static Text** type and name it **CookingTimeHeader**.
2. Click on the **Static Text** tab.
3. In the **Search text** field, type the text to search: **minutes|hour|hours**.
4. Set Max error percentage to 10.
5. Click on the **Advanced** tab.
6. The field which contains the cooking time is located below all the other objects on the image. This means that you can set an additional search constraint in the **Advanced pre-search relations** field: *Search for the object closest to the bottom of the page*. In the FlexiLayout Language this must be written as follows: **NearestY: PageRect.Bottom;**

 **Note:** The same constraint can also be specified via the program's graphical user interface. Click the **Relations** tab, select **Nearest**, and in the **To:** drop-down list select **Page bottom edge**.

## Step 17. Creating the CookingTime element

To create the **CookingTime** element:

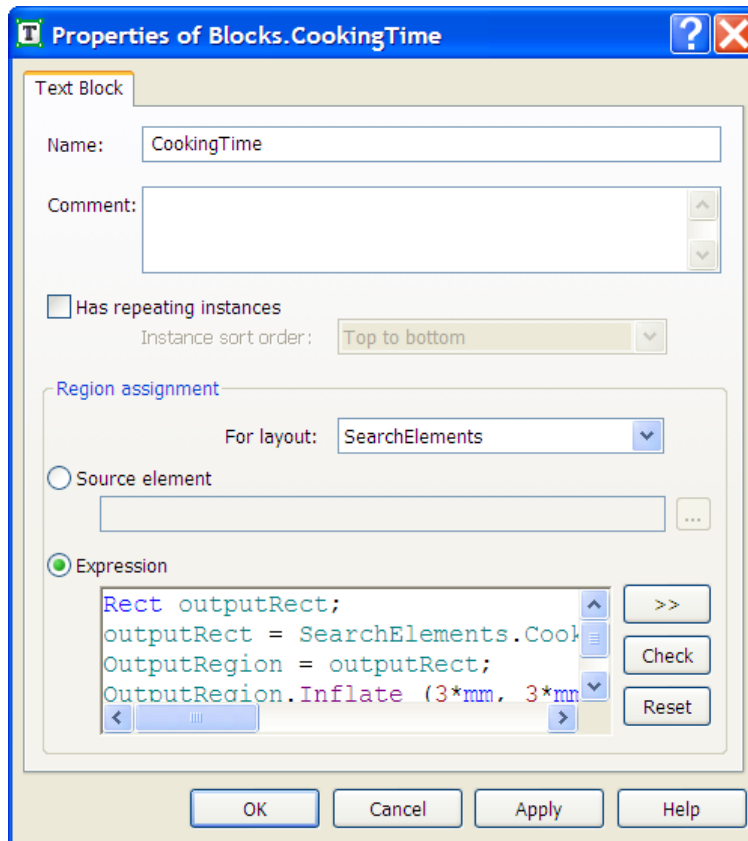
1. Create an element of the **Character String** type and name it **CookingTime**.
2. Click on the **Character String** tab.
3. Set the required alphabet. Judging by the test images, the alphabet includes only digits.
4. Set the Percentage of alphabet characters to 100.
5. Clear the Allow embedded hypotheses box.
6. Set the Percentage of non-alphabet characters to 10%.
7. In the **String length** field, specify this fuzzy range {0, 1, 3, 4}. This interval is an estimation of the length of the string of characters. We assume that the string may be 1 to 3 characters long. Any hypotheses outside this range will be penalized.
8. In the **total length of spaces** field, specify the fuzzy range that estimates the total length of all the spaces between the characters in the hypothesis (by default measured in dots, 1 dot = 1/300 inch). Do not change the default range of {-1, -1, INF, INF}. This means that there are no constraints on the number of spaces in the hypothesis.
9. Click on the **Relations** tab.
10. On the **Relations** tab, specify that the number indicating the cooking time is always located to the left of the units of time, and that the distance between the digits and the time units cannot exceed 100 dots (i.e. **Leftof** the **CookingTimeHeader** element, **Offset** = 0 and **Rightof** the left border of the **CookingTimeHeader** element, **Offset** = -100), **Element border** = left
11. Since the cooking time is always written on the same level as the units of time, add the following search constraints:
  - **Above** the **CookingTimeHeader** element, **Offset** = -20, **Element border** = bottom
  - **Below** the **CookingTimeHeader** element, **Offset** = -20, **Element border** = top

## Step 18. Creating the CookingTime block

To create the **CookingTime** block:

1. Activate the **FlexiLayout** window in the program main window.
2. Select the **Blocks** object in the FlexiLayout tree.
3. Select **Add Block>Text** in the **FlexiLayout** menu or **New>Text** in the local menu.
4. In the **Properties** dialog box that opens, type a name for the block in the **Name** field, e.g. **CookingTime**.
5. to describe the location of the block, select **Region expression**.

6. Describe the block as a rectangle created by merging the rectangles around the **CookingTimeHeader** and **CookingTime** elements. To give the program some leeway, enlarge the height and width of the resulting rectangle by 3mm. In the FlexiLayout Language this must be written as follows:
- ```
Rect outputRect;
outputRect = SearchElements.Cooking.CookingTime.Rect Or
SearchElements.Cooking.CookingTimeHeader.Rect;
OutputRegion = outputRect;
OutputRegion.Inflate (3*mm, 3*mm);
```

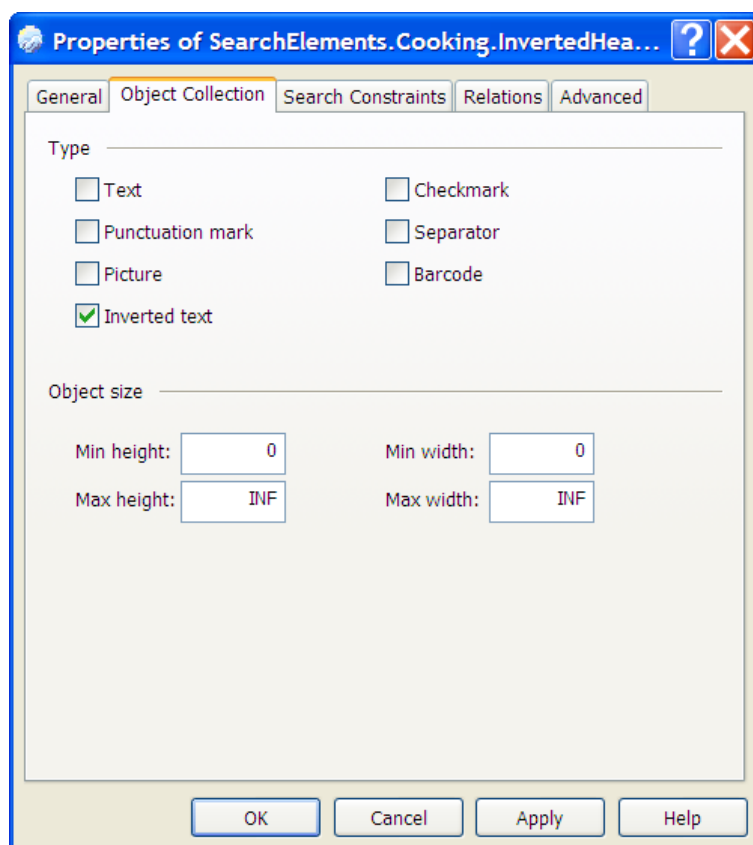


## Step 19. Creating the InvertedHeader element

The title **Cooking** is inverted text (i.e. light text against a dark background).

To create the **InvertedHeader** element:

1. Create an element of the **Object Collection** type and name it **InvertedHeader**.
2. Click on the **Object Collection** tab.
3. Select the **Inverted text** option in the **Type** group.



Try matching the FlexiLayout with the test images and make sure that the program successfully finds the inverted text on all the images.

## Step 20. Describing the Cooking field

In this step you will tell the program how to look for the field under the **Cooking** title, which contains cooking instructions and information about the number of servings.

Note that information about the number of servings is given after the word **Serves** on all the test images. However, on images 2, 3 and 4 the word "serves" is preceded by other words. Describe the field in such a way as to exclude this unwanted text from the block to be recognized.

For easier navigation in the FlexiLayout tree and to reduce the number of hypotheses for the **Cooking** element, create a group element and name it **Instructions**:


1. Within the **Cooking** element, create an element of the **Group** type and name it **Instructions**.
2. Within the **Instructions** element, create an element of the **Static Text** type and name it **Serves**. This element will be used to locate the word "Serves" (see Step 21 for details).
3. Within the **Instructions** element, create an element of the **Character String** type and name it **Servings**. This element will be used to locate the number of servings (see Step 22 for details).
4. Within the **Instructions** element, create an element of the **Text Fragment** type and name it **CookingInstructions**. This element will correspond to the **Cooking** field (see Step 23 for details).
5. Create a block corresponding to the **Cooking** field and name it **CookingInstructions** (see Step 24 for details).

## Step 21. Creating the Serves element

To create the **Serves** element:

1. Create an element of the **Static Text** type and name it **Serves**.
2. Click on the **Static Text** tab.
3. In the **Search text** field, type the text to search: **Serves**.
4. Set Max error percentage to 20.
5. Click on the **Advanced** tab.

6. In the **Advanced pre-search relations** field, specify an additional search constraint: *Look for the object closest to the cooking time.* In the FlexiLayout Language this must be written as follows: **Nearest: SearchElements.Cooking.CookingTimeHeader;**

 **Note:** The same constraint can also be specified via the program's graphical user interface. Click the **Relations**, select **Nearest**, and in the **To:** drop-down list select **Element** and specify **CookingTimeHeader** as the **Reference element**.

## Step 22. Creating the Servings element

To create the **Servings** element:

1. Create an element of the **Character String** type and name it **Servings**.
2. Click on the **Character String** tab.
3. Set the required alphabet. Judging by the test images, the alphabet includes all the digits and the letters "T", "t", "O", and "o".
4. Set the Percentage of alphabet characters to 100.
5. Clear the Allow embedded hypotheses box.
6. Set the Percentage of non-alphabet characters to 20.
7. In the **String length** field, specify this fuzzy range {0, 1, 10, 15}. This interval is an estimation of the length of the character string. We assume that the number of characters may be from 1 to 10. Any hypotheses outside this range will be penalized.
8. In the **total length of spaces** field, specify the fuzzy range that estimates the total length of all the spaces between the characters in the hypothesis (by default measured in dots, 1 dot = 1/300 inch). Do not change the default range of {-1, -1, INF, INF}. This means that there are no constraints on the number of spaces in the hypothesis.
9. Click on the **Relations** tab.
10. Since the number of servings is always located to the right of the word "Serves" and the distance between the right edge of the word "Serves" and the right edge of the number cannot exceed 200 dots, specify the following search constraints:
  - **Rightof** the **Serves** element, **Offset** = 0
  - **Leftof** the **Serves** element, **Offset** = -200, **Element border** = right
11. Since the number of servings is always written on the same level with the word **Serves**, add the following search constraints:
  - **Above** the **Serves** element, **Offset** = -50
  - **Below** the **Serves** element, **Offset** = -50

## Step 23. Describing the Cooking Instructions field

To create the **CookingInstructions** element:

1. Create an element of the **Text Fragment** type and name it **CookingInstructions**.
2. Click on the **Relations** tab.
3. Since the cooking instructions are always located below the title **Cooking** but above the word "Serves", specify the following search constraints:
  - **Below** the **InvertedHeader** element, **Offset** = 0
  - **Above** the **Serves** element, **Offset** = 0

## Step 24. Creating the CookingInstructions block

To create the **CookingInstructions** block:

1. Click on the **FlexiLayout** tab in the program main window.
2. Select the **Blocks** object in the FlexiLayout tree.
3. Select **Add Block>Text** in the **FlexiLayout** menu (or **New>Text** in the local menu).
4. In the **Properties** dialog box that opens, specify the name of the block in the **Name** field: **CookingInstructions**.
5. To describe the location of the block, select **Region expression**.
6. You must describe the location of the block so that it includes only the cooking instructions and the number of servings. The block must not include unwanted text (i.e. any text that may precede the word "Serves"). Describe the block as a collection of rectangles around the following elements: **CookingInstructions**, **Serves** and **Servings**. This method allows you to specify non-rectangular regions.

In the FlexiLayout Language this must be written as follows:

```
RectArray outputRectArray;
//initializing the variable with the help of an empty constructor
outputRectArray = RectArray();
//adding the rectangles of the hypotheses of three elements
```

```
outputRectArray = RectArray();  
outputRectArray.Add ( SearchElements.Cooking.Intsructions.CookingIntsructions.Rect );  
outputRectArray.Add( SearchElements.Cooking.Intsructions.Serves.Rect );  
outputRectArray.Add( SearchElements.Cooking.Intsructions.Servings.Rect );  
OutputRegion = Region( outputRectArray );
```

## **Step 25.** The FlexiLayout is ready

The resulting FlexiLayout can be saved as an \*.afl file and exported into ABBYY FlexiCapture. For more about exporting FlexiLayouts see Sample 1, Step 21 and Step 22.