


Roxen CMS 5.4

Forms & Response Module Manual

 Roxen Internet Software AB
© 2011 Roxen Internet Software AB.
All rights reserved.

Under the copyright laws, this document may not be copied, in whole or in part, without the written consent of Roxen Internet Software.

Roxen Internet Software
Box 449
SE-581 05 Linköping
Sweden
www.roxen.com

Your rights to the software are governed by the accompanying software license agreement.

Every effort has been made to ensure that the information in this document is accurate. Roxen Internet Software is not responsible for printing or clerical errors.

Other company and product names mentioned herein are trademarks of their respective companies. Roxen Internet Software assumes no responsibility with regard to the performance or use of these products.

Contents

1	Forms & Response Module	4
1.1	Features	4
1.2	Creating a new form	4
1.3	Viewing and exporting responses	5
1.4	Deleting responses	6
2	RXML scripting	7
2.1	RXML handles	7
2.2	Form page numbers	8
2.3	Validation	8
2.4	Skipping form pages	8
2.5	Calculations	9
2.6	Custom e-mail	10
2.7	Accessing the response database	11

1 Forms & Response Module

The Forms and Response module makes it possible to easily setup and maintain surveys, questionnaires, polls, application forms, order forms, and more.

Note!

If you're setting up Forms and Response on a multiserver setup you need to add the *CMS Editor Component: F&R Submit* module on the frontend servers.

1.1 Features

- Text fields
- Single and multiple choice selections: radio buttons, drop down lists and check boxes
- Mix with other components, such as Text & Picture
- Multiple pages with page breaks
- Require answer
- Default answer
- Component based, easily edited from the Insite Editor
- Responses are stored in an SQL database
- E-mail responses
- RXML scripting:
 - Dynamically fill in default values
 - Flow control – disable questions and skip pages
 - Custom validation
 - Custom e-mail
 - Calculations
- View list of responses and single responses
- Export responses to other applications such as Microsoft Excel
- Mark responses as deleted in the database, purge later
- Share responses between Edit and Acceleration servers

1.2 Creating a new form

Edit the page where you want to have the form. Insert components such as "Form: Text" and "Form: Selection" in the order you would like them to appear. Other components, such as "Text & Picture", can be used on the same page as well. To split a form into multiple pages, insert "Form: Page break" components. Finish the form by inserting the "Form: Submit" component.

Components inserted after the "Form: Submit" component will be displayed after the form has been submitted. No "Form" components are allowed after "Form: Submit",

except "Form: View" which will display the submitted response. Please note that "Form: View" cannot be used before the "Form: Submit" component.

The screenshot shows a page editor interface for a contact form. On the left, a vertical toolbar contains several components: Header, Form: Selection, Form: Text, Form: Text, Form: Text, Form: Submit, Text & Picture, Footer, and another Form: Text. Each component is preceded by an 'Insert' button. The main content area displays the rendered form. At the top is the 'ROXEN' logo. Below it is the heading 'Contact us'. The form contains a dropdown menu labeled 'Please select subject', three text input fields labeled 'Full name', 'E-mail', and 'Message', and a 'Submit Query' button. Below the form, there is a 'Thank you for your message!' message and a timestamp 'Administrator 15 April 2005, Friday 09:55'.

1.3 Viewing and exporting responses

Edit the page where you want to view the responses. The page must not contain a form page. Insert the "Form: View" component. The "Form link" setting is the path to the original form page. You can have several "Form: View" components on the same page. To export the results to a table format compatible with applications such as Microsoft Excel, use the "export" variant in the component.

To view single responses, exit the page editor and click on the links in the form view list.


The screenshot shows the page editor interface for the 'Responses' view of the contact form. The main content area displays the heading 'Responses' followed by a list of three responses, each with a timestamp and the name 'Administrator':

- [15 April 2005, Friday 13:29](#) (Administrator)
- [15 April 2005, Friday 13:28](#) (Administrator)
- [15 April 2005, Friday 13:20](#) (Administrator)

Below the list is a 'Contact us' link with a small icon. At the bottom, there is a timestamp 'Administrator 15 April 2005, Friday 13:29'. The left toolbar shows components: Header, Form: View, Form: View, Footer, and another Form: View, each with an 'Insert' button.

1.4 Deleting responses

Delete a response by viewing it and clicking on the link "[Delete this response]". Write access to the forms page is required. Deleted responses are only marked as deleted in the database. A response can be undeleted by removing the corresponding delete mark in the database manually. Deleted responses can be permanently purged by clicking on the "Purge" button in the "F&R: Submit" module in the Roxen Administration Interface.

Print page

Responses

[\[Delete this response\]](#)

- Services

Full name

Kalle Anka

E-mail

kalle.anka@roxen.com

Message

Can you update my web page, please?

Administrator 15 April 2005, Friday 13:52

2 RXML scripting

The RXML component can be inserted on form pages to perform various tasks such as:

- Dynamically fill in default values
- Flow control – disable questions and skip form pages
- Custom validation
- Custom e-mail
- Calculations

2.1 RXML handles

The "Form: Text" and "Form: Selection" components have a setting called "RXML handle". It is useful to create simple and easy-to-remember names of form variables. For instance, assume we use "rxmlhandle" as an RXML handle for one of our components. The following entities will then be available:

&form.rxmlhandle;

Contains the submitted response

&var.default-rxmlhandle;

Setting the entity sets the default value

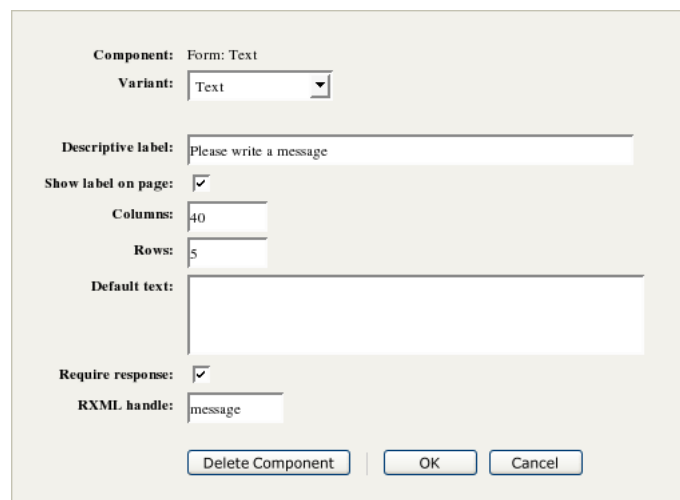
&var.disable-rxmlhandle;

Setting the entity (to any value) disables the field

&var.hide-rxmlhandle;

Setting the entity (to any value) hides the field

In the example below, we insert a "Form: Text" component and set "message" as the RXML handle. We will use this name to reference this component later on.



The screenshot shows a configuration dialog for a "Form: Text" component. The "Component" is set to "Form: Text" and the "Variant" is "Text". The "Descriptive label" is "Please write a message". The "Show label on page" checkbox is checked. The "Columns" are set to 40 and "Rows" to 5. The "Default text" field is empty. The "Require response" checkbox is checked. The "RXML handle" is set to "message". At the bottom, there are buttons for "Delete Component", "OK", and "Cancel".

Next, we insert the RXML component above the "Form: Text" component and have the default message be the current date as shown below.

2.2 Form page numbers

&var.form-display-page;

The currently displayed form page number

&var.form-display-page-total;

The total form page count

2.3 Validation

&var.form-invalid;

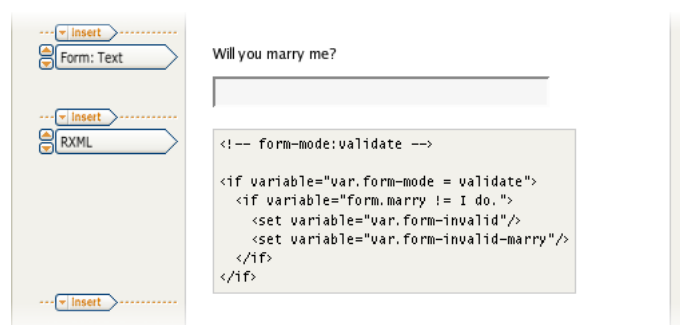
Settings this entity indicates that the form does not validate properly; the entity must be set in combination with the entities for individual RXML handles (see below) and it must be set in the validate form-mode

&var.form-invalid-rxmlhandle;

Settings this entity indicates that a field with an RXML handle called rxmlhandle did not validate properly; the entity must be set in the validate form-mode

In the example below, the "Form: Text" component has an RXML handle called "marry". The RXML component will only validate if the text form contains exactly "I do.". Several important concepts about validation are illustrated:

- The RXML must be evaluated in a special mode because normally RXML is only evaluated for form pages that are to be displayed. This can be done by declaring the string "form-mode:validate" somewhere, for example in a comment as show below.
- Declaring a "form-mode" only adds a mode; the RXML is still evaluated when displayed. Therefore we need to test that `&var.form-mode;` is set to "validate" to make sure that we're only running the code when validating.
- Setting `&var.form-invalid;` indicates that the form did not validate. Setting `&var.form-invalid-marry;` indicates that the component with the RXML handle "marry" did not validate. Several components can be marked this way.
- The placement of the RXML component is not important as long as it is on the same form page.



2.4 Skipping form pages

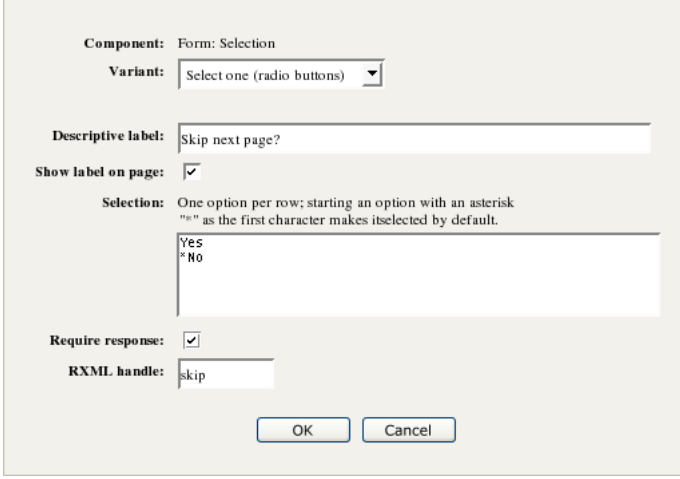
&var.form-skip-page;

Setting the entity skips current form page; it must be set in the skip-page form-mode


Note!

The "Previous" button will not work across form pages that have been skipped.

In the example below, we create a radio button selection. By selecting "Yes", the next page is skipped.



The screenshot shows a configuration window for a 'Form: Selection' component. The 'Variant' is set to 'Select one (radio buttons)'. The 'Descriptive label' is 'Skip next page?'. The 'Show label on page' checkbox is checked. The 'Selection' options are 'Yes' and '*No', with the asterisk indicating that 'No' is the default selection. The 'Require response' checkbox is checked. The 'RXML handle' is set to 'skip'. There are 'OK' and 'Cancel' buttons at the bottom.



The screenshot shows a form preview. On the left is a component palette with 'Form: Selection', 'Form: Page break', 'RXML', and 'Text & Picture'. The main preview area shows the 'Skip next page?' question with radio buttons for 'Yes' and 'No'. Below the question are 'Previous' and 'Next' buttons. A code block shows the RXML for the skip logic:

```
<!-- form-mode:skip-page -->
<if variable="var.form-mode = skip-page">
  <if variable="form.skip = 1:*">
    <set variable="var.form-skip-page"/>
  </if>
</if>
```

Below the code block, there is a note: "If you selected "Yes" on the previous page, this text will not be displayed."

2.5 Calculations

In the example below, we have created a selection with the RXML handle "items". On the next form page, the RXML component uses this handle to calculate the number of checked items.

Component: Form: Selection

Variant: Select many (check boxes)

Descriptive label: Select items of interest

Show label on page:

Selection: One option per row; starting an option with an asterisk "*" as the first character makes it selected by default.

A
B
C
D
E

Require response:

RXML handle: items

Delete Component | OK | Cancel

Form: Selection

Select items of interest

A

B

C

D

E

Previous Next

```
<set variable="var.item-count" value="0"/>
<if variable="form.items">
  <emit source="values" variable="form.items">
    <inc variable="var.item-count"/>
  </emit>
</if>
<p>You have selected &var.item-count; item(s).</p>
```

2.6 Custom e-mail

In the example below, we have created an e-mail form. The three text fields have the RXML handles "subject", "text" and "email" respectively. The e-mail tag is used after the "Form: Submit" component.



2.7 Accessing the response database

The name of the database used by Forms & Response is stored in the entity `&forms-and-response.db-name;`. Two tables are used:

form_submission

Contains one row per submitted response.

form_item

Contains one row per given answer; multiple choice responses may require several rows

Accessing the database can be useful on form pages following the "Form: Submit" component. All form variables are automatically emulated in the RXML form scope after submission (please see the "Custom e-mail" example). In addition, two form variables are provided after the form submission:

&form.form-id;

Contains the id in the `form_submission` table. Note! Use the `&form.form-id;` in combination with the `&form.form-session;` variable to make sure that the id cannot be faked.

&form.form-session;

Contains the session string in the `form_submission` table. It is a random string of digits to prevent faked `&form.form-id;` variables.