Part II: Comprehensive Sample PDF Form using RenderX

John/Jane Doe, welcome to part II of your demonstration. This is a sample survey form created using **RenderX's PDF Checklist *CoolTool***. It shows the variety of things you can easily do with this technology. Form objects such as checkboxes, textboxes, multilines, option groups and comboboxes are all supported through specifying simple XML structures. The Checklist document structure supports sections, categories and instruction areas, and also has an easy customization XSL where all appearance properties are specified. It also has a default JavaScript library that you can extend with custom functions.

You likely have already completed the first part of this demonstration, filling out and submitting our survey form. Now, this is a sample form showing the different elements you can use. As part of the submit process, RenderX's PDF Forms Processing Server software has auto-generated this XML file with your projectID and your email as the form adminstrator. These two items are attributes on this XMLs <form> element. We've also added a set of buttons to show the other capabilities available to you.

Section 1 - Form Elements

Category 1 - Simple Checkboxes

Group Heading

- ☑ Mandatory checkbox ("value" and "readonly" are true in the source)
- ☐ Optional checkbox ("value" is false in the source)
- ☐ Even checkboxes included inline ☐ with the content of an item. This example also shows the checklist JavaScript implementation. The two checkboxes in this <item> are linked and always have the same state.

Category 2 - Textboxes

- ☐ Textbox inline ("length" set to 1in) [ ] where the state of the checklist checkbox is controlled by the content (see the JavaScript discussions later).
- ☑ Textbox inline ("length" set to 1in) with some [default text], followed by a textbox that is a complete line (no length specified), followed by a multiline textbox. The multiline textbox has a height of 1in in the fillable form and has six print lines in the blank form.
- ☐ [ ]
- ☐ [ ]

Category 3 - Option Groups

- ☑ Option Group (Choice 2 default, horizontal layout), since the question is already answered the checklist checkbox is checked and readonly.
  ☐ Choice 1 ☑ Choice 2 ☐ Choice 3 ☐ Choice 4

☐ Option Group (no choice, vertical layout), since nothing is answered yet, the checklist checkbox is not yet checked. It will be checked on selecting an item in the option group.

- ☐ Choice 1
- ☐ Choice 2
- ☐ Choice 3
- ☐ Choice 4

Category 4 - Combobox

☑ Combobox (Choice 2 default), selecting any item other than 'No Choice' will set the checklist item when leaving the field. Choice 2

Section 2 - JavaScript

Category 1 - Creating a checklist

This set of JavaScript functions can be helpful in creating a Checklist. They evaluate form fields for specific content and will automatically check the corresponding checklist box. In order to ensure things are kept in sync, you should set the checklist checkbox initial value to the appropriate value based on your rules and you should also set the "readonly" flag to "true". This ensures that only the JavaScript controls the state of the corresponding checklist checkbox. Most examples in this document use the various JavaScripts available to evaluate the state of checkboxes, textboxes, multilines, comboboxes and optiongroups.

☐ Autocheck for textboxes like . It will automatically set the checklist state based on the text length (checked if greater than "0").

☐ Autocheck for optiongroups without an exclude item will automatically set the checklist state when an option is specified.

☐ Choice 1 ☐ Choice 2 ☐ Choice 3 ☐ Choice 4

☐ Autocheck for option groups with an exclude item, selecting any item other than the excluded one will set the checklist state. You specify the option that is the no selection item, in this case it is 'q1_5' which is 'No Answer'.

- ☐ Answer One
- ☐ Answer Two
- ☐ Answer Three
- ☐ Answer Four
- ☐ No Answer

☐ Autocheck for comboboxes, selecting any item other than the excluded one will set the checklist state when leaving the field. You specify the string that is the no selection item, in this case it is 'No Choice'. No Choice

Category 2 - Numeric Validation

☐ Ensure integer numbers only in textboxes like . This script checks every keystroke and only allows integer numeric input.

☐ Ensure numbers only in textboxes like _____. This script checks every keystroke and only allows numeric input.

Category 3 - Range Checking

☑ Ensures the number entered is within a range (use "false" to exclude the endpoints, "true" to include them) 3.3. This script checks the value entered is between (not including) two numbers. In this example the numbers are 2.4 and 6.7

☐ Ensures the number entered is within a range (use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is between (and including) two numbers. In this example the numbers are 2.4 and 6.7

☐ Ensures the number entered is not within a range (use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is not between (not including) two numbers. In this example the numbers are 2.4 and 6.7

☑ Ensures the number entered is not within a range (use "false" to exclude the endpoints, "true" to include them) 1.3. This script checks the value entered is not between (and including) two numbers. In this example the numbers are 2.4 and 6.7

☐ Ensures the number entered is greater than or less than a number (use 'greater' for greater and 'less' for less than. Use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is greater than or equal to 2.4

☐ Ensures the number entered is greater than or less than a number (use 'greater' for greater and 'less' for less than. Use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is greater than 2.4

☐ Ensures the number entered is greater than or less than a number (use 'greater' for greater and 'less' for less than. Use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is less than or equal to 2.4

☐ Ensures the number entered is greater than or less than a number (use 'greater' for greater and 'less' for less than. Use "false" to exclude the endpoints, "true" to include them) _____. This script checks the value entered is less than 2.4

Section 3 - Hidden Fields

The form fields you see above are not the only fields in this PDF. You can also create hidden fields in the document that can be used for further processing. For the most part, this is used to link the submitted information in the form with a specific user, or at least to ensure it's uniqueness if you are generating unique forms to be submitted by individuals. Hidden fields can contain any string of information you desire (name, account number, email, account balance, etc.). You may also choose to represent the information in the actual print form itself or not, this is of course up to you as the form designer.

The style sheets provided will automatically create a hidden form for any XML element placed in the <recipient> tag that has an attribute "hidden" set to "true". This example has four hidden fields -- name, email and two custom fields. You can create as many as you desire. The style sheets also allow an easy way to insert content from these fields anywhere in the document as text should you choose to use them.

You use a <link> tag with a "nameref" attribute set to the same name as the hidden field name. For example, a link to the hidden name field in this document is:

<link nameref="recipientname"/> which results in: John/Jane Doe

Hidden fields are a very important part of a form design and as you can see they are fully supported by this application.

Section 4 - Form Actions

Form actions

Form actions can be specified in the XML file. Each action creates a button that is located wherever you place the <buttons> tag. This tag also supports attributes which control their placement. The attribute "button_position" can be "top", "bottom" or "inline" and the "button_align" can be "left", "right" or "center". There are several types of actions supported, set by specifying one of the following in the "type" attribute.

Specify "**submit**" to send the form data to the server for further processing. The form data is automatically submitted to the action's "url" property in XFDF format (Adobe's XML format for form fields). Of course you can specify your own "url". RenderX provides some standard ones documented below with our PDF Forms Processing Server.

Specify "**email**" to create an email of the form data to be submitted to the form administrator via the user's default email application. This style sheet defaults to using the XFDF format.

Specify "**reset**" to create a form reset button that will return the whole form back to its original state.

RenderX PDF Forms Processing Server URLs for Submit

RenderX provides several different URLs should you choose to use our PDF Forms Processing Server. These are specified on the "url" attribute of the <action> element. All of the examples are in this sample XML document. Essentially, these submit url's are computer programs that accept the form submittal and perform a pre-determined application. Each of these are described in more detail below.

Echo is used to send the current selections in the form and echo the form submittal results back to submitter. The user will see that their form has been submitted and you as the forms administrator will receive an email with their responses. The URL to specify on the <action> is:

http://www.foactive.com/Forms/Service.PDF.Echo.aspx

Burn is used to "burn in" the current selections in the form and send this fillable form back to the user. This allows them to locally save the form with their selections or even forward along to another user as part of a workflow. You as the forms administrator will receive an email with their responses. The URL to specify on the <action> is:

http://www.foactive.com/Forms/Service.PDF.Burn.aspx

Flatten is used to generate a non-fillable PDF document with all their current selections in place and send back to the user. Questions that are not answered are also presented. You as the forms administrator will receive an email with their responses. The URL to specify on the <action> is:

http://www.foactive.com/Forms/Service.PDF.Flatten.aspx

Compact is similar to flatten but works in conjunction with the checklist functionality. If you use the structure in this document with <item> containing a <checkbox> as it's first child, this function when called will flatten the form but compact it so only those questions answered are included in the PDF document. Unanswered

questions (without checklist checks) will be omitted from the results. You as the forms administrator will receive an email with their responses. The URL to specify on the <action> is:

http://www.foactive.com/Forms/Service.PDF.Compact.aspx

Blank is a way of representing the form on a blank piece of paper as if you need someone to fill it out offline by hand. No form values are presented on the printed page. Textboxes and multilines are replaced by underlines for data entry; comboboxes are replaced by a list of checkboxes. You as the forms administrator will receive an email with whatever was in the form fields at that time. The URL to specify on the <action> is:

http://www.foactive.com/Forms/Service.PDF.Blank.aspx

Section 5 - Additional Information

Getting Started

So you have all the elements you need for a complete RenderX PDF forms solution. This XML file and associated style sheet are merely examples but should be sufficient for most to complete even a very complex project. You do need to make sure you set your projectID and form administrator attributes to the proper values. You should have been assigned a projectID from RenderX, if not feel free to contact us for one.

You can use this XML as a guide or start from scratch. When you have designed and tested all your elements locally, you **should contact us to publish your XSLs** and JavaScript library to our server **if you have made any changes to the base XSLs** we have sent you. Your project directory already has all of the initial files installed so you can start immediately. In the future we will provide a capability for you to upload your own but for now we have chosen to publish them ourselves. That's all you need really. You can generate and send custom, user specific forms or even post a blank form to any website. If you use our PDF Forms Processing Server, it provides the default functions shown in this document and will email you as the form administrator with each form submittal.

You can create any type of document like a survey, a course evaluation or even training course tests. Individual responses are sent to your email whenever a submit is made. As this is a demonstration system, we'll do our best to keep it all oiled and working but remember its for demonstration. If you want to purchase a production-level installation, contact us.

Technical Details for the Tech Savvy Folks

All of this is done with pure standards with some extensions in RenderX's implementation of XSL FO for making PDF forms and allowing JavaScript in PDF. Only the PDF Forms Processing Server has some custom code to process and handle results but even then, there is a lot of XML and XSL going on. If you look deeper into the PDF form XSLs provided, you will see a little magic going on.

One of those "magic" areas is that the original XML that generates the form is embedded in the PDF document. Why? Well, that XML file is extracted by the PDF Forms Processing Server and the values in the XML file are replaced by the values submitted in the form response. This matching is done by using the form field "name" attribute so it is **important** to give a unique name to your fields. The style sheet will do this with *generate-id()* if you do not specify your own field name. This allows you to create forms without specifying names, but only *for testing purposes*. If you ever want to use the data you are recieving from the form submittal, you need to know what that data represents. Identifying your form fields with known name identifiers is the proper method.

You should not (for obvious reasons) change the areas of the provided stylesheets that send the XML file, projectID or the form adminstrator back to the server or you will be unable to process your forms on the server.

Custom PDF Forms Proccessing Server plugins are used to do things like send emails or even perform operations like cutting a license key. The default PDF Forms Processing Server takes the original XML file embedded in the form, replaces the data with that of the form submission, uses your projectID to lookup your XSL, and optionally re-run RenderX setting the proper parameter for the desired output. It delivers back the form echo results or the resulting PDF, streamed to the user's browser. Nothing is written to our system at all, all responses are immediately processed in memory and sent out.

The package includes the XML ("*sample.xml*") that produces this form and some other associated files. "*checklist.xsl*" is the main XSL that processes the XML file. "*customize.xsl*" contains many of the style parameters that control various formatting options for both the content as well as the PDF form fields. "*xml-to-string.xsl*" is an included stylesheet that converts the source XML into a string that can be embedded into one (or more) form fields. There is a limitation in Adobe PDF form fields that allow only about 30,000 characters in a field so this string is chopped into multiple fields if necessary. "*header-footer.xsl*" just separates the header and footer templates from the main form template for easier editing of these frequently changed templates.

"*RXjslib.js*" is a small XML file with the general JavaScript library. This file is inserted into the form and was put into a separate file so that you can use context-sensitive JavaScript editors to assist you in writing your own JavaScript functions.

Good luck on your projects and of course we at RenderX are always available to help you. Visit RenderX's Website for more information.