

How to integrate Changeforms into an Application Server

This document describes one possible strategy for using the changeforms functionality within a separate Web applications server infrastructure.

Purpose

The purpose of the example integration is to generate PDF documents containing completed forms. In the most complex case this process might require several intermediate steps in order to (1) create a “prepopulated” form from a generic document and (2) create a final document using values filled in by the application client. The following table lists the steps anticipated in this document A through H and indicate whether the action is completed by the outside client, the application server or the changeforms components.

Step	Outside	App Server	Changeforms
A) request		Application server formulates an XML message describing what fields to populate and what fields to mark read only. Message is sent to Changeforms with the location of the source PDF file and the callback URL.	
B) prepopulation			Changeforms rewrites the PDF contents from the source, modifying the appropriate PDF fields and stores the result in a temporary file location.
C) Prepopulated form is transferred		Application server sends the prepopulated form to the client	
D) Client fills prepopulated form	In Acrobat reader the client fills the form and submits it, resulting in a post request.		

E) POST to XML translation.		Application server receives the post request from the client and constructs an XML message which describes the fully filled and frozen form.	
F) Document Freeze			Changeforms receives the XML and constructs the frozen PDF document, storing it in a temporary location.
G) Archive frozen document		Application server archives the frozen PDF document and transfers it back to the client.	
H) download	Client (optionally) downloads the frozen form		

For some document types the prepopulation step may not be required, in which case the steps A and B may be omitted and the C step can deliver the source unmodified source document.

For the purposes of this document it suffices to discuss the step combinations A/B and E/F since the other steps pertain to the application server and the client and do not relate to the changeforms functionality.

XML for changing a PDF document containing a form

The XML markup dialect described by `changeforms/formdocument.dtd` provides a mechanism for describing changes to AcroForms stored inside XML documents and for specifying how those forms should be changed.

Dumping form information

The changeforms program provides a mechanism for extracting a description of the Acroform in a PDF document using the command line

```
changeforms.py --dumpxml fromdocument.pdf > xmlparameters.xml
```

For example the command

```
changeform.py --dumpxml formtest.pdf > test.xml
```

Yields the file test.xml with contents

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">

<formDocument filename="formtest.pdf"

    frozen="0">

    <fieldInfo name="checkbox" kind="check">
        <value>Yes</value>
        <option>Yes</option>
        <option>Off</option>
    </fieldInfo>

    <fieldInfo name="listbox" kind="choice">
        <value>item3</value>
        <option>item1</option>
        <option>item2</option>
        <option>item3</option>
    </fieldInfo>

    <fieldInfo name="radio" kind="radio">
        <value>radio 1</value>
        <option>radio 1</option>
        <option>radio 2</option>
        <option>radio 3</option>
    </fieldInfo>

    <fieldInfo name="submit" kind="push">
        <url>http://localhost/cgi-bin/test.cgi</url>
    </fieldInfo>

    <fieldInfo name="text" kind="text">
        <value>this is the default</value>
    </fieldInfo>

</formDocument>
```

Developers may use the “—dumpxml” feature both to determine the contents of an acrobat form and to test the results.

Generating an altered form in a new document

In both steps A and E above the Application server software creates an XML description used to generate a new document containing an altered acrobat form. Use changeform to process the source document and the XML description to generate a new document using the command line:

```
changeforms.py --rewrite fromdocument.pdf todocument.pdf parameters.xml
```

For example if the file test.xml contains the form description

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">

<formDocument filename="formtest.pdf"
  frozen="1">

  <fieldInfo name="radio">
    <value>radio 2</value>
  </fieldInfo>

  <fieldInfo name="submit">
    <url>http://finance.yahoo.com/index.html</url>
  </fieldInfo>
</formDocument>
```

A new PDF file f.pdf may be generated using the command line

```
changeforms.py --rewrite formtest.pdf f.pdf test1.xml
```

In this case the f.pdf document will be “frozen” since the XML specifies `frozen="1"` and the value of the “radio” field will be “radio 2” instead of “radio 1”.

XML Form Specification

An XML form specification for modifying a form in a source file is “similar” to an XML dump of the form in a source file, except:

1. Only fields that require modification need be listed (each once only).
2. The options may be omitted.
3. Field kinds may be omitted.
4. Values may be omitted if they are not altered.
5. The “frozen” attribute of the formDocument tag must always be present with value either “0” or “1”.

There are four basic ways the applications server software may need to modify acrobat forms.

1. Change the default value for fields.
2. Mark an individual field “read only.”
3. Alter a submit button URL.
4. Freeze the document.

Each of these are discussed below.

How to change the default value for fields

To modify the default value for a field in the XML description, include the field in the sequence of fields and specify the new value using the value tag. If the field is a field with a limited number of option values the new value must be one of the options.

For example the following description specifies that the “radio” field should have the value “radio 2”.

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">

<formDocument filename="formtest.pdf" frozen="0">
  <fieldInfo name="radio">
    <value>radio 2</value>
  </fieldInfo>
</formDocument>
```

In this case if we specified “xxx” for the new value the changeform.py program would terminate in an error condition since “xxx” is not one of the options permissible for the “radio” field.

How to mark an individual field “read only”

To mark a field read only in the XML description, include the field in the sequence of fields and add the “<readonly/>” tag to the field tag contents.

For example the following XML changes the value of the “radio” field as above and also requests that the “text” field be marked readonly.

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">

<formDocument filename="formtest.pdf" frozen="0">
  <fieldInfo name="text">
    <readonly/>
  </fieldInfo>
  <fieldInfo name="radio">
    <value>radio 2</value>
  </fieldInfo>
</formDocument>
```

As shown any number of fields may be modified in one XML description.

How to alter a submit button URL

To alter the URL of a submit button, include the submit button field in the sequence of fields and add the “<url>” tag to the field tag contents with the new URL.

For example the following XML changes the value of the “radio” field as above, requests that the “text” field be marked readonly, and modifies the URL of the “submit” button to <http://www.sun.com/go.cgi>.

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">
```

```
<formDocument filename="formtest.pdf" frozen="0">
  <fieldInfo name="submit" kind="push">
    <url>http://www.sun.com/go.cgi</url>
  </fieldInfo>
  <fieldInfo name="text">
    <readonly/>
  </fieldInfo>
  <fieldInfo name="radio">
    <value>radio 2</value>
  </fieldInfo>
</formDocument>
```

Note that URLs for submit buttons should generally be absolute URLs because the PDF form has no HTTP base context (unlike HTML documents in a browser).

How to freeze the document

To freeze a document specify frozen="1" in the top level tag. For example the following XML changes the value of the "radio" field as above, changes the value of the "text" field to "garbanzo beans", and freezes the resulting document.

```
<!DOCTYPE formDocument SYSTEM "formdocument.dtd">

<formDocument filename="formtest.pdf" frozen="1">
  <fieldInfo name="text">
    <value>garbanzo beans</value>
  </fieldInfo>
  <fieldInfo name="radio">
    <value>radio 2</value>
  </fieldInfo>
</formDocument>
```

Frozen documents will be encrypted with "no password" to ensure that no further modifications to the document or to the form inside the document are permitted.