



PDF SHARE FORMS

Online, Offline, OnDemand

PDF forms and SharePoint are better together

Repeatable section synchronization

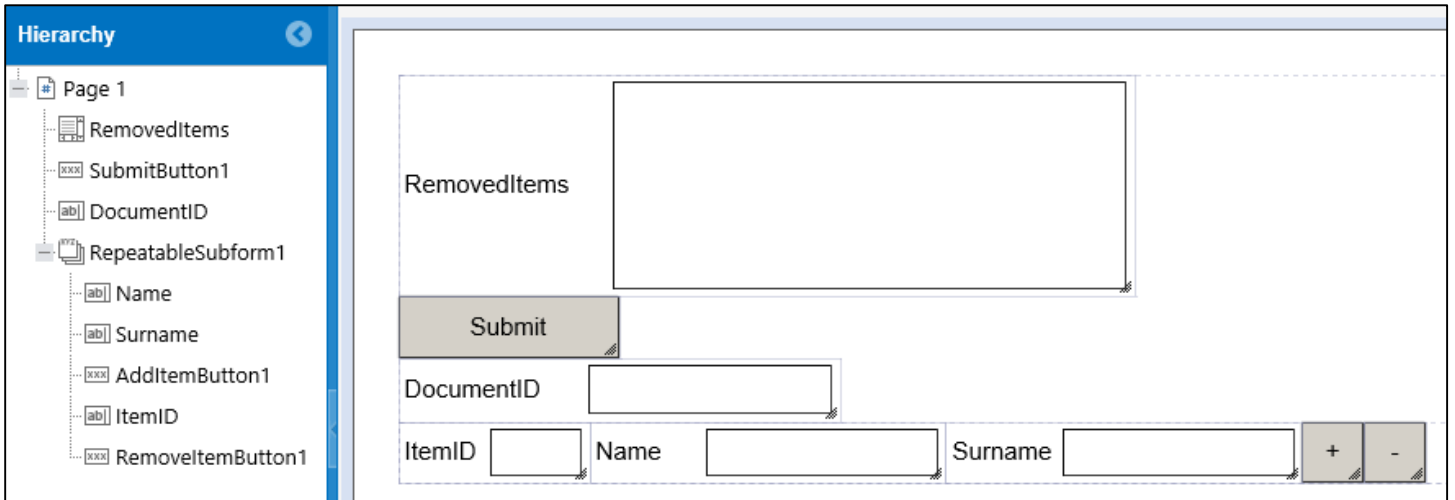
Contents

Template	2
Runtime.....	5
Form Submit script.....	9
Form Load script	10

This guide describes how to synchronize form that contains repeatable sections with SharePoint list.

Template

Step 1. Prepare template – Dynamic (XFA) form



The screenshot shows a PDF form template design tool. On the left is a 'Hierarchy' pane with a tree view containing the following elements: Page 1, RemovedItems, SubmitButton1, DocumentID, RepeatablSubform1, Name, Surname, AddItemButton1, ItemID, and RemoveItemButton1. The main area displays a visual representation of the form. It includes a large multiline text field labeled 'RemovedItems', a 'Submit' button, a single-line text field labeled 'DocumentID', and a repeatable subform section. The subform section contains three single-line text fields labeled 'ItemID', 'Name', and 'Surname', followed by '+' and '-' buttons for adding and removing items.

This template has multiline text field, submit button, text field and repeatable subform with three text fields in it and add/remove buttons:

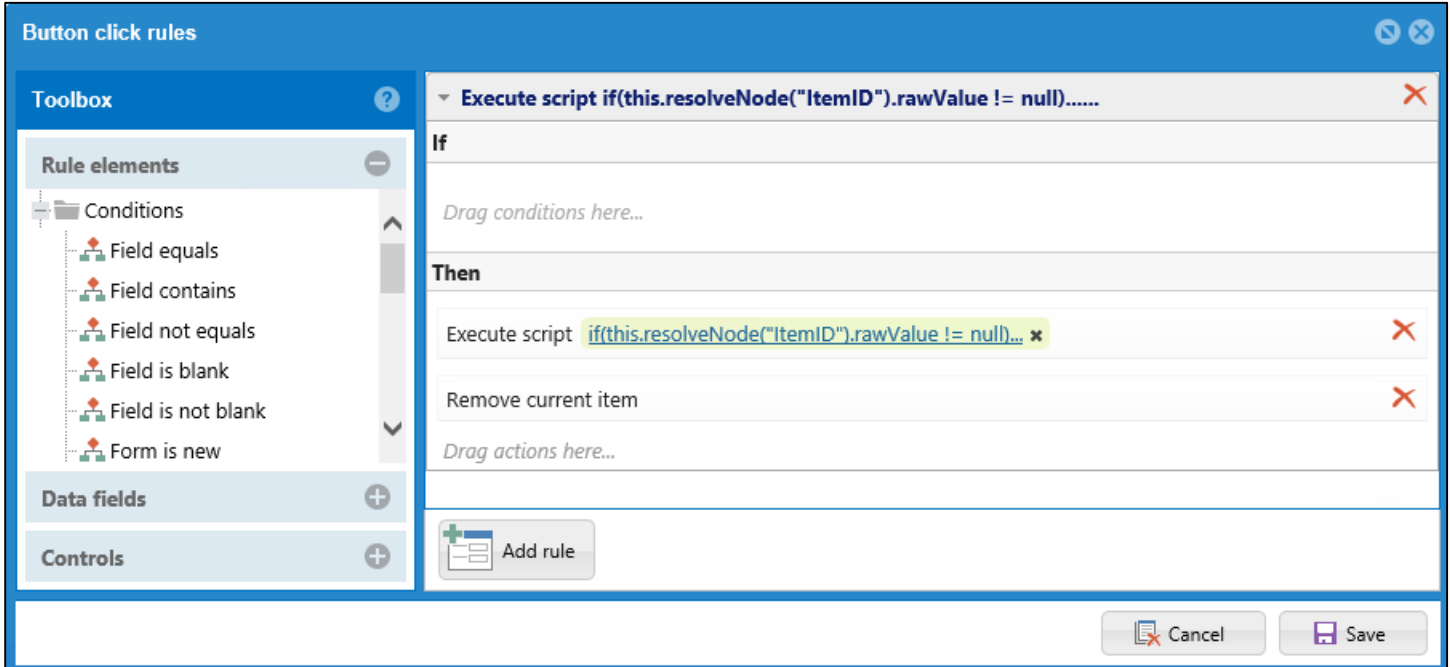
RemovedItems – this MultilineTextField is used to store IDs of deleted items, should be set to hidden;

DocumentID – to store document's ID;

RepeatablSubform1:

- ItemID – store list item id, should be set to hidden;
- Name and Surname – store information from list/save information from form to list;
- +/- buttons – add new repeatable node.

Step 2. Add logic to the remove item button “–”. Select button and navigate to **Properties → Actions**

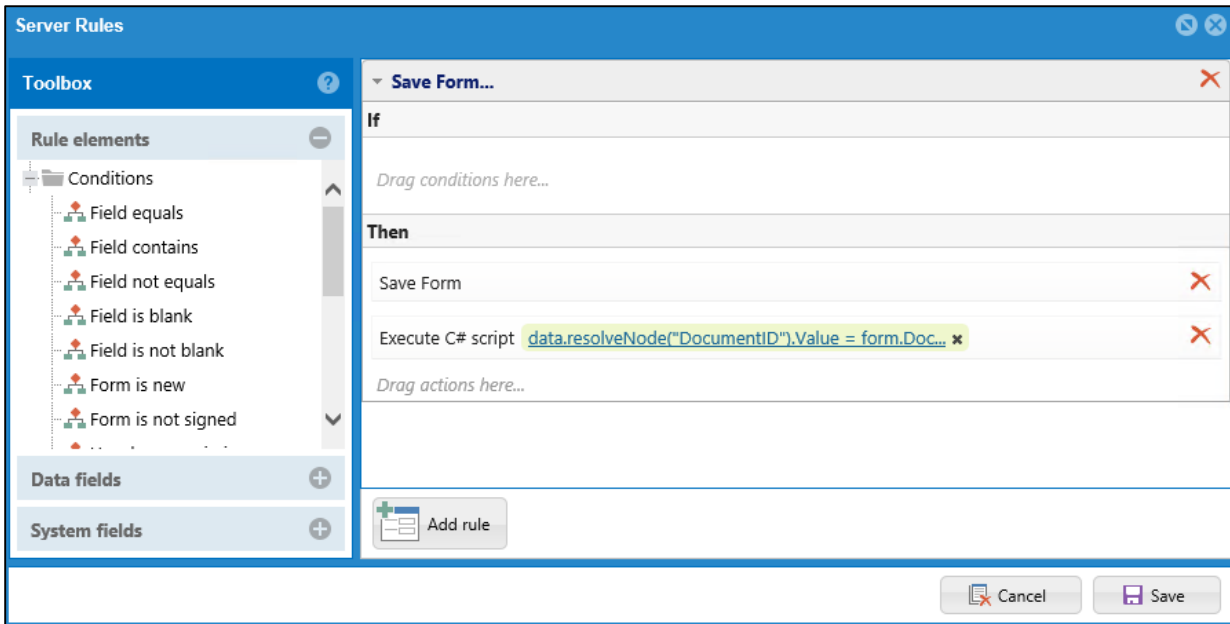


Place **Execute script** before **Remove current item** action. Place this script:

```
if(this.resolveNode("ItemID").rawValue != null){
    if(xfa.resolveNode("$data..RemovedItems").value == null)
    {
        xfa.resolveNode("$data..RemovedItems").value = this.resolveNode("ItemID").rawValue + ";";
    }
    else
    {
        xfa.resolveNode("$data..RemovedItems").value =
xfa.resolveNode("$data..RemovedItems").value + this.resolveNode("ItemID").rawValue + ";";
    }
}
```

This script will write down ID of items that are deleted.

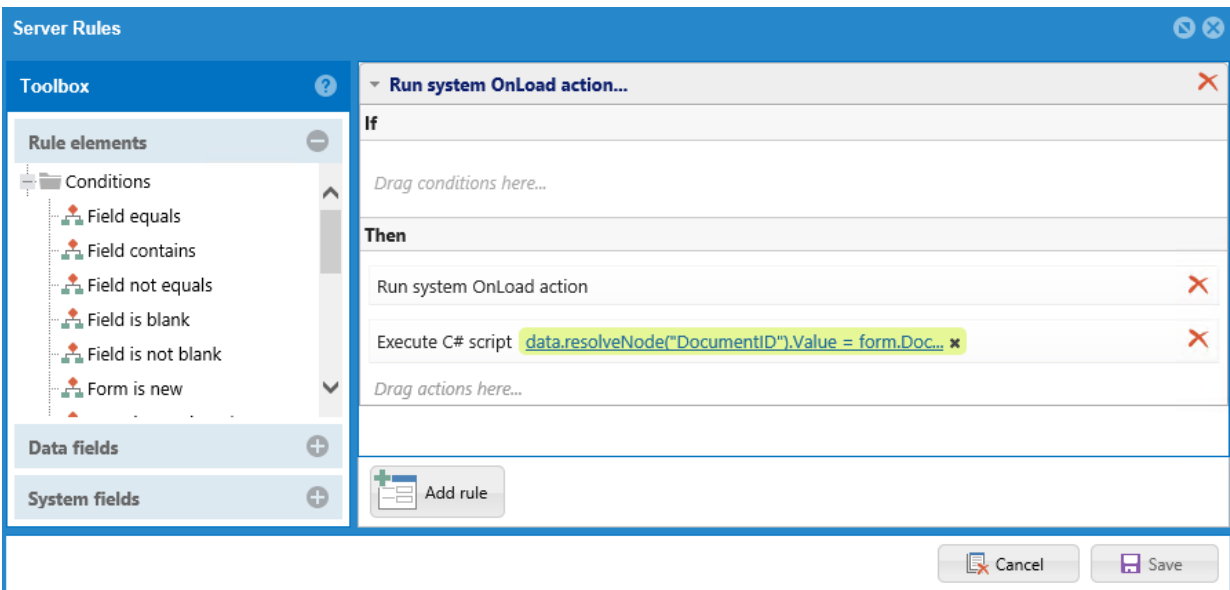
Step 3. Add script to form submit rule. Place **Execute C# script** action under **Save Form** action, navigate to **Developer → Form Submit**



Script

This script will create new items in specified list and update them on form load.

Step 4. Add **Execute C# script** action under **Run system OnLoad** action. Navigate to **Developer → Form Load**



This script will populate nodes with information from list and change information in node if information was changed in list.

Script

Runtime

Step 1. Create new form, fill in and submit

RemovedItems

Submit

DocumentID

ItemID	<input type="text"/>	Name	<input type="text" value="John"/>	Surname	<input type="text" value="Smith"/>	+	-
ItemID	<input type="text"/>	Name	<input type="text" value="Aren"/>	Surname	<input type="text" value="Kit"/>	+	-
ItemID	<input type="text"/>	Name	<input type="text" value="Alice"/>	Surname	<input type="text" value="Torm"/>	+	-

New items will be created in specified list

GuideList

+ [new item](#) or [edit this list](#)

All Items ...

	Name	Surname	DocumentID
✓	John	Smith	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c
	Aren	Kit	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c
	Alice	Torm	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c

Step 2. Re-open submitted form. All nodes contain information about **ItemID**, which they are connected to

RemovedItems

Submit

DocumentID 18d63892-86f5-4584

ItemID	585	Name	John	Surname	Smith	+	-
ItemID	586	Name	Aren	Surname	Kit	+	-
ItemID	587	Name	Alice	Surname	Torm	+	-

Step 3. Click on Remove Item button (n this example near **Aren Kit** node with **586 ItemID**):

RemovedItems 586;

Submit

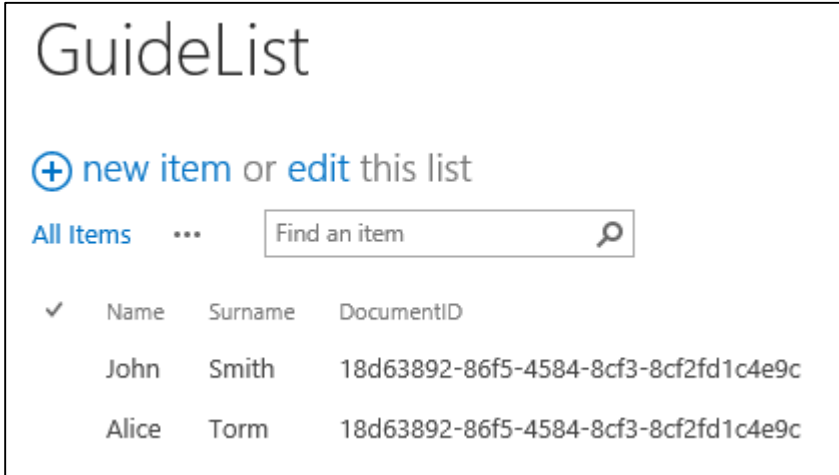
DocumentID 18d63892-86f5-4584

ItemID	585	Name	John	Surname	Smith	+	-
ItemID	587	Name	Alice	Surname	Torm	+	-

RemovedItems field contains ID of item which was deleted.

Click **Submit**.

Aren Kit is removed from the list



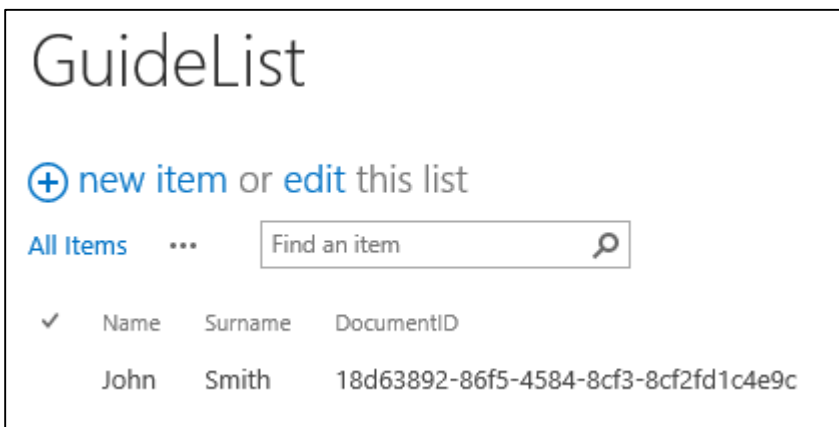
GuideList

+ new item or edit this list

All Items ... Find an item

✓	Name	Surname	DocumentID
	John	Smith	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c
	Alice	Torm	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c

Step 4. Remove item from the list (in this example **John Smith** is removed)



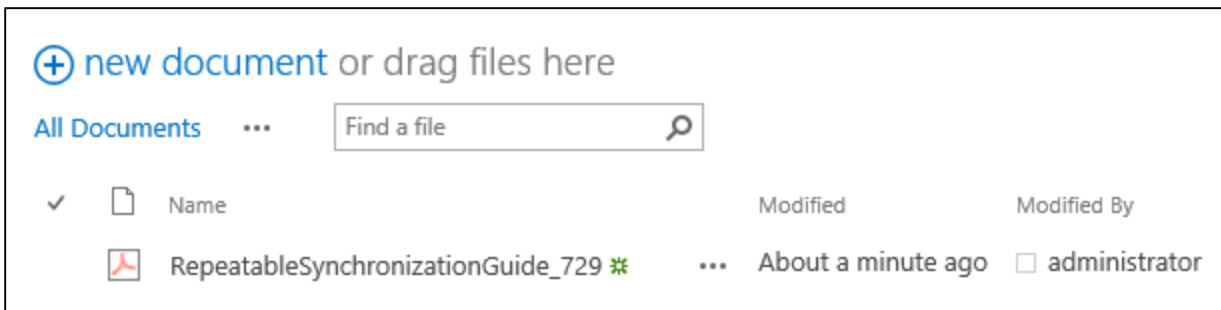
GuideList

+ new item or edit this list

All Items ... Find an item


✓	Name	Surname	DocumentID
	John	Smith	18d63892-86f5-4584-8cf3-8cf2fd1c4e9c

Open document, which was connected with this item



+ new document or drag files here

All Documents ... Find a file

✓	📄	Name	Modified	Modified By
		RepeatableSynchronizationGuide_729 ✳	... About a minute ago	<input type="checkbox"/> administrator



Item, which was deleted in the list, is removed in the form.

RemovedItems

Submit

DocumentID 18d63892-86f5-4584

ItemID 585 Name John Surname Smith + -

Form Submit script

```

data.resolveNode("DocumentID").Value = form.DocumentID.ToString();
SPListItemCollection listItems = currentWeb.Lists["GuideList"].Items;
SPList list = currentWeb.Lists["GuideList"]; //list name is marked with red - it must be specified
                                              for proper case

if(form.IsNew()){
//Add new items to the list
    foreach (XElement node in data.Root.Elements("RepeatableSubform1"))
    {
        SPListItem item = listItems.Add();
        item["Name"] = node.Element("Name").Value;
        item["Surname"] = node.Element("Surname").Value;
        item["DocumentID"] = form.DocumentID.ToString();
        item.Update();
    }
}
else{
    foreach (XElement node in data.Root.Elements("RepeatableSubform1")){

        if(node.Element("ItemID").Value == "")
        {
            //Add new items to the list when form is not new
            SPListItem item = listItems.Add();
            item["Name"] = node.Element("Name").Value;
            item["Surname"] = node.Element("Surname").Value;
            item["DocumentID"] = form.DocumentID.ToString();
            item.Update();
            node.Element("ItemID").Value = item.ID.ToString();
        }
        else
        {
            //Update items in the list
            SPListItem item = listItems.GetItemById(int.Parse(node.Element("ItemID").Value));
            item["Name"] = node.Element("Name").Value;
            item["Surname"] = node.Element("Surname").Value;
            item.Update();
        }
    }
}
//if RemovedItems field contains IDs, on form submit items in the list with these IDs will be
deleted
if(data.resolveNode("RemovedItems").Value != "")
{
    foreach(string s in data.resolveNode("RemovedItems").Value.Split(';'))
    {
        SPListItem item = list.GetItemById(int.Parse(s));
        item.Delete();
    }
}

```

[Back to guide](#)

Form Load script

```
data.resolveNode("RemovedItems").Value = "";
data.resolveNode("DocumentID").Value = form.DocumentID.ToString();
foreach (XElement node in data.Root.Elements("RepeatableSubform1"))
{
    if(node.Element("Name")==null)
    {
        node.Remove();
    }
}

if(!form.IsNew()){
    data.Root.Elements("RepeatableSubform1").Remove();
    SPList list = currentWeb.Lists["GuideList"]; //list name must be specified for proper case
    SPListItemCollection items = list.GetItems("Name", "Surname", "DocumentID");
    foreach(SPListItem item in items){
        if(item["DocumentID"].ToString()==form.DocumentID.ToString()){
            data.Root.Add(
                new XElement("RepeatableSubform1",
                    new XElement("ItemID", item.ID.ToString()),
                    new XElement("Name", item["Name"]),
                    new XElement("Surname", item["Surname"])));
        }
    }
}
```

[Back to guide](#)