

MAX 2007. ENGAGE.

Designing PDF Forms and Flex Form Guides

Level: **Intermediate**

Stefan Cameron

scameron@adobe.com

Computer Scientist

LC Designer ES

Adobe Systems Inc.



- Design a dynamic form to use for registering people to a race:
 - Define a **schema data connection** to ensure that submitted data adheres to schema rules
 - Add **script** to enable multiple registrations per form
 - Use basic dynamic **form design techniques** to ensure form objects flow properly
 - Create a Flash-based **Form Guide Wizard view** for the form to better engage customers online

- Need to collect data which matches specific schema
- Data connections let you **bind** specific fields to specific data nodes in a data structure such as:
 - Schema
 - Database table/query/stored procedure
 - Web service
- LC Designer ES uses rules defined in schema when creating fields/subforms based on a data connection
 - Do this by dragging and dropping from the **Data View palette**
- XML Data submitted to email address or server adheres to the schema
 - Assuming you used the default field/subform configuration obtained by dragging and dropping the entire connection onto your form or that you correctly structured your form objects and configured your bindings.
- Only **one** schema data connection per form is permitted; multiple database (ODBC) and web service data connections permitted



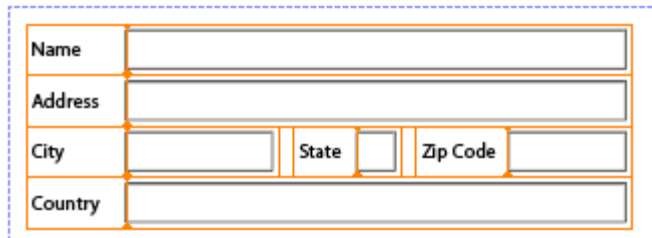
- Define a data connection to a schema in a new form
- Quickly create fields bound to the data connection by dragging and dropping
- Inspect the resulting fields to see how they relate to the schema's types and constraints

Exercise #1: Define Schema Data Connection



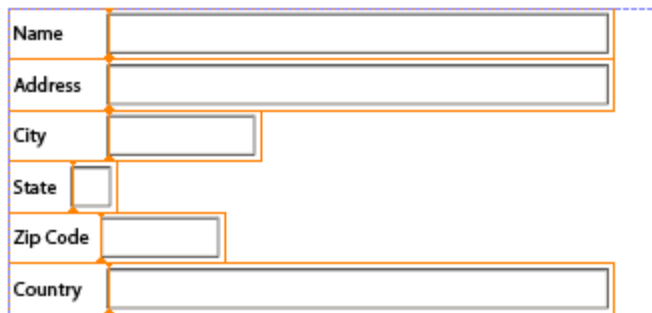
- Define a data connection to a schema and use the drag and drop technique to link existing fields on a form to items in the data connection. Use the Binding Properties tool to update properties of those fields to match schema rules.
- See “**Walk-through One**” in “**Section I**” in your notes for **instructions**.
- Time: 10 minutes.

- Subforms typically used to represent sections
- **Positioned** subforms contain objects in specific locations



A diagram illustrating a positioned subform. It consists of a dashed blue rectangular border containing several input fields. The fields are arranged in a grid-like structure: 'Name' and 'Address' are full-width fields at the top. Below them, 'City', 'State', and 'Zip Code' are arranged horizontally, with 'City' and 'Zip Code' being wider than 'State'. 'Country' is a full-width field at the bottom. Each field has an orange border and a small orange dot at its top-left corner, indicating its position within the subform.

- **Flowed** subforms contain objects which flow one after another



A diagram illustrating a flowed subform. It consists of a dashed blue rectangular border containing several input fields. The fields are arranged vertically, with each field starting at the same left margin but having a different width. 'Name' and 'Address' are the widest, followed by 'Country' at the bottom. 'City', 'State', and 'Zip Code' are narrower and positioned to the right of the 'Name' and 'Address' fields. Each field has an orange border and a small orange dot at its top-left corner, indicating its position within the subform.

- Need a **repeatable** section to handle 1 to 5 registrants per form
 - When data is imported, we want all occurrences of registrant data to be visible (up to max of 5)
 - When filling the form, we want user to be able to add up to 5 registrants (e.g. for a family)
- Must use the **Instance Manager** to control instances (adding, removing, etc.)
- Instance Manager is property of **all** subforms:
 - AddressBlockSubform.instanceManager
 - Problem when zero instances exist (becomes inaccessible via script)
- By default, subform has min and max occurrence of 1 – making repeatable removes the max occurrence (becomes unlimited)
- Subform must be in **flowed** container before it can be made “repeatable”
- **Alternate/Suggested** way to access Instance Manager: “**underscore syntax**”
 - `_AddressBlockSubform`
 - **Always available** via script even if zero instances exist



- Simple form with repeatable subform and “add” button for new instances using “instanceManager” property
- Remove minimum (leaving zero initial instances)
- Try adding new instances
- See **error** in JavaScript console because repeatable subform doesn’t exist



- Return to form from Demo 1
- Use “underscore syntax” instead of “instanceManager” property
- Try to add new instances
- No more error!

- Adding instances:
 - `_RepeatingSubform.addInstance(0);`
- Removing instances:
 - `_RepeatingSubform.removeInstance(index);`
- Counting instances:
 - `_RepeatingSubform.count`

Exercise #2: Enable Add and Delete Buttons



- Enable the applicant to enter multiple registrants (from one to five) in the form. The applicant will be able to click the “Add” button to add registrants and each instance of the “Registrant” subform will have its own “X” button which can be used to remove that specific registrant from the set.
- See “**Walk-through Two**” in “**Section II**” in your notes for **instructions**.
- Time: 10 minutes.

- Repeatable sections should be placed in **flowed** containers (subforms)
 - Allows container to grow width/height to accommodate new instances
- Root subform (has name of “form1” by default) is **flowed**.
 - Subforms inside flow from page to page
 - Pages are achieved by inserting positioned subform same size as master page
- Using page subforms as containers for sections is ideal
 - Decreases subform layers
 - Simplifies hierarchy
 - Avoids nasty layout problems caused by having flowed content within positioned page subform, within flowed root container (e.g. content doesn't automatically flow onto new page)

Dynamic Form Design – cont'd



Name

Address

City State Zip Code

Country

List Box

Item 1	<input type="text"/>
Item 2	<input type="text"/>
Item 3	<input type="text"/>

City State Zip Code

Country

Before

Name

Address

City State Zip Code

Country

Page 1

Name

Address

City State Zip Code

Country

List Box

Item 1	<input type="text"/>
Item 2	<input type="text"/>
Item 3	<input type="text"/>

Page 2

After



- Simple form with repeatable subform placed within flowed container inside default page subform
- Fields under flowed subform
- Add instances of repeatable subform: Fields below repeatable subform aren't flowed downward because they aren't flowed with respect to repeatable subform




- Rebuild Demo 1 form by placing repeatable fields in resized page subform and “other” fields in separate positioned subform
- Add new instances
- Notice how “other” fields are reflowed downwards rather than remaining in front of the repeated fields

- **Tip:** Use “Insert > New Page” command to add new subforms to the root subform
 - Don't forget to set its **Place** property to “Following Previous” – defaults to “Top of Next” which causes it to break to the next content area resulting in a new page
- Fields can also resize to fit their data like flowed containers resize to fit instances of repeatable subforms
 - Set “Expand to fit” option on Layout palette with respect to “Width” and/or “Height” property
 - Ensures entire data is visible when form is printed
 - Avoids scroll bars
 - Must reside in flowed container to ensure following form objects are properly re-flowed to accommodate new width/height

Please list all medical conditions

Line 1
Line 2
Line 3



Please list all medical conditions

Line 1
Line 2
Line 3
Line 4



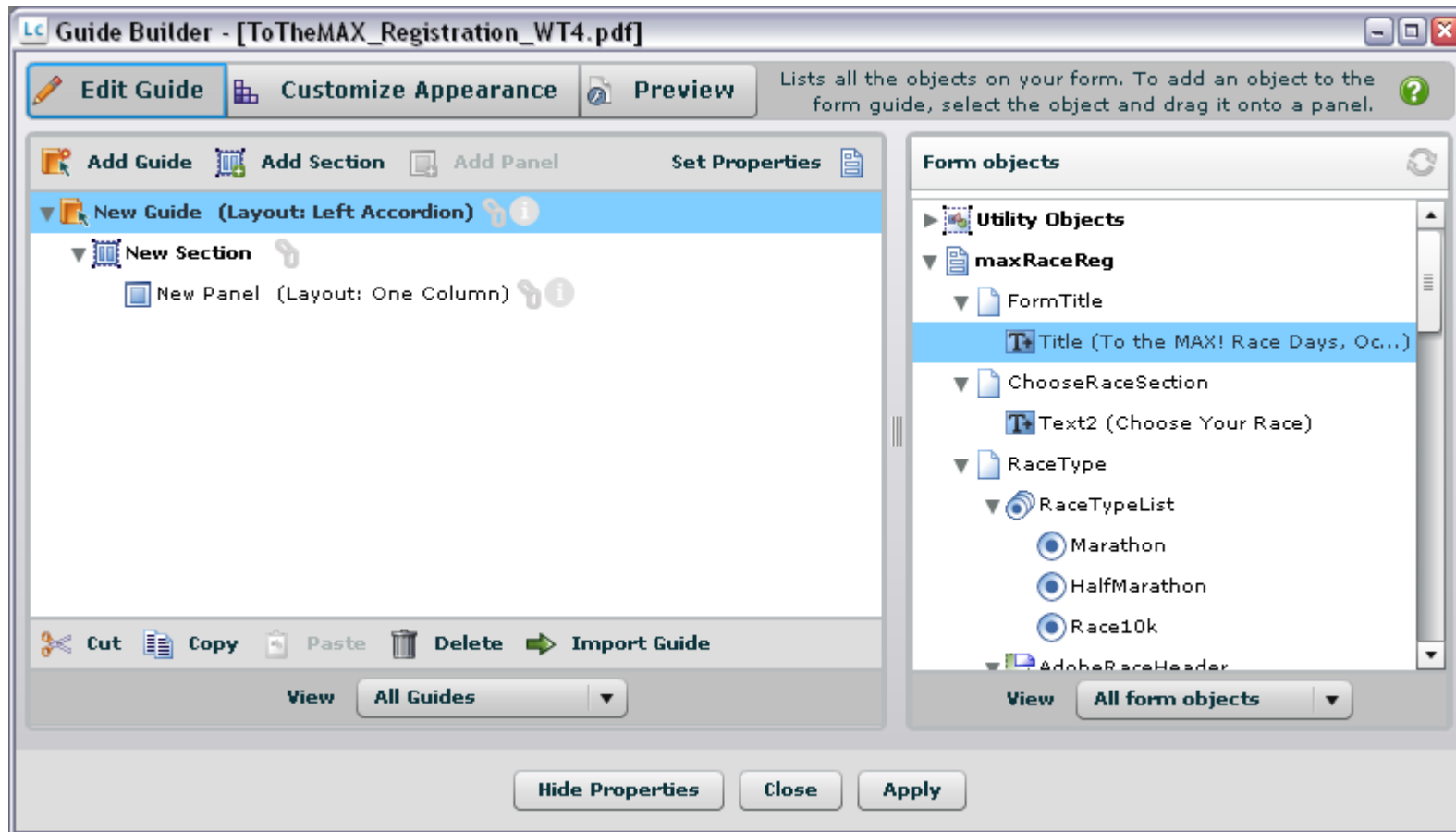
- Expandable field in positioned container: Field overlaps adjacent objects
- Place it in flowed container: Adjacent objects are re-flowed when field expands

Exercise #3: Enable Proper Content Flow

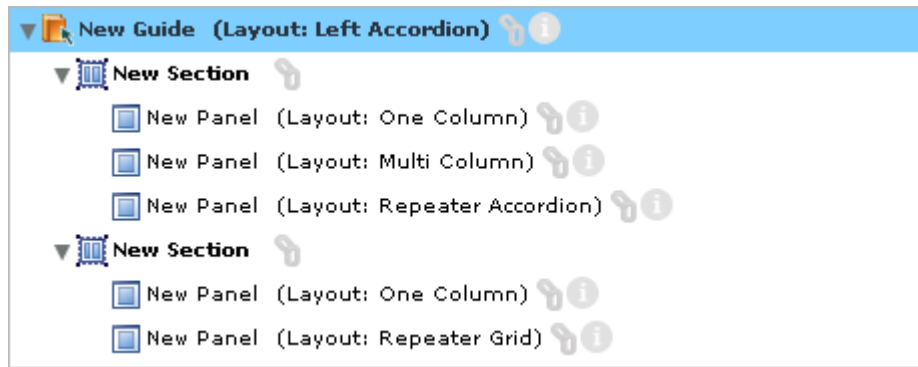


- Fix re-flow problems with Registrant Information section and its Medical Conditions field
- Import some data into the form to see the result of all this hard work!
- See **“Walk-through Three”** in **“Section III”** in your notes for **instructions**.
- Time: 10 minutes.

- New to LC Designer ES: **Guide Builder plug-in**



- **Alternate rendering** of form as **Rich Internet Application** using Flex running in Flash player
- Multiple guides within a single form is allowed
- Form Guide consists of sections which contain panels with different layouts



- Properties of guide objects linked to properties of form objects
 - Changes on form objects are reflected in guide objects



- Customize appearance of guides using Guide Builder and, for more advanced cases, Flex Builder
- Preview functionality in Guide Builder but **LiveCycle Forms ES** is required to deploy to browser
- Switch between PDF and Form Guide is possible at runtime (not preview)
 - Data entered in one place is automatically reflected in alternate rendering

Exercise #4: Design and Deploy a Form Guide



- Use the Guide Builder plug-in tool to quickly re-purpose the registration form as a Flash-based Form Guide and use Forms-IVS to deploy the form as a guide to a web browser.
- See “**Walk-through Four**” in “**Section IV**” in your notes for **instructions**.
- Time: 15 minutes.

- J.P. Terry's (SmartDocs Technologies) hands-on session on “**Extreme Form Makeover – with LiveCycle Designer 8, ES and Flex**”. This session will deal more in-depth with **form design techniques**.
- Anthony Rumsey's (Adobe) session on “Everything You Want to Know about **LiveCycle Form Guides**”. Deep-dive into Form Guides and customization.

- **scameron@adobe.com**
- Class notes and walk-through collateral available at **<http://forms.stefcameron.com/max2007/>**

Better by Adobe.™