

Acrobat™ Forms Field Naming Hints

What's in a Name?

A lot—especially if you're authoring a PDF-based form. Careful name selection for fields can make form authoring and data collection easy—poor name selection can make it a downright chore. After all, it's tough enough filling out forms let alone authoring one. Enough said, let's talk about field names and how best to use them. We'll cover naming and how it changes the way that form fields are organized into a hierarchy within the document, how to take advantage of this hierarchy in JavaScript, and the Personal Field Name Specification and how it can make a form filler's life easier.

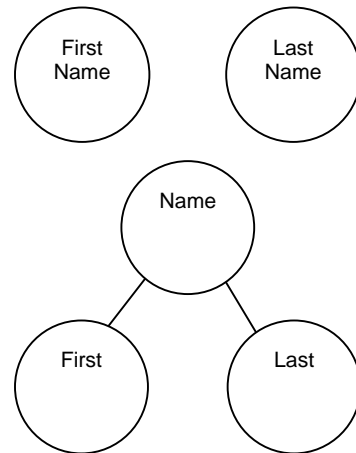
Data Sharing

A useful property about Acrobat Form fields is that fields that share the same name also share the same value. They can have different *presentations* of that data; they can appear on different pages, be rotated differently, have a different font or background color, etc. but they have the same value. This means that if you modify one field all others with the same name get updated automatically. In some forms packages, you have to perform a lot of work to get this to happen correctly. In Acrobat Forms, this happens pretty much for free.

Form Field Hierarchies

Typically, form fields have names like **FirstName**, **LastName**, etc. They are what we call *flat names*, there is no association between these fields. For many form applications, this flat hierarchy of names is sufficient and works well.

By tweaking the field names slightly, we can create a hierarchy of fields within the document. For example, if we change them to **Name.First** and **Name.Last** we form a tree of fields. The period ('.') separator in Acrobat Forms is used to denote a hierarchy shift. The **Name** portion of these fields is the parent, and **First** and **Last** are the children. There is no limit to the depth of a hierarchy that can be constructed but it is important that the hierarchy remain manageable. It is also important to clarify some terminology: the field **Name** is known as an *internal* field (that is, it has no visible manifestation) and the fields **First** and **Last** are *terminal* fields (and show up on the page).



Note that this hierarchy can be useful in a number of ways including speeding up authoring and easier manipulation of groups of fields via JavaScript. In addition, a form field hierarchy can improve the performance of the forms engine if there are many fields in the form.

Name Clashes

One of the questions frequently asked is why you get the cryptic “You have chosen a name for your field that conflicts with an existing field. Either choose a new name or change the type of your field to match the existing field” alert dialog. As shown in the above diagram, you can't have two terminal fields where one is named **Name.First** and the other is **Name**. The diagram illustrates that **Name** already exists as an internal field and so you can't have it as a terminal field as well (i.e. one drawn on the screen).

You will also get this message if you have two fields with the same name and try to change one of them to a different type of field (i.e. text, radio, check, combo, button, list box). Fields that have the same name must also be of the same type.

Authoring

Acrobat Forms has the ability to automatically increment or decrement field names if they conform to a certain style. This feature helps to quickly copy and rename fields in a table. It is typical for cells in a table to be named with a numeric suffix (e.g. .0, .1, etc.). Acrobat Forms recognizes this type of numeric suffix and allows you to rename fields using the plus ('+') and minus ('-') keys when the field is selected without having to go to the properties dialog.

| | | | |
|--------|---------|------------|---------|
| Item 0 | Price 0 | Quantity 0 | Total 0 |
| Item 1 | Price 1 | Quantity 1 | Total 1 |
| Item 2 | Price 2 | Quantity 2 | Total 2 |

JavaScript

One of the other uses of field hierarchies is the ability to manipulate a group of fields using a single JavaScript statement. Given our original flat names for **FirstName** and **LastName**, imagine that we want to change the background color of these fields to yellow (to indicate missing data, perhaps). We would need one line of JavaScript code per field.

```
this.getField("FirstName").fillColor = color.red;  
this.getField("LastName").fillColor = color.red;
```

Given our hierarchy of **Name.First** and **Name.Last** above (and perhaps, **Name.Middle**), we can change the background color of these fields with one line of code:

```
this.getField("Name").bgColor = color.red;
```

This same technique can be used to make fields read-only or hidden or to change any of the other properties available for manipulation through JavaScript.

This same naming scheme is also handy for summation and other tasks. The built-in scripts for Acrobat Forms can take advantage of your naming schemes to make summing an entire column of figures easy. In our invoice example above, you can calculate the value of a field **GrandTotal** by summing the totals using the old method and a custom calculation script:

```
var total = 0.0;  
for (i = 0; i < 2; i++)  
    total += this.getField("Total." + i).value;  
event.value = total;
```

Alternatively, you can use the simple calculation feature and author the **GrandTotal** using just the user interface; no custom script is necessary.



Personal Field Names

How many times have you filled out your social security number on a form? Too many, I'm sure. So Adobe has come up with a way of naming fields, called the *Personal Field Names Specification*, that can make it such that you never have to type your social security number and other common data ever again.

There is a document (PFNForm.pdf) that comes on your Acrobat Forms Maker CD in the Samples/Adobe/PFN folder that allows a user to enter in all the data they've ever had to enter (just one last time) and save this data to their local hard drive. Any form that they come across that conforms to the Personal Field Names (PFN) Specification allows them to instantly populate that form with their saved data. Sounds simple, right?

Well, as a forms author, you have to be careful about how you name fields (yes, the whole naming thing again). The PFN document lists several common fields and their suggested names. Corporations might want to add their own names to this list (e.g. **Adobe.CostCenter**) using a prefix that is corporate-specific. As an author, you also have to add a special button to the document that has the PFN logo and an action to import a user's profile from disk. A little extra work for a lot of convenience for your form user. A standard name for fields also makes it easier on the back-end process to collect data from a variety of forms and process it in similar ways. Your webmaster will be much happier treating form data in a consistent manner.

Summary

So what is in a field name? Productivity, simplicity, and enhanced capabilities for starters. We've seen how naming fields appropriately can ease authoring, calculations and interactivity, and fill-in.

It is encouraged that you become an Acrobat Forms Expert! Listed below is a suggested reading list:

Portable Document Format Reference Manual v1.3. Sections on Form Fields and Actions and the Form Data Format specification. This document can be found on Adobe Systems Incorporated web site at <http://partners.adobe.com/asn/developer/acrosdk/DOCS/pdfspect.pdf>

Acrobat Forms JavaScript Objects Specification (AcroJS.pdf) and *the JavaScript Language Specification* (JSSpec.pdf). These can be found in your Acrobat Exchange help folder if you've installed the Acrobat Form Maker version 3.5 update.

Acrobat Forms Data Format Toolkit. This document can be found on Adobe Systems Incorporated web site at <http://partners.adobe.com/asn/developer/acrosdk/forms.html>.

Tips and Techniques for Acrobat Forms. These short tutorials can be found on Adobe Systems Incorporated web site at <http://www.adobe.com/studio/tipstechniques/acrobat.html>.