

Moving from Acrobat to LiveCycle Designer JavaScript

Thom Parker

WindJack Solutions, Inc.

<http://www.windjack.com>

Presented At



Copyright ©2005, WindJack Solutions, Inc

Goals

To Understand:

- Differences between AF¹ and XFA² JS³
- What can and can't be done with XFA JS
- The Conversion Process: AF to XFA

1. AF – AcroForm

2. XFA – Extensible Forms Architecture

3. JS – JavaScript



XFA JavaScript Info

- XML Forms Architecture Documents

http://partners.adobe.com/public/developer/xml/index_arch.html

- XFA SOM Reference
- XML Form Object Model Reference
- More than you want to know

- Converting from Acrobat JavaScript to XFA JS

http://partners.adobe.com/public/developer/en/acrobat/sdk/AcroJS_DesignerJS.pdf

- General Acrobat JavaScript

http://partners.adobe.com/public/developer/pdf/topic_js.html



Differences between AcroForms and XFA

AcroForm PDF

- AcroForm refers to the Form-centric parts of a PDF document
 - Includes Form Fields and JavaScript that acts on Form Fields
- AcroForm Form Fields Live in the PDF Structure in a top level dictionary named “**AcroForm**”
- Form Fields are interactive elements like:
 - Buttons, Text Boxes, Lists, Check Boxes, etc.
- Form Fields are usually added to a PDF with the Acrobat Form Tools, but may also be added with 3rd party tools.



Folder Level JavaScripts

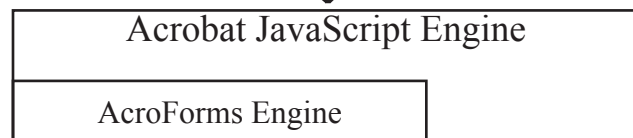
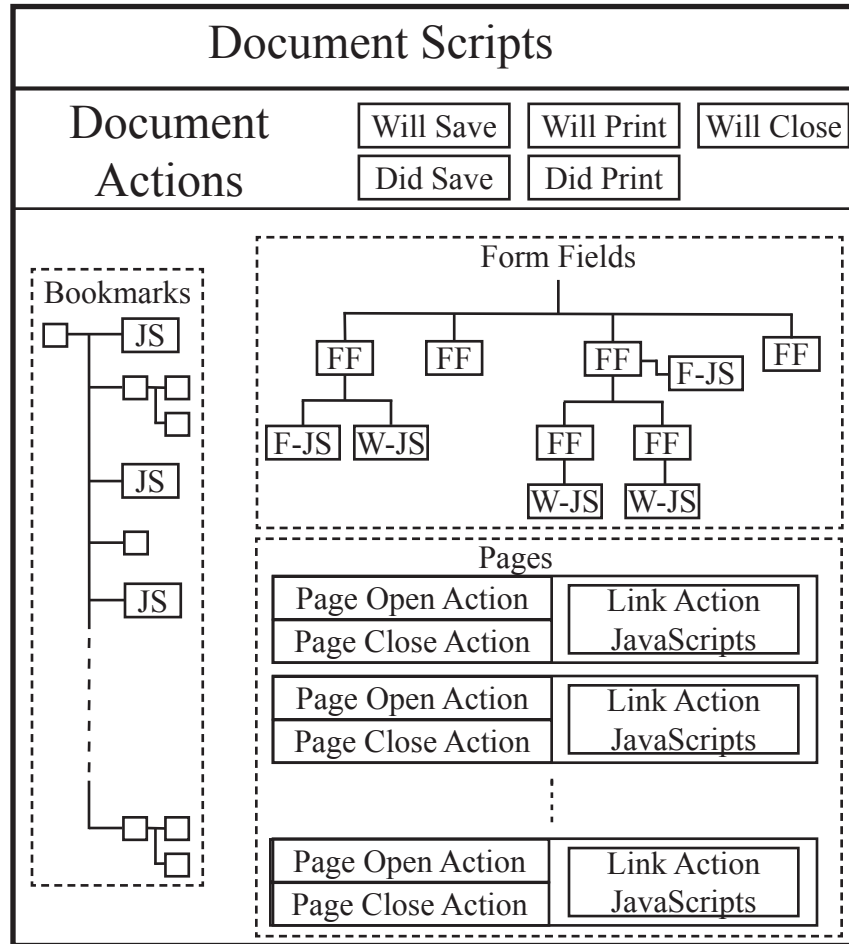
Timer Events

Menu and Toolbar JavaScripts

Batch Sequence JavaScript

Console Window

AcroForm PDF



XFA PDF Documents

- LiveCycle Designer 7 Creates XFA Docs
- XFA is an XML Spec for Forms
- XFA can be embedded into a PDF File
- XFA is not PDF, It's completely different
- The PDF part of a Designer 7 Doc is only window-dressing
- Acrobat 7 has a separate engine for operating on XFA docs and executing XFA Scripts
- Acrobat 7 turns off JavaScript for AcroForm processing when it loads an XFA Doc
- XFA Scripting Model is not related to traditional AcroForm JavaScript in any way

Folder Level JavaScripts

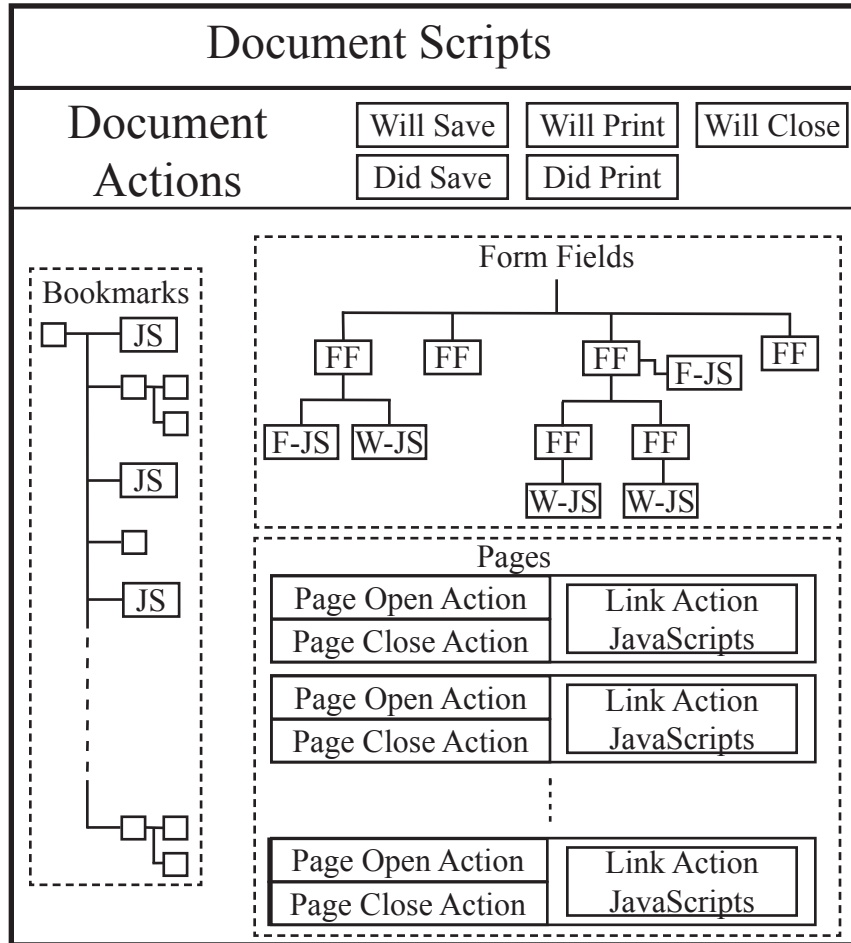
Timer Events

Menu and Toolbar JavaScripts

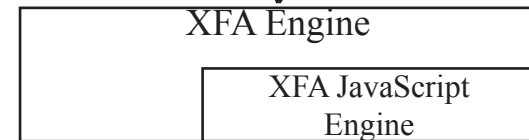
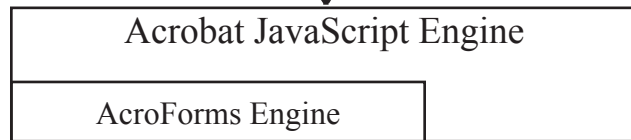
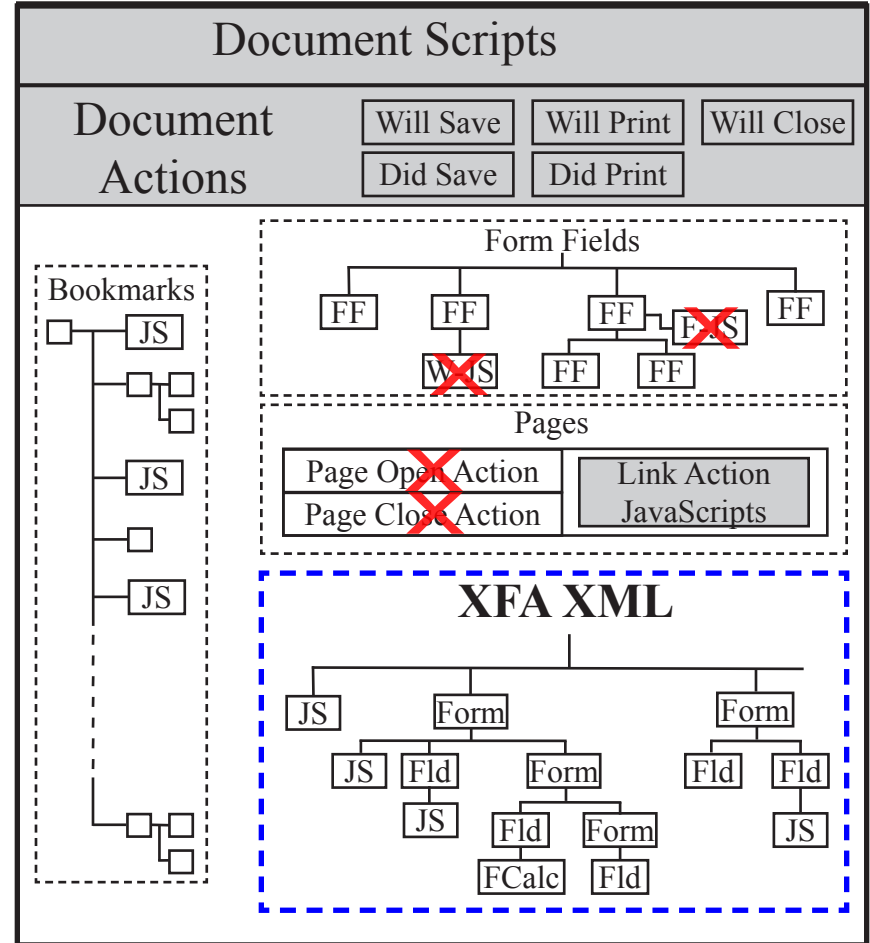
Batch Sequence JavaScript

Console Window

AcroForm PDF



XFA PDF



AcroForm to XFA JavaScript

Copyright ©2005, WindJack Solutions, Inc



What Can and Can't be Done with XFA JavaScript

What Can and Can't be Done with XFA JavaScript

What You Can do with XFA JavaScript

- The Normal Forms Operations
 - Calculations
 - Custom Formatting and Validation
 - Reset Fields
 - Show/Hide Fields
 - Import/Export Data
 - Print Form
 - Set/Get Field Geometry
 - Set/Get Field Font Parameters
 - Set/Get Field Values
 - Manipulate List Fields (Add/Delete Items)
- Navigate the entire XFA XML
- Access Acrobat JavaScript



What you don't need to do with XFA JavaScript

All of these operations are built
into the XFA Engine

- Ordinary Form Data Submission
- Common Formatting
- Soap Connections
- Parse Custom XML

What you can't do with XFA JavaScript

- *Anything not specifically related to Forms
- Add new Form Fields
- Add Field JavaScript Actions
- Set Default and Export Values
- Change Calculation Order
- Change Tabbing Order
- *Custom Data Submission
- *Digital Signature Processing
 - * Operations can be done with Acrobat JS

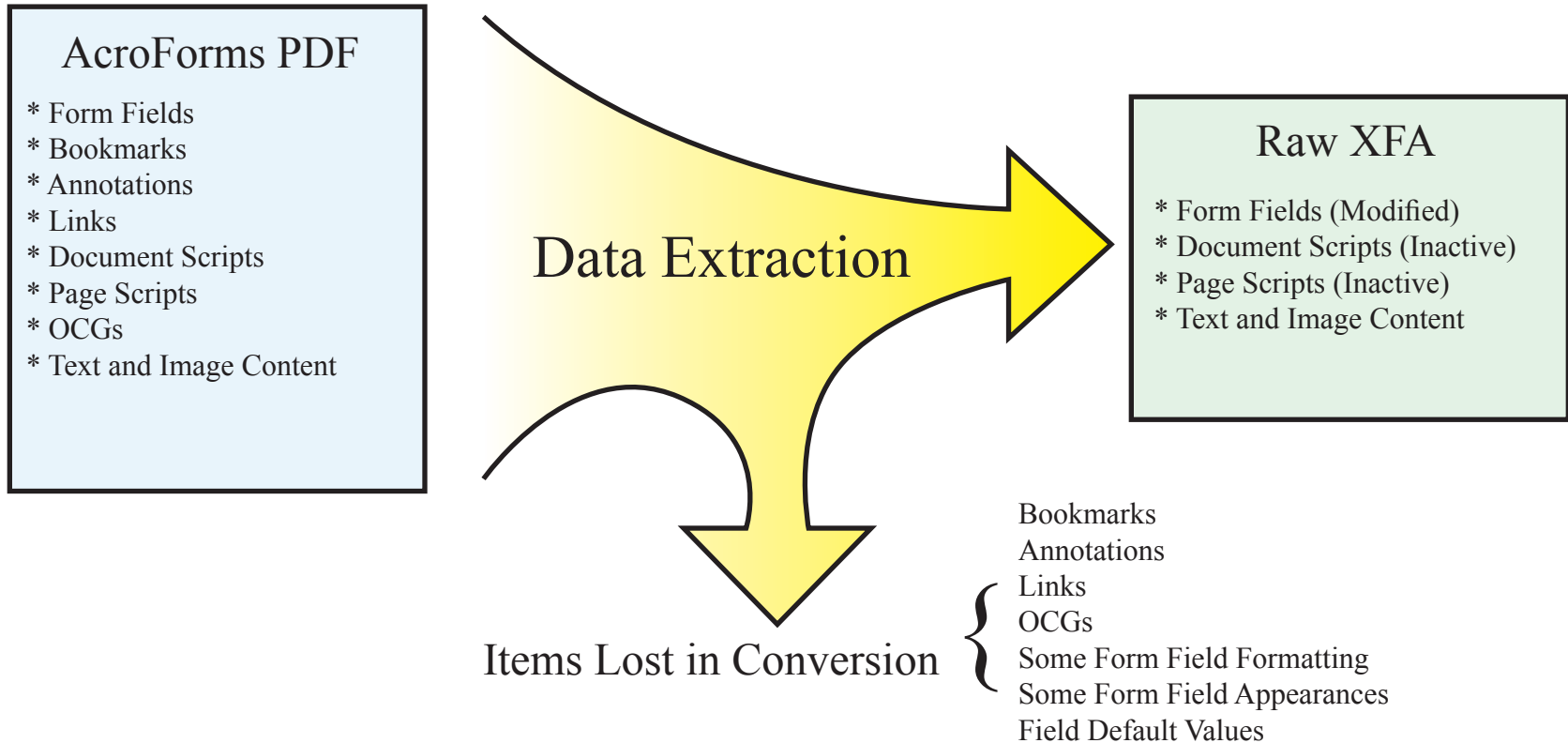


The Conversion Process: AcroForm PDF to Live Cycle Designer PDF

Conversion Process: AcroForm PDF to LiveCycle Designer 7 XDP

LiveCycle Designer analyzes the PDF for those items that can be represented in XFA, and extracts only that information into the XDP (XML Data Package)

- * JavaScript is copied into XDP verbatim, but commented out.
- * Some Formatting, Calculations, and other interactive items are converted, some aren't.



Items Converted

- Text and Graphical Content
- All Form Fields
 - Field Options (Rotation, Read Only, Required...)
 - Most Field Appearance Parameters (Font, color...)
- Some Field PDF Actions
 - Reset Form (Built-in Action)
 - Submit Form (Built-in Action)
 - Number and Currency Formatting
 - Simplified Notation for Calculations
 - Built-in Calculations



Items Lost in the Conversion

- Document Properties
- Initial View
- Annotations (Stamps, Notes, Text Highlights)
- Links
- Bookmarks
- Named Destinations
- Duplicate Fields (i.e., ones with the same name)
- Some Field Appearances
- Some Formatting Options (All special, percent,...)
- Most Field Actions (Other than JS, Reset & Submit)
- Optional Content Groups
- File Attachments
- Tabbing Order



Basic JavaScript Concerns

- Field Names
- Setting and Getting Field Values
- Resetting Fields
- Data Submission
- Event Handling
 - Validate, Format, KeyStroke, Calculation
- Document Object Access
- Accessing and using Acrobat JavaScript



Accessing Form Data

- Form data and properties are accessed through the SOM
- The SOM provides a syntax for specifying the logical location of a particular object and object property within the Form
- The SOM is used differently by JavaScript and FormCalc, but is identical in structure for both



XFA Form Structure

- Composed of a hierarchy of Sub-Forms (Enclosures) and Form Fields
- Think of Boxes in Boxes
 - Each Box contains either another Box or an Item

SOM Root

- The Root of the SOM is the “xfa” node
- Different elements of the SOM are accessed using the “dot” object notation
- The SOM is divided at the top level into Logical areas of functionality
 - **xfa.host** – system and global doc functionality
 - **xfa.datasets** - field data values
 - **xfa.localdata** – language localization data
 - **xfa.template** – form structural data
 - **xfa.form** – form field data/event scripts
 - **xfa.layout** – physical structure of form (pages, etc.)



SOM Paths

- All fully qualified paths start with the root
ex: `xfa.form.form1.address.street.rawValue`
- Relative paths can be used within a subform to access all sibling elements

From the example above a node in the “address” subform could access the “street” field like this

ex. `street.rawValue`



XFA XML Nodes

- The SOM is a parsed DOM style XML
- Each part of a SOM path is an XML node
- General Node Operators
 - **nodes**: List of child nodes
 - **nodes.length**: Number of child nodes
 - **nodes.item(*index*)**: Access to child node
 - **nodes.remove(*node*)**: Removes specified node
 - **nodes.insert(*node*)**: Inserts node into list
 - **className**: XML tag name
 - **saveXML(*style*)**: Returns XML text for node
 - **loadXML(*strXML*)**: Parses XML text and adds to node
 - **name**: Name of this node
 - **parent**: Parent of this node



XFA Objects

- Each XFA SOM Node is a Specific XFA Object.
 - SubForm, Field, Value, fillColor
- Each XFA Object Type has a Set of Properties and Methods that is specific to that Object Type.
 - These Properties and Methods reflect the Attributes and sub-Elements of the XML node.
 - Ex: value is a property of a Field Object.
 - They are accessed using a standard XFA Path
 - Ex: xfa.form.F.P1.ResetButton.fillColor



Where Do the Scripts Go?

- All LiveCycle Designer Scripts are accessed via the “Scripts Editor.” Open from the “Window” Menu
- Document Level Scripts
 - Converted to an XFA Variable
 - In Designer on the Hierarchy WindowEach Document Level Script will be represented by a “Variables” entry of the same name



Where Do the Scripts Go?

- Document Actions
 - Converted to a Form Script
 - In Designer on the Hierarchy Window
Select the node representing the top level Form

Acrobat Event	Designer Event
WillClose	docClose
Will Save	preSave
DidSave	postSave
Will Print	prePrint
Did Print	postPrint



Where Do the Scripts Go?

- Page Actions
 - Converted to a Sub-Form Script. Pages in an XFA Document are represented by a Sub-Form
 - On the Hierarchy Window select a node that represents a page. This will normally be a node just below the top level

Acrobat Event	Designer Event
PageOpen	enter
PageClose	exit

Where Do the Scripts Go?

- Form Event Scripts
 - Converted to a Form Script
 - Within a group of fields having the same name, only one field will acquire the Form Scripts. All other fields in the group will have empty Form Scripts
 - In Designer, Select the Form Field

Acrobat Event	Designer Event
Keystroke (Change)	change & exit
Format	exit
Validate	validate
Calculate	calculate



Where Do the Scripts Go?

- Form Widget Actions (Mouse Actions)
 - Converted to a Form Script
 - In Designer, Select the Form Field

Acrobat Event	Designer Event
Mouse Up	mouseUp
Mouse Down	mouseDown
Enter	mouseenter
Exit	mouseleave
Focus	enter
Blur	exit



3 Easy JavaScript Conversions

1. console, app, search, and other non “doc” related objects remain the same
2. Replace “this” with “event.target”. In XFA JS “this” is the field object that caused the event, not the “doc”, as it is with AcroForms
3. Replace “this.resetForm([...])” with:
`xfa.host.resetData(“...”);`
 - The list of fields to reset is quoted rather than in JS array format, i.e., using the “[.]” (square bracket notation)



Accessing Acrobat JavaScript

- XFA scripting is focused on forms operations. For some operations it is necessary to use the Acrobat JavaScript model
- “app” object is globally available
 - xfa.host encapsulates many of the app functions and properties
- Static DOM objects are unchanged for XFA PDFs

console	ADBC	SOAP
global	security	Collab
spell	Report	dbg
search	XMLData	tts

- event.target returns the current “doc” object
 - Most Doc Methods that change the doc are unavailable
- Field methods and properties are generally unavailable, only a few are available for static XFA docs



XFA Data Submission

- Data Submission is built into the XFA Engine
 - AcroForm Submit Actions are Converted Directly into XFA Submit Actions
- AcroForm JavaScript Data Submission works from XFA JavaScript
 - change “this.submitForm(...)” to “event.target.submitForm(...)”

Events in XFA

- Like AcroForm JavaScript, All XFA JavaScript is Executed as the Result of an Event
- The “Event” object Provides Access to Information About the Context of the Event
- There are 2 event objects Available from XFA JavaScript
 - The old AcroForm Event Object, Slightly disabled
 - The XFA JavaScript Event with Properties for the XFA Context.



Event Objects

- **AcroForm JavaScript Event**
 - Access with unqualified “event” keyword
 - Ex: `var strEventName = event.name;`
 - Provides Access to the Document Object
 - Ex: `var strDocTitle = event.target.path;`
- **XFA JavaScript Event**
 - Access through “xfa.form”
 - Ex: `xfa.form.event.name;`
 - Provides access to event type name
 - For the “Change” Event, Provides Access to the Data Dtate of the Field Element that Caused the Event.



Thank You for Attending this Session

For more information on JavaScript
please visit our Web Site:
<http://www.windjack.com>