

SCRIPTING SUPPORT FOR HTML FORMS AND GUIDES



© 2010 Adobe Systems Incorporated and its licensors. All rights reserved.

Scripting Support for HTML Forms and Guides

September 24, 2010

This user guide is licensed for use under the terms of the Creative Commons Attribution Non-Commercial 3.0 License. This License allows users to copy, distribute, and transmit the user guide for noncommercial purposes only so long as (1) proper attribution to Adobe is given as the owner of the user guide; and (2) any reuse or distribution of the user guide contains a notice that use of the user guide is governed by these terms. The best way to provide notice is to include the following link. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Adobe, the Adobe logo, Adobe Reader, Acrobat, Flash, and LiveCycle are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Apple is a trademark of Apple Inc., registered in the United States and other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

Chapter 1: About This Document

Who should read this document?	1
How to use this document	1
Additional information	1

Chapter 2: Overview

Objects that support scripting	6
--------------------------------------	---

Chapter 3: Creating Scripts for HTML Forms and Guides

Choosing a scripting language	8
Executing JavaScript in a specific client application	8
Considerations when creating JavaScript for Guides	9
Limitations when working with data sources	10

Chapter 4: Object-Level Scripting Support

xfa	12
-----------	----

Chapter 5: Scripting Events

Chapter 6: Scripting Properties

Chapter 7: Scripting Methods

Chapter 1: About This Document

This document outlines the XML Form Object Model scripting events, objects, properties, and methods that are available for forms rendered in HTML and as Guides. For more information about the availability of other features, see [Target Version Reference](#) and [Transformation Reference](#).

Who should read this document?

This document is intended for form developers interested in either creating scripts for forms that will be rendered into HTML or as Guides, or in maintaining existing scripting capabilities across several form output types. Knowledge of JavaScript™, the XML Form Object Model, as well as data binding using Adobe® LiveCycle® Designer ES2 is expected.

How to use this document

This document is divided into sections to present similar kinds of information in different ways. The intent is to allow you to locate what you need as quickly as possible.

Topic	Provides information about
“Overview” on page 3	The scope of the support for scripting in HTML forms and Guides
“Creating Scripts for HTML Forms and Guides” on page 8	Considerations for creating scripts that may need to run on multiple client applications, as well as techniques for adapting your existing scripts
“Object-Level Scripting Support” on page 12	What scripting objects are supported for HTML forms and Guides. For each object, any supported properties and methods are also listed. Use this chapter if you want to know, generally, what properties and methods are supported for a particular scripting object.
“Scripting Events” on page 19	A complete, alphabetical list of all scripting events and their availability for HTML forms and Guides.
“Scripting Properties” on page 20	A complete, alphabetical list of all scripting properties and their availability for HTML forms and Guides. Use this chapter if you want to know the availability for a specific scripting property.
“Scripting Methods” on page 29	A complete, alphabetical list of all scripting methods and their availability for HTML forms and Guides. Use this chapter if you want to know the availability for a specific scripting method.

Additional information

The resources in this table can help you learn more about Adobe® LiveCycle® Enterprise Suite 2.5 (ES2.5).

For information about	See
Detailed information about creating and editing Guides using the Guide Design perspective in Adobe® LiveCycle® Workbench 9.5	Creating Guides
Getting started creating scripts in Designer	LiveCycle Designer ES2 Scripting Basics
Complete reference information detailing the scripting objects, properties, methods, and events available in Designer	LiveCycle Designer ES2 Scripting Reference
Adobe LiveCycle ES2.5 terminology	LiveCycle ES2.5 Glossary
Other services and products that integrate with Adobe LiveCycle ES2.5	www.adobe.com
Patch updates, technical notes, and additional information on this product version	LiveCycle Technical Support

Chapter 2: Overview

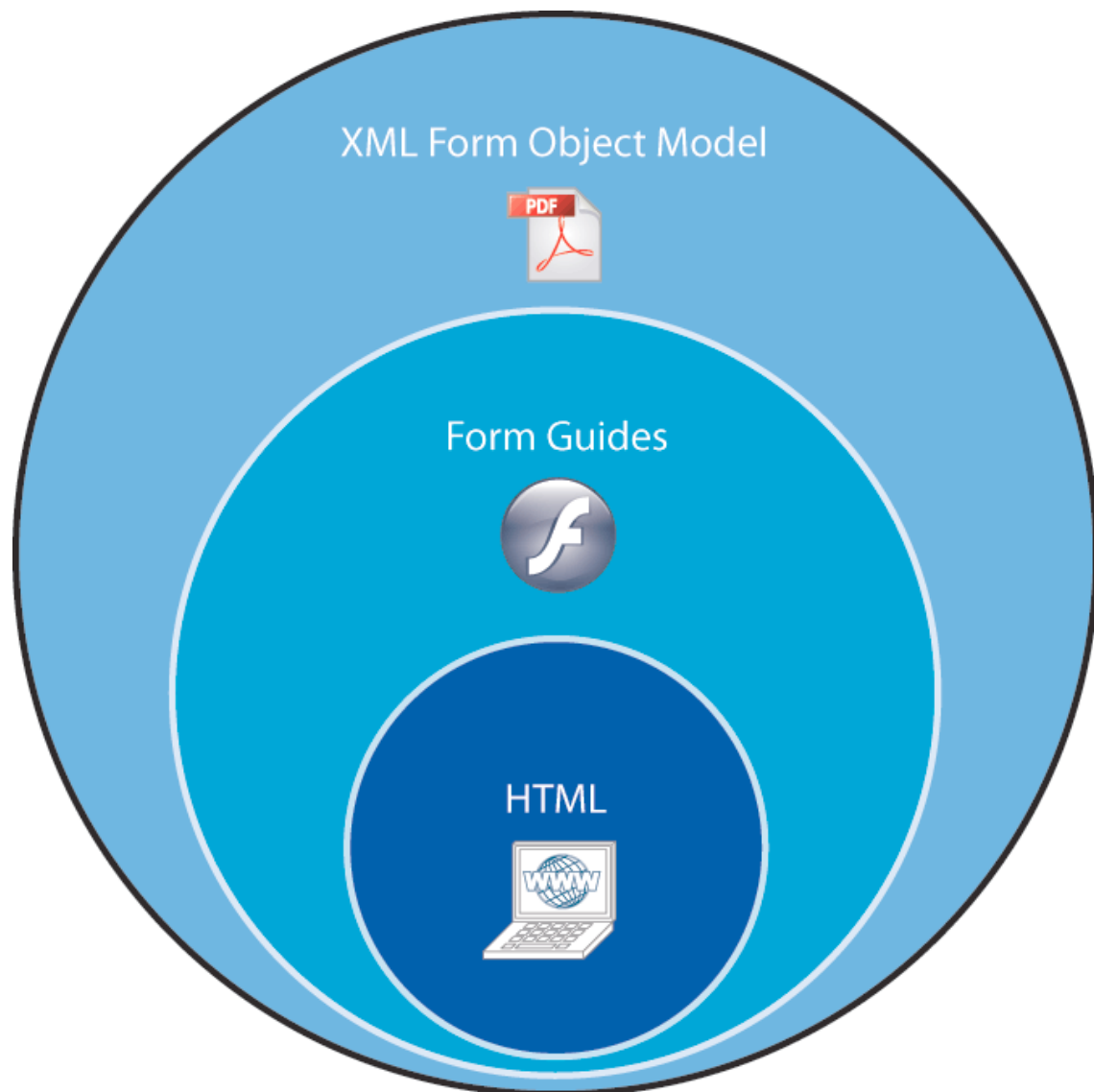
When you create forms using Designer, you have the option of rendering those forms in either PDF or HTML format. Using the Guide Design perspective in Workbench, you can also create Guides based on Adobe Flash® technology. End users view each of the formats (PDF, HTML, and Guides) by using an associated client application. The following table provides an overview of the client applications required to view PDF forms, HTML forms, and Guides.

Format	Client application
PDF	Adobe Acrobat® Professional, Acrobat Standard, Adobe Reader®
HTML	Microsoft® Internet Explorer, Mozilla Firefox, Netscape Navigator, Apple® Safari
Guide	Adobe Flash Player

Note: While end users may view the PDF form, HTML form, or Guide in an application such as Adobe® LiveCycle® Workspace 9, it is important to remember that it is the underlying client technology that is responsible for displaying the form or Guide and its data.

Overview

Not all form features available in Designer are available across all output formats. Forms rendered in PDF format, by default, have access to the full XML Form Object Model, including all scripting events, objects, properties, and methods. Forms rendered in HTML format or as Guides have access to only a portion, or *subset*, of the XML Form Object Model. The following diagram illustrates the relationship between the full XML Form Object Model and the subsets available for HTML forms and Guides. Note that the support for Guides is a subset of the full XML Form Object Model, and that the support for HTML forms is a subset of what is supported for Guides.



Overview

The reduced scope of the XML Form Object Model for HTML forms and Guides means that if you are creating form designs that may be rendered in either of those formats, you will need to consider how to structure your form scripting. To prevent unexpected results, avoid referencing objects, properties, or methods that are not available.

Note: *Guides support version 2.8 of the XML Form Object Model (XFA).*

Objects that support scripting

Designer uses two separate definitions of objects with respect to scripting:

- Objects that you add to your form design from the Object Library palette
- Objects exposed from the underlying XML architecture of a form design

Architecturally, objects that exist within the Object Library palette are more meaningful representations of objects exposed in the underlying XML. In fact, all objects available in the Object Library palette are derived from only four XML objects: contentArea, draw, field, and subform.

The Designer scripting model allows you to reference objects directly using the name of the object. For example, if you have a text field on your form, and you want to set its value, you could use the following JavaScript:

```
TextField1.rawValue = "Hello World";
```

From an architectural perspective, this is logically equivalent to the following:

```
field.rawValue = "Hello World";
```

The following table outlines how the Object Library objects map to their underlying XML equivalents.

Object on the Standard tab of the Object Library	Derived from
Content Area	contentArea
Circle, Image, Line, Rectangle, Text	draw
Barcodes (all), Button, Check Box, Date/Time Field, Decimal Field, Drop-Down List, Email Submit Button, HTTP Submit Button, Image Field, List Box, Numeric Field, Paper Forms Barcode, Password Field, Print Button, Radio Button, Reset Button, Signature Field, Text Field	"field" on page 15
Subform, Table (each body row, header row, and footer row is a distinct subform object)	"subform" on page 16 "subform" on page 16. "instanceManager" on page 17

Note: *You cannot script against Designer objects derived from contentArea or draw objects. To find what properties and methods are available for your Object Library objects, consult either the "field" on page 15 or "subform" on page 16 sections of "Object-Level Scripting Support" on page 12.*

In addition to objects available from the Object Library in Designer, objects that are only available through scripting are supported by HTML forms and Guides. The table below outlines how you reference the supported scripting model objects using script:

Supported scripting model objects	Syntax
"data" on page 12	<code>xfa.data.</code>
"data" on page 12."datasets" on page 13	<code>xfa.data.datasets.</code>
"eventpseudomodel" on page 13	<code>xfa.event.</code>
"hostpseudomodel" on page 14	<code>xfa.host.</code>
"form" on page 14	<code>xfa.form.</code>
"xfa" on page 12	<code>xfa.</code>

For more information on the entire Adobe XML Form Object Model, see [LiveCycle Designer ES2 Scripting Reference](#).

Chapter 3: Creating Scripts for HTML Forms and Guides

When you add scripts to a form you create in Designer, you must decide where those scripts will execute: either on the client application, on the server, or on both the client and the server. If you choose to execute your scripts on the server in any way, your form must be deployed in conjunction with Adobe® LiveCycle® Forms 9 so that the form and its data can return to the server when the script executes. After the script executes, the form is re-rendered and posted back to the end user. Executing scripts on the server provides you with access to the full XML Form Object Model objects, scripting properties, and scripting methods.

If you choose to run your scripts on the client application, there are restrictions to the scope of the XML Form Object Model you can access, and even what scripting language you can use. This section discusses the considerations you need to keep in mind when creating scripts for HTML forms or Guides.

Choosing a scripting language

If you are creating scripts that execute on the server, you are free to create scripts in either JavaScript or FormCalc. However, if you are creating scripts that will execute on the client application, you can only use JavaScript. This restriction applies because FormCalc is not natively available in client applications, such as internet browsers or Flash Player.

If you have an existing PDF form that you want to make available in HTML format or as a Guide, one strategy is to set your existing FormCalc scripts to execute on the server. This technique may not always be appropriate for your specific use case, for example, if you don't want the form to make a round trip to the server as part of the form filling experience. However, this technique may be useful if you do not want to rewrite your existing FormCalc script in JavaScript.

***Note:** Not all scripting events can execute on the server. For example, interactive events are specific to user interactions with the form in the client application, so they cannot be set to execute on the server.*

Executing JavaScript in a specific client application

If you want to customize your scripting depending on the particular client application in which the form or Guide is being viewed, you can use the `name` property of the `host` model to conditionalize your scripts. For example, the following JavaScript executes only if the form is rendered as a Guide:

```
if (xfa.host.name == "Flash")
{
    xfa.host.messageBox("This is executing in Flash Player!");
}
```

Similarly, the following JavaScript executes only if the form is rendered in HTML format (and the form is viewed in Internet Explorer):

```
if (xfa.host.name == "IE")  
{  
    xfa.host.messageBox("This is executing in Internet Explorer!");  
}
```

Form output format	Client application	xfa.host.name
PDF	Acrobat Standard, Acrobat Professional, Acrobat Pro Extended, and Adobe Reader	Acrobat
Guide (SWF)	Adobe Flash Player	Flash
HTML	Internet Explorer	Microsoft Internet Explorer
HTML	Mozilla Firefox	Netscape
HTML	Apple Safari	Netscape

If you want to distinguish between Firefox and Safari, you can use the standard HTML object model to access the `window.navigator.userAgent` property. The `userAgent` property returns a string that should contain the name of the application. For example, for Firefox 2.0.0.13 the `userAgent` property contains the following value:

```
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.13) Gecko/20080311 Firefox/2.0.0.13
```

The following JavaScript uses the `String.search()` method to determine if the value of `userAgent` contains either Firefox or Safari, and displays a customized message box for each case.

```
var sVar = window.navigator.userAgent;  
  
if (xfa.host.name == "Netscape")  
{  
    if (sVar.search(/FireFox/) != -1)  
    {  
        xfa.host.messageBox("This is executing in Mozilla Firefox!");  
    }  
    if (sVar.search(/Safari/) != -1)  
    {  
        xfa.host.messageBox("This is executing in Apple Safari!");  
    }  
}
```

Note: Individual applications can configure the `userAgent` property to display any string value. You should verify the value of `userAgent` for specific versions of the HTML client application on which your users will view the rendered form.

Considerations when creating JavaScript for Guides

When you create a Guide based on a form design, you should consider the following:

- Scripting on master pages is ignored.

Guides do not use page-based layout. Instead, Guides use the concept of sections and panels which the Guide author creates in Workbench. Therefore, no scripting on master pages defined in Designer is preserved when the Guide is rendered.

- Field level dependencies are ignored, with the exception of the `rawValue` property.

Field data values, controlled using the `rawValue` property, that are dependent on the values of another field or fields will continue to operate as expected. All other field properties that are dependent on the values of other fields will not persist on a Guide. For example, if the color of a field's border is set to change when the border color of another field changes, this behavior will not persist on a Guide.

- Guides cannot correctly use the `this` JavaScript shortcut reference as part of a script object.

For example, the following function will not evaluate correctly on a Guide.

```
function addToValue( nNumber )
{
    this.rawValue = this.rawValue + nNumber;
}
```

In this case, you must modify the function to either explicitly pass the `this` shortcut reference as a variable, or pass a variable that represents a pointer to the current object as a formal parameter. For example:

```
addToValue( this, 10 );
- or -
function addToValue( obj, nNumber)
{
    obj.rawValue = obj.rawValue + nNumber;
}
```

- Guides return the value of an empty field as "" (empty string), while Acrobat and Adobe Reader return `NULL`. In situations where a script may execute in both Acrobat/Adobe Reader and Flash Player, you should make sure you test for both values using conditional statements.

For example, the following script evaluates to `True` (0) for both Guides and PDF forms, provided `TextField1` does not contain a value:

```
if ((TextField1.rawValue == NULL) || (TextField1.rawValue == ""))
{
    return True;
}
```

- Guides do not support multiple instances of a message box to display synchronously. This behavior is different from Acrobat and Adobe Reader which support the display of multiple instances. The result is that, on Guides, only the first message box in a sequence displays to the user.

For example, if the following script is placed on the exit event of a text field:

```
xfa.host.messageBox("Read this first.");
xfa.host.messageBox("Then read this.");
```

On a runtime Guide, when a user exits the text field, only the first message displays.

Limitations when working with data sources

If your form is bound to a data source, you should consider the following restrictions:

- Only simple binding against a data value is supported. Complex binding is not supported.

- If you use data binding, you must use explicit binding. For example, if a subform is using normal binding, the name of the subform must appear in the data source.
- Subform objects that can grow in response to the amount of data must be explicitly bound to elements in the data.
- The concept of form state is not supported. Only states represented in the data are preserved.

For more information on data binding and form state, see [LiveCycle Designer ES2 Help](#).

Chapter 4: Object-Level Scripting Support

This section lists only those scripting objects that are available for HTML forms and Guides, along with any scripting properties and methods that each object supports for those output formats. For a complete, alphabetic list of scripting properties and methods, and their availability in HTML forms and Guides, see “[Scripting Properties](#)” on page 20 and “[Scripting Methods](#)” on page 29.

***Note:** Scripting properties and methods listed as Not applicable do not have any logical meaning within the specified output format. For example, Guides do not contain a concept of pages, so properties or methods related to information about form pages do not apply. These properties and methods will fail gracefully and will not generate errors or return exceptions.*

xfa

The xfa object corresponds to the following Designer scripting syntax:

```
xfa.*
```

Properties

Scripting property	Supported in HTML	Supported in Guides
className	no	yes

Methods

Scripting method	Supported in HTML	Supported in Guides
resolveNode	yes	yes
resolveNodes	yes	yes

data

The data object corresponds to the following Designer scripting syntax:

```
xfa.data.*
```

Properties

None

Methods

Scripting method	Supported in HTML	Supported in Guides
resolveNode	no	yes
resolveNodes	no	yes
saveXML	no	yes To save the root level data, use the following syntax: var originalData = xfa.data.saveXML();

datasets

The data object corresponds to the following Designer scripting syntax:

`xfa.data.datasets.*`

Properties

None

Methods

Scripting method	Supported in HTML	Supported in Guides
loadXML	no	yes The optional parameters on this method are not supported in Guides. This method is supported only at the root level. You cannot load data at a subform level. To save the root level data, use the following syntax: var originalData = xfa.data.saveXML(); To later reload the data, use the following syntax: xfa.datasets.loadXML(originalData);
resolveNode	no	yes
resolveNodes	no	yes

eventpseudomodel

The eventpseudomodel object corresponds to the following Designer scripting syntax:

`xfa.event.*`

Properties

Scripting property	Supported in HTML	Supported in Guides
change	no	yes
className	no	yes
name	no	yes
newText	no	yes

Methods

None

hostpseudomodel

The hostpseudomodel object corresponds to the following Designer scripting syntax:

```
xfa.host.*
```

Properties

Scripting property	Supported in HTML	Supported in Guides
calculationsEnabled	yes	yes
className	no	yes
currentPage	yes	Not applicable
name	yes	yes
numPages	yes	Not applicable
validationsEnabled	yes	yes

Methods

Scripting method	Supported in HTML	Supported in Guides
gotoURL	yes	no
messageBox	yes	yes
pageDown	yes	no
pageUp	yes	no
print	no	no
resetData	yes	yes
setFocus	yes	no

form

The form object corresponds to the following Designer scripting syntax:

```
xfa.form.*
```

Supported child objects

“[field](#)” on page 15

“[subform](#)” on page 16

Properties

Scripting property	Supported in HTML	Supported in Guides
className	no	yes
nodes	no	yes

Methods

Scripting method	Supported in HTML	Supported in Guides
execCalculate	yes	yes
execInitialize	yes	yes
execValidate	yes	yes
recalculate	no	yes
remerge	no	yes
resolveNode	yes	yes
resolveNodes	yes	yes

field

The field object corresponds to the following Designer scripting syntax:

*field_name.**

Properties

Scripting property	Supported in HTML	Supported in Guides
access	yes	yes
all	no	yes
borderColor	yes	Not applicable
borderWidth	yes	Not applicable
classAll	no	yes
className	no	yes
errorText	yes	no <i>Note: Guides support XFA 2.8.</i>
fillColor	yes	Not applicable
fontColor	yes	Not applicable
formattedValue	yes	yes
h	yes	Not applicable
index	yes	yes
isContainer	no	yes
mandatory	yes	yes
mandatoryMessage	yes	yes
name	yes	yes
nodes	no	yes
parent	yes	yes
parentSubform	no	yes

Scripting property	Supported in HTML	Supported in Guides
presence	yes	yes
somExpression	no	yes
validationMessage	yes	yes
w	yes	Not applicable
x	yes	Not applicable
y	yes	Not applicable

Methods

Scripting method	Supported in HTML	Supported in Guides
addItem	yes	Drop-down lists and
boundItem	no	Drop-down lists only
clearItems	yes	Drop-down lists only
deleteItem	no	Drop-down lists only
execCalculate	yes	yes
execEvent	yes	yes
execInitialize	yes	yes
execValidate	yes	yes
getDisplayItem	no	Drop-down lists only
getItemState	no	Drop-down lists only
getSaveItem	no	Drop-down lists only
resolveNode	yes	yes
resolveNodes	yes	yes
setItemState	no	Drop-down lists only

subform

The subform object corresponds to the following Designer scripting syntax:

*subform_name.**

Supported child objects

“[instanceManager](#)” on page 17

“[occur](#)” on page 18

Properties

Scripting property	Supported in HTML	Supported in Guides
all	no	yes
borderColor	yes	Not applicable
borderWidth	yes	Not applicable
classAll	no	yes
className	no	yes
fillColor	yes	Not applicable
index	no	yes
instanceIndex	no	yes
isContainer	no	yes
name	yes	yes
nodes	no	yes
parent	yes	yes
presence	yes	yes
somExpression	no	yes
validationMessage	yes	yes

Methods

Scripting method	Supported in HTML	Supported in Guides
execCalculate	yes	yes
execEvent	yes	yes
execInitialize	yes	yes
execValidate	yes	yes
getInvalidObjects	yes	no <i>Note: Guides support XFA 2.8.</i>
resolveNode	yes	yes
resolveNodes	yes	yes

instanceManager

The instanceManager object corresponds to the following Designer scripting syntax:

`subform_name.instanceManager.*`

Properties

Scripting property	Supported in HTML	Supported in Guides
className	no	yes
count	yes	yes
isContainer	no	yes
max	yes	yes
min	yes	yes
name	yes	yes
somExpression	no	yes

Methods

Scripting method	Supported in HTML	Supported in Guides
addInstance	yes	yes
insertInstance	yes	yes
instanceCount	yes	yes
moveInstance	yes	yes
removeInstance	yes	yes
setInstances	yes	yes

OCUR

The occur object corresponds to the following Designer scripting syntax:

```
subform_name.occur.*
```

Properties

Scripting property	Supported in HTML	Supported in Guides
initial	no	yes
max	yes	yes
min	yes	yes

Methods

None

Chapter 5: Scripting Events

This section lists support for XML Form Object Model scripting events in HTML forms and Guides.

For more information on scripting events and event timing, see [LiveCycle Designer ES2 Scripting Basics](#).

Scripting event	Supported in HTML	Supported in Guides
calculate	yes	yes
change	Drop-down lists only	yes
click	yes	yes
docClose	no	no
docReady	no	no
enter	yes	yes
exit	yes	yes
form:ready	no	yes
full	no	no
indexChange	no	no
initialize	yes	yes
layout:ready	no	no
mouseDown	yes	yes
mouseEnter	no	yes
mouseExit	no	yes
mouseUp	yes	yes
postOpen	no	no
postPrint	no	no
postSave	no	no
postSign	no	no
postSubmit	no	no
preOpen	no	Drop-down lists only
prePrint	yes	no
preSave	no	no
preSign	no	no
preSubmit	Submit buttons only	
validate	yes	yes

Chapter 6: Scripting Properties

This section lists support for XML Form Object Model scripting properties in HTML forms and Guides.

For more information on scripting properties, see [LiveCycle Designer ES2 Scripting Reference](#).

Note: Items marked as not applicable will not generate an error or cause a script to fail; however, they do not produce any effect on the rendered form or Guide.

Scripting property	Supported in HTML	Supported in Guides
#text	no	no
{default}	no	no
access	yes	yes
accessKey	no	no
action	no	no
activity	no	no
addRevocationInfo	no	no
aliasNode	no	no
all	no	yes
allowMacro	no	no
allowNeutral	no	no
allowRichText	no	no
anchorType	no	no
appType	no	no
archive	no	no
aspect	no	no
baselineShift	no	no
bind	no	no
binding	no	no
blankOrNotBlank	no	no
bofAction	no	no
borderColor	yes	Not applicable
borderWidth	yes	Not applicable
bottomInset	no	no
break	no	no
calculationsEnabled	yes	yes
cancelAction	no	no

Scripting property	Supported in HTML	Supported in Guides
cap	no	no
change	no	yes
charEncoding	no	no
checksum	no	no
circular	no	no
classAll	no	yes
classId	no	no
classIndex	no	no
className	no	yes
codeBase	no	no
codeType	no	no
colSpan	no	no
columnWidths	no	no
commandType	no	no
commitKey	no	no
commitOn	no	no
connection	no	no
contains	no	no
content	no	no
contentType	no	no
count	yes	yes
credentialServerPolicy	no	no
crfSign	no	no
cSpace	no	no
currentPage	yes	Not applicable
currentRecordNumber	no	no
currentValue	no	no
cursorLocation	no	no
cursorType	no	no
data	no	no
dataColumnCount	no	no
dataDescription	no	no
dataEncipherment	no	no
dataLength	no	no

Scripting property	Supported in HTML	Supported in Guides
dataPrep	no	no
dataRowCount	no	no
db	no	no
decipherOnly	no	no
delayedOpen	no	no
delimiter	no	no
digitalSignature	no	no
disable	no	no
disableAll	no	no
duplexImposition	no	no
editValue	no	no
embedPDF	no	no
encipherOnly	no	no
endChar	no	no
eofAction	no	no
errorCorrectionLevel	no	no
errorText	yes	no <i>Note: Guides support XFA 2.8.</i>
excludeAllCaps	no	no
excludelInitialCap	no	no
executeType	no	no
fillColor	yes	Not applicable
fontColor	yes	Not applicable
fontHorizontalScale	no	no
fontVerticalScale	no	no
format	no	no
formatMessage	no	no
formattedValue	yes	yes
formatTest	no	no
fracDigits	no	no
from	no	no
fullText	no	no
h	yes	Not applicable
hAlign	no	no

Scripting property	Supported in HTML	Supported in Guides
hand	no	no
highlight	no	no
href	no	no
hScrollPolicy	no	no
hyphenate	no	no
id	no	no
imagingBBox	no	no
index	yes	yes
initial	no	yes
initialNumber	no	no
input	no	no
instanceIndex	no	yes
intact	no	no
inverted	no	no
isContainer	no	yes
isDefined	no	no
isNull	no	no
join	no	no
kerningMode	no	no
keyAgreement	no	no
keyCertSign	no	no
keyDown	no	no
keyEncipherment	no	no
labelRef	no	no
ladderCount	no	no
language	no	no
layout	no	no
leadDigits	no	no
leader	no	no
leftInset	no	no
length	no	no
letterSpacing	no	no
lineHeight	no	no
lineThrough	no	no

Scripting property	Supported in HTML	Supported in Guides
lineThroughPeriod	no	no
listen	no	no
locale	no	no
lockType	no	no
long	no	no
mandatory	yes	yes
mandatoryMessage	yes	yes
marginLeft	no	no
marginRight	no	no
mark	no	no
match	no	no
max	yes	yes
maxChars	no	no
maxH	no	no
maxLength	no	no
maxW	no	no
mergeMode	no	no
min	yes	yes
minH	no	no
minW	no	no
model	no	no
modifier	no	no
moduleHeight	no	no
moduleWidth	no	no
multiLine	no	no
name	yes	yes
newContentType	no	no
newText	no	yes
next	no	no
nodes	no	yes
nonRepudiation	no	no
ns	no	no
nullTest	no	no
numbered	no	no

Scripting property	Supported in HTML	Supported in Guides
numberOfCells	no	no
numPages	yes	Not applicable
oddOrEven	no	no
oneOfChild	no	no
open	no	no
operation	no	no
orientation	no	no
output	no	no
override	no	no
pagePosition	no	no
parent	yes	yes
parentSubform	no	yes
passwordChar	no	no
permissions	no	no
placement	no	no
platform	no	no
posture	no	no
presence	yes	yes
preserve	no	no
prevContentType	no	no
previous	no	no
prevText	no	no
printCheckDigit	no	no
priority	no	no
pushCharacterCount	no	no
radius	no	no
radixOffset	no	no
rate	no	no
rawValue	yes	yes
ready	no	no
recordsAfter	no	no
recordsBefore	no	no
reenter	no	no
ref	no	no

Scripting property	Supported in HTML	Supported in Guides
relation	no	no
relevant	no	no
remainCharacterCount	no	no
reserve	no	no
restoreState	no	no
rightInset	no	no
role	no	no
rotate	no	no
rowColumnRatio	no	no
runAt	no	no
save	no	no
savedValue	no	no
scope	no	no
scriptTest	no	no
selectedIndex	no	no
selEnd	no	no
selStart	no	no
server	no	no
shape	no	no
shift	no	no
short	no	no
signatureType	no	no
size	no	no
slope	no	no
soapFaultCode	no	no
soapFaultString	no	no
somExpression	no	yes
spaceAbove	no	no
spaceBelow	no	no
startAngle	no	no
startChar	no	no
startNew	no	no
stateless	no	no
stock	no	no

Scripting property	Supported in HTML	Supported in Guides
stroke	no	no
sweepAngle	no	no
tabDefault	no	no
tabStops	no	no
target	no	no
targetType	no	no
textEncoding	no	no
textEntry	no	no
textIndent	no	no
textLocation	no	no
thickness	no	no
this	no	no
timeout	no	no
timeStamp	no	no
title	no	no
topInset	no	no
trailer	no	no
transferEncoding	no	no
transient	no	no
truncate	no	no
type	no	no
typeface	no	no
underline	no	no
underlinePeriod	no	no
upsMode	no	no
url	no	no
urlPolicy	no	no
usage	no	no
use	no	no
usehref	no	no
uuid	no	no
validationMessage	yes	yes
validationsEnabled	yes	yes
vAlign	no	no

Scripting property	Supported in HTML	Supported in Guides
value	no	no
valueRef	no	no
variation	no	no
version	no	no
vScrollPolicy	no	no
w	yes	Not applicable
weight	no	no
wideNarrowRatio	no	no
wordCharacterCount	no	no
wordSpacingMaximum	no	no
wordSpacingMinimum	no	no
wordSpacingOptimum	no	no
x	yes	Not applicable
xdpContent	no	no
y	yes	Not applicable

Chapter 7: Scripting Methods

This section lists support for XML Form Object Model scripting methods in HTML forms and Guides.

For more information on scripting methods, see [LiveCycle Designer ES2 Scripting Reference](#).

Note: Items marked as not applicable will not generate an error or cause a script to fail; however, they do not produce any effect on the rendered form or Guide.

Scripting method	Supported in HTML	Supported in Guides
absPage	no	no
absPageCount	no	no
absPageCountInBatch	no	no
absPageInBatch	no	no
absPageSpan	no	no
addInstance	yes	yes
addItem	yes	Drop-down lists only
addNew	no	no
append	no	no
applyXSL	no	no
assignNode	no	no
beep	no	no
boundItem	no	Drop-down lists only
cancel	no	no
cancelBatch	no	no
clear	no	no
clearErrorList	no	no
clearItems	yes	Drop-down lists only
clone	no	no
close	no	no
createNode	no	no
currentDateTime	no	no
delete	no	no
deleteItem	no	Drop-down lists only
documentCountInBatch	no	no
documentInBatch	no	no
emit	no	no

Scripting method	Supported in HTML	Supported in Guides
enumerate	no	no
evaluate	no	no
execCalculate	yes	yes
execEvent	yes	yes
execInitialize	yes	yes
execute	no	no
execValidate	yes	yes
exportData	no	no
first	no	no
formNodes	no	no
getAttribute	no	no
getDelta	no	no
getDeltas	no	no
getDisplayItem	no	Drop-down lists only
getElement	no	no
getFocus	no	no
getInvalidObjects	yes	no <i>Note: Guides support XFA 2.8.</i>
getItemState	no	Drop-down lists only
getSaveItem	no	Drop-down lists only
gotoRecord	no	no
gotoURL	yes	no
hasDataChanged	no	no
importData	no	no
insert	no	no
insertInstance	yes	yes
instanceCount	yes	yes
isBOF	no	no
isCompatibleNS	no	no
isEOF	no	no
isPropertySpecified	no	no
isRecordGroup	no	no
item	no	no
last	no	no

Scripting method	Supported in HTML	Supported in Guides
loadXML	no	yes The optional parameters on this method are not supported in Guides. This method is supported only at the root level. You cannot load data at a subform level. To save the root level data, use the following syntax: var originalData = xfa.data.saveXML(); To later reload the data, use the following syntax: xfa.datasets.loadXML(originalData);
messageBox	yes	yes
metadata	no	no
moveCurrentRecord	no	no
moveInstance	yes	yes
namedItem	no	no
next	no	no
open	no	no
openList	no	no
page	no	no
pageContent	no	no
pageCount	no	no
pageDown	yes	no
pageSpan	no	no
pageUp	yes	no
previous	no	no
print	no	no
recalculate	no	yes
record	no	no
relayout	no	no
relayoutPageArea	no	no
remerge	no	yes
remove	no	no
removeAttribute	no	no
removeInstance	yes	yes
requery	no	no
reset	no	no
resetData	yes	no

Scripting method	Supported in HTML	Supported in Guides
resolveNode	no	yes
resolveNodes	no	yes
response	no	no
restore	no	no
resync	no	no
saveFilteredXML	no	no
saveXML	no	yes
selectedMember	no	no
setAttribute	no	no
setElement	no	no
setFocus	yes	no
setInstances	yes	yes
setItems	no	no
setItemState	no	Drop-down lists only
sheet	no	no
sheetCount	no	no
sheetCountInBatch	no	no
sheetInBatch	no	no
sign	no	no
update	no	no
updateBatch	no	no
verify	no	no

