LiveCycle® Designer ES Scripting Reference

Adobe® LiveCycle® Designer ES

Version 8.1

July 2007

Adobe® LiveCycle® Designer ES 8.1 Scripting Reference for Microsoft® Windows®
Edition 3.1, July, 2007

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

Contents

Preface **15**
What's in this guide? 15
Who should read this guide? 15
How this guide is organized 15
Related documentation 16

1 Overview **17**
Subforms and containers 17

2 XML Form Object Model Class Hierarchy **18**
object class 18
list class 18
treeList class 19
tree class 19
node class 20
container class 22
content class 23
model class 23
textNode class 24

3 Scripting Objects **25**
arc 25
area 26
assist 26
barcode 27
bind 29
bindItems 30
bookend 30
boolean 31
border 32
break 33
breakAfter 34
breakBefore 35
button 36
calculate 37
caption 37
certificate 38
certificates 39
checkButton 40
choiceList 40
color 41
comb 42
command 43
connect 43
connectionSet 44
connectString 44
contentArea 45

corner....46
dataGroup....47
dataModel....47
dataValue....48
dataWindow....48
date....49
dateTime....50
dateTimeEdit....51
decimal....51
defaultUi....52
delete....53
delta....53
deltas....54
desc....54
digestMethod....55
digestMethods....56
draw....56
dSigData....58
edge....59
encoding....59
encodings....60
encrypt....60
event....61
eventPseudoModel....62
exclGroup....63
exData....66
execute....67
exObject....68
extras....68
field....69
fill....72
filter....73
float....74
font....75
form....76
format....77
handler....77
hostPseudoModel....78
image....80
imageEdit....81
insert....81
instanceManager....82
integer....83
issuers....83
items....84
keep....85
keyUsage....85
layoutPseudoModel....87
line....88
linear....88
manifest....89

map 90
margin 90
mdp 91
medium 92
message 93
numericEdit 93
occur 94
oid 95
oids 96
operation 96
overflow 97
packet 98
pageArea 98
pageSet 99
para 100
password 101
passwordEdit 102
pattern 102
picture 103
proto 104
query 105
radial 105
reason 106
reasons 106
recordSet 107
rectangle 108
ref 108
rootElement 109
script 110
select 111
setProperty 111
signature 112
signatureProperties (deprecated) 113
signaturePseudoModel 113
signData 114
signing 114
soapAction 115
soapAddress 116
solid 116
source 117
sourceSet 118
speak 118
stipple 119
subform 120
subformSet 122
subjectDN 123
subjectDNs 123
submit 124
template 125
text 125
textEdit 126

time ... 127
timeStamp ... 128
toolTip ... 128
traversal ... 129
traverse ... 130
ui ... 130
update ... 131
uri ... 131
user ... 132
validate ... 132
value ... 133
variables ... 134
wsdlAddress ... 135
wsdlConnection ... 135
xfa ... 136
xmlConnection ... 136
xsdConnection ... 137

4 Scripting Properties ... 138

#text ... 138
{default} ... 138
access ... 139
accessKey ... 140
action ... 141
activity ... 142
addRevocationInfo ... 145
after ... 146
afterTarget ... 148
aliasNode ... 148
all ... 149
allowMacro ... 149
allowNeutral ... 150
allowRichText ... 151
anchorType ... 152
appType ... 153
archive ... 154
aspect ... 154
baselineShift ... 155
before ... 156
beforeTarget ... 158
bind ... 158
binding ... 159
blank (deprecated) ... 159
blankOrNotBlank ... 160
bofAction ... 161
bookendLeader ... 162
bookendTrailer ... 162
borderColor ... 163
borderWidth ... 164
bottomInset ... 164
break ... 165

calculationsEnabled ... 165
cap ... 166
change ... 167
charEncoding ... 167
checksum ... 169
circular ... 170
classAll ... 171
classId ... 172
classIndex ... 172
className ... 173
codeBase ... 173
codeType ... 174
colSpan ... 174
columnWidths ... 175
commandType ... 176
commitKey ... 176
commitOn ... 177
connection ... 178
contains ... 179
content ... 179
contentType ... 180
context (deprecated) ... 181
count ... 182
credentialServerPolicy ... 182
crlSign ... 183
cSpace ... 183
currentPage ... 184
currentRecordNumber ... 185
currentValue ... 185
cursorLocation ... 186
cursorType ... 186
data ... 187
dataColumnCount ... 188
dataDescription ... 189
dataEncipherment ... 189
dataLength ... 190
dataPrep ... 191
dataRowCount ... 192
db ... 192
decipherOnly ... 193
delayedOpen ... 193
delimiter ... 194
digitalSignature ... 194
disable ... 195
editValue ... 196
embedPDF ... 196
encipherOnly ... 197
endChar ... 197
eofAction ... 198
errorCorrectionLevel ... 198
executeType ... 199

fillColor ... 200
fontColor ... 201
format ... 201
formatMessage ... 202
formattedValue ... 203
formatTest ... 204
fracDigits ... 205
from ... 205
fullText ... 206
h ... 206
hAlign ... 207
hand ... 208
highlight ... 209
href ... 210
hScrollPolicy ... 211
id ... 212
imagingBBox ... 212
index ... 213
initial ... 213
initialNumber ... 214
input ... 215
instanceIndex ... 215
intact ... 216
inverted ... 217
isContainer ... 217
isDefined ... 218
isNull ... 219
join ... 219
keyAgreement ... 220
keyCertSign ... 221
keyDown ... 221
keyEncipherment ... 222
labelRef ... 223
language ... 223
layout ... 224
leadDigits ... 225
leader ... 225
leftInset ... 226
length ... 227
lineHeight ... 228
lineThrough ... 228
lineThroughPeriod ... 229
locale ... 230
lockType ... 230
long ... 231
mandatory ... 232
mandatoryMessage ... 232
marginLeft ... 233
marginRight ... 234
mark ... 234
match ... 235

max ... 236
maxChars ... 237
maxH ... 238
maxLength ... 238
maxW ... 239
min ... 240
minH ... 240
minW ... 241
model ... 242
modifier ... 242
moduleHeight ... 243
moduleWidth ... 244
multiLine ... 244
name ... 245
newContentType ... 246
newText ... 247
next ... 247
nodes ... 248
nonRepudiation ... 249
ns ... 250
nullTest ... 250
numbered ... 251
numberOfCells ... 252
numPages ... 253
oddOrEven ... 253
oneOfChild ... 254
open ... 255
operation ... 256
orientation ... 258
output ... 258
overflowLeader ... 259
overflowTarget ... 260
overflowTrailer ... 260
overline ... 261
overlinePeriod ... 261
override ... 262
pagePosition ... 264
parent ... 264
parentSubform ... 265
passwordChar ... 266
permissions ... 266
placement ... 267
platform ... 268
posture ... 268
presence ... 269
preserve ... 271
prevContentType ... 271
previous ... 272
prevText ... 273
printCheckDigit ... 274
priority ... 274

radius ... 275
radixOffset ... 276
rate ... 276
rawValue ... 277
ready ... 278
recordsAfter ... 279
recordsBefore ... 279
reenter ... 280
ref ... 281
relation ... 281
relevant ... 282
reserve ... 283
restoreState ... 284
rightInset ... 285
role ... 285
rotate ... 286
rowColumnRatio ... 287
runAt ... 287
save ... 288
savedValue ... 289
scope ... 289
scriptTest ... 290
selectedIndex ... 291
selEnd ... 292
selStart ... 293
server ... 293
shape ... 294
shift ... 294
short ... 295
signatureType ... 296
size ... 296
slope ... 297
soapFaultCode ... 298
soapFaultString ... 298
somExpression ... 299
spaceAbove ... 299
spaceBelow ... 300
startAngle ... 300
startChar ... 301
startNew ... 302
stateless ... 302
stock ... 303
stroke ... 304
sweepAngle ... 305
tabDefault ... 305
tabStops ... 306
target ... 307
targetType ... 308
textEncoding ... 308
textEntry ... 310
textIndent ... 311

textLocation ... 311
thickness ... 312
this ... 313
timeout ... 314
timeStamp ... 314
title ... 315
topInset ... 315
trailer ... 316
transferEncoding ... 317
transient ... 317
truncate ... 318
type ... 318
typeface ... 323
underline ... 323
underlinePeriod ... 324
upsMode ... 325
url ... 325
urlPolicy ... 326
usage ... 326
use ... 327
usehref ... 328
uuid ... 330
validationMessage ... 330
validationsEnabled ... 331
vAlign ... 331
value ... 332
valueRef ... 334
variation ... 334
version ... 335
vScollPolicy ... 336
w ... 336
weight ... 337
wideNarrowRatio ... 338
x ... 339
xdpContent ... 339
y ... 340

5 Scripting Methods ... 342

absPage ... 342
absPageCount ... 342
absPageCountInBatch ... 343
absPageInBatch ... 344
absPageSpan ... 344
addInstance ... 345
addItem ... 345
addNew ... 346
append ... 347
applyXSL ... 347
assignNode ... 348
beep ... 349
boundItem ... 349

cancel ... 350
cancelBatch ... 350
clear ... 351
clearErrorList ... 352
clearItems ... 352
clone ... 353
close ... 353
createNode ... 354
delete ... 355
deleteItem ... 356
documentCountInBatch ... 356
documentInBatch ... 357
emit ... 357
enumerate ... 358
evaluate ... 358
execCalculate ... 358
execEvent ... 359
execInitialize ... 360
execute ... 360
execValidate ... 361
exportData ... 362
first ... 363
formNodes ... 363
getAttribute ... 364
getDelta ... 364
getDeltas ... 365
getDisplayItem ... 365
getElement ... 366
getFocus ... 366
getItemState ... 367
getSaveItem ... 367
gotoRecord ... 368
gotoURL ... 368
h ... 369
hasDataChanged ... 370
importData ... 370
insert ... 371
insertInstance ... 371
isBOF ... 372
isCompatibleNS ... 373
isEOF ... 373
isPropertySpecified ... 374
isRecordGroup ... 375
item ... 375
last ... 376
loadXML ... 377
messageBox ... 377
metadata ... 379
moveCurrentRecord ... 380
moveInstance ... 381
namedItem ... 381

next ... 382
open ... 382
openList ... 383
page ... 384
pageContent ... 384
pageCount ... 386
pageDown ... 386
pageSpan ... 387
pageUp ... 388
previous ... 388
print ... 389
recalculate ... 391
record ... 391
relayout ... 392
relayoutPageArea ... 393
remerge ... 393
remove ... 394
removeAttribute ... 394
removeInstance ... 395
requery ... 396
reset ... 396
resetData ... 397
resolveNode ... 398
resolveNodes ... 398
response ... 399
restore ... 400
resync ... 401
saveFilteredXML ... 401
saveXML ... 402
selectedMember ... 402
setAttribute ... 403
setElement ... 403
setFocus ... 404
setInstances ... 405
setItemState ... 405
sheet ... 406
sheetCount ... 407
sheetCountInBatch ... 407
sheetInBatch ... 407
sign ... 408
update ... 409
updateBatch ... 410
verify ... 410
w ... 411
x ... 412
y ... 412

6 Understanding the XML Form Object Model 414
XML Form Object Model DOMs 415
connectionSet Model 415
Data Model 416
Event Model 416
Form Model 417
Host Model 417
Layout Model 418
Signature Model 418
sourceSet Model 418
XFA Model 419
A JavaScript Examples 420
Referencing objects 420
Creating a node in the data model 422
Manipulating instances of a subform 424
Getting or setting object values 425
Working with page numbers and page counts 426
Concatenating data values 427
Calculating totals 428
Changing the background color 428
Populating a drop-down list 430
Saving a form 431
Making an object visible or invisible 432
Using radio buttons and check boxes 433
Determining that a form has changed 433
Disabling all form fields 434
Index 435

Preface

The Adobe® XML Form Object Model, based on the Adobe XML Forms Architecture, represents the underlying technology behind the Adobe XML form solution and incorporates XML architectural concepts such as Document Object Model (DOM). Using this technology, form developers can create complex and flexible form-based applications for use with the client or the server.

Adobe LiveCycle® Designer ES enables a form developer to build intelligent forms using only the options provided in the LiveCycle Designer ES graphical interface.

By scripting against the XML Form Object Model, the form developer may further manipulate all aspects of the form, extending the functionality of the form beyond what is available through the LiveCycle Designer ES interface. For example, you might use a simple calculation to automatically update costs on a purchase order, or you might use scripting to modify the appearance of your form in response to the locale of the user.

Scripting is supported in two languages: FormCalc, a calculation language created by Adobe Systems Incorporated, and JavaScript™, a powerful and popular scripting language.

What's in this guide?

This guide describes the different objects available in the XML Form Object Model for scripting and their associated properties and methods. It briefly describes the different models that the objects belong to. It also provides scripting examples that illustrate how to use the properties and methods to perform various tasks.

Who should read this guide?

This guide is intended for form developers interested in extending their form designs using the XML Form Object Model and scripting. The basics of creating form designs that incorporate scripting are provided in *LiveCycle Designer ES Help*.

How this guide is organized

This guide is organized into chapters based on the various models available in the XML Form Object model:

- The Overview chapter introduces the XML Form Object Model and briefly explains key concepts, such as subforms and containers.
- The XML Form Object Model Class Hierarchy chapter provides a list of the base classes from which all the objects are derived. Each object is described in a subsequent chapter and is linked back to its associated class.
- The Scripting Objects chapter provides an alphabetical reference to all the objects available to all models within the XML Form Object Model. For each object, a brief description of the associated properties and methods is provided, along with links to more detailed descriptions in the "Properties" and "Methods" chapters. In addition, each object has an accompanying table that shows the parent

and child object hierarchy in relation to the current object. This parent/child hierarchy is meant to provide a mechanism for quickly determining the scripting syntax required to reference a particular object.

- The Scripting Properties chapter provides an alphabetical reference to all the properties available to all models within the XML Form Object Model, and outlines the models and objects that each property applies to.
- The Scripting Methods chapter provides an alphabetical reference to all the methods available to all models within the XML Form Object Model, and outlines the models and objects that each method applies to.
- The Understanding the XML Form Object Model chapter describes the XML Form Object Model and how the different models interact with one another. It describe the purpose of the models and lists the objects contained within each model.
- The JavaScript Examples chapter provides illustrative examples of the properties and methods that are supported in this scripting environment. Each example includes a hyperlinked list of the properties and methods that it uses.

Related documentation

In addition to this guide, Adobe provides additional documentation on specific scripting topics.

For information about	**Refer to**
Creating forms using LiveCycle Designer ES	*LiveCycle Designer Help*
The FormCalc scripting language and its functions	*LiveCycle Designer ES FormCalc Reference* guide at http://www.adobe.com/go/learn_lc_formCalc
JavaScript and the object model in Adobe Acrobat® Professional and Acrobat Standard	*JavaScript for Acrobat API Reference* located at http://www.adobe.com/go/learn_lc_AcrobatDeveloper

1 Overview

The Adobe XML Form Object Model provides a form design-based approach to creating forms that distinguishes between the form's layout and content. A form design, the design-time version of a form, specifies a set of layout, data capture, and presentation rules for the form. The content is the application data; any format of XML data is acceptable. Though they are often packaged together, the form design and the data are separate entities and are handled separately by the object model.

Typically, forms are created by using one of the following methods:

- Using LiveCycle Designer ES, the interactive forms authoring tool
- Machine-generated, creating a form based on some input, such as an XML schema

This guide focuses on the form developer who is using LiveCycle Designer ES to create forms.

Subforms and containers

In LiveCycle Designer ES, forms are documents that are created from a hierarchy of optionally repeating building-blocks known as *subforms*. Each subform controls a portion of the overall structure, presentation, and behavior of the form. Individual subforms enclose a combination of objects that produce fillable regions (fields) and non-fillable regions (draws). Subforms may also contain other subforms, and each subform may have properties that determine how and when the subform is instantiated into a constructed form.

Within each form is a concept of a container. A *container* is an object that holds data or values. Simple containers, those that are not capable of holding other containers or objects, include fields (text, numeric, buttons) and drawn objects (text, circle, line). All containers capable of holding other containers as well as non-container objects are considered *complex containers*. Subforms are an example of a complex container.

2 XML Form Object Model Class Hierarchy

The XML Form Object Model consists of models that each contain a set of objects. Each object is derived from one of the set of classes that define common properties and methods. An object, in turn, inherits these common properties and methods but may also add properties and methods that are unique to that object, relative to other objects derived from the same class.

As with traditional class structures, each class inherits properties and methods from its parent class. Objects, in turn, inherit from the parent class from which they derive.

Each model uses a hierarchy of objects. Objects do not inherit properties and methods from other objects, but instead inherit directly from the class hierarchy. The hierarchy of objects within a model represents the XML structure of that model.

object class

The `object` class is the base class from which all other classes, objects, and models are either directly or indirectly derived.

Class hierarchy

Parent class	Current class	Objects derived from this class
None	`object`	dataWindow eventPseudoModel hostPseudoModel layoutPseudoModel signaturePseudoModel

Properties

Name	Description	Type	Access
className	Determines the name of the class of this object.	String	Get

Methods

None

list class

The `list` class represents a list of nodes.

Class hierarchy

Parent class	Current class	Objects derived from this class
object	`list`	None

Properties

Name	Description	Type	Access
length	Specifies the number of objects in the list.	Integer	Read

Methods

Name	Description	Returns
append	Appends a node to the end of the node list.	Empty
insert	Inserts a node before a specific node in the node list.	Empty
item	Describes a zero-based index into the collection.	Object
remove	Removes a node from the node list.	Empty

treeList class

The `treeList` class represents a list of tree nodes.

Class hierarchy

Parent class	Current class	Objects derived from this class
list	`treeList`	None

Properties

None

Methods

Name	Description	Returns
namedItem	Gets the first child of this node with the given name.	Object

tree class

The `tree` class represents the structure from which the node class is derived.

Class hierarchy

Parent class	Current class	Objects derived from this class
object	`tree`	None

Properties

Name	Description	Type	Access
all	Returns a collection of like-named, in-scope nodes.	Object	Read
classAll	Returns a collection of like-class, in-scope nodes.	Object	Read

Name	Description	Type	Access
classIndex	Returns the position of this object in its collection of like-class, in-scope objects.	Integer	Read
index	Returns the position of this node in its collection of like-named, in-scope nodes.	Integer	Read
name	Specifies an identifier that may be used to specify this object or event in script expressions.	String	Read /Write
nodes	Returns a list of all child objects of the current object.	Object	Read
parent	Returns the parent object of the current object.	Object	Read
somExpression	Reads the reference syntax expression for this node.	String	Read

Methods

Name	Description	Returns
resolveNode	Evaluates the specified reference syntax expression, beginning with the current XML form object model object, and returns the value of the object specified in the reference syntax expression.	Object
resolveNodes	Evaluates the specified reference syntax expression, beginning with the current XML form object model object, and returns the value of the object or objects specified in the reference syntax expression.	Object

node class

The `node` class represents the primary data type for XML Form Object Model objects.

Class hierarchy

Parent class	Current class	Objects derived from this class			
tree	node	arc	dateTimeEdit	keep	recordSet
		assist	defaultUi	keyUsage	rectangle
		barcode	desc	line	script
		bind	digestMethod	linear	setProperty
		bindItems	digestMethods	manifest	signature
		bookend	dSigData	map	signData
		border	edge	margin	signing
		break	encoding	mdp	solid
		breakAfter	encodings	medium	source
		breakBefore	encrypt	message	stipple
		button	event	numericEdit	subjectDN
		calculate	exclGroup	occur	subjectDNs
		caption	execute	oids	submit
		certificates	exObject	overflow	textEdit
		checkButton	extras	packet	timeStamp
		choiceList	fill	para	traversal
		color	filter	passwordEdit	traverse
		comb	font	pattern	ui
		command	format	picture	validate
		connect	image	proto	value
		corner	imageEdit	query	wsdlConnection
		dataGroup	instanceManager	radial	xmlConnection
		dataValue	issuers	reasons	xsdConnection
		dateTime	items		

Properties

Name	Description	Type	Access
id	Specifies a generic user-defined XML ID type.	String	Read /Write
isContainer	Specifies whether this object is a container object.	Boolean	Read
isNull	Indicates whether the current data value is the null value.	Boolean	Read
model	Specifies the model for the current object.	Object	Read
ns	Returns the namespace for the object.	String	Read
oneOfChild	Retrieves or sets that child object in the case where a parent object can only have one of a particular child object.	Object	Read /Write

Methods

Name	Description	Returns
applyXSL	Applies an XSL transformation to the XML representation of the current node. It is equivalent to calling `saveXML` and transforming the result with the specified XSL document.	String
assignNode	Evaluates the reference syntax expression using the current context and sets the value of the found node. If the node doesn't exist, it can be created.	Object
clone	Makes a copy of an object.	Object
getAttribute	Gets a specified property value.	String
getElement	Returns a specified object property.	Object
isPropertySpecified	Checks if a specific property has been defined for this node.	Boolean
loadXML	Loads and appends a specified XML document to the current object.	Empty
saveFilteredXML	Saves the current node to a string, but includes only a subset of the child nodes.	String
saveXML	Saves the current node to a string.	String
setAttribute	Sets the value of a specified property.	Empty
setElement	Sets a specified object to be the current object.	Empty

container class

The `container` class provides container objects for the form model.

Class hierarchy

Parent class	Current class	Objects derived from this class
node	`container`	area contentArea draw field pageArea pageSet subform subformSet variables

Properties

None

Methods

Name	Description	Returns
getDelta	Gets a delta script object for a specific property.	Object
getDeltas	Recursively gets all the deltas script objects for this container object and all its descendants.	Object

content class

The `content` class provides content objects for the form and template models. Form designs and completed forms are visually composed of objects that represent content, such as images and text.

Class hierarchy

Parent class	Current class	Objects derived from this class
node	`content`	boolean date dateTime decimal exData float integer text time

Properties

None

Methods

None

model class

The `model` class is the base class for the root objects of each model.

Class hierarchy

Parent class	Current class	Objects derived from this class
node	`model`	connectionSet dataModel form template sourceSet xfa

Properties

Name	Description	Type	Access
aliasNode	Specifies the object that is represented by the alias for this model.	Object	Read /Write
context (deprecated)	Specifies the current object, which is the starting object for the resolveNode and resolveNodes methods.	Object	Read /Write

Methods

Name	Description	Returns
clearErrorList	Removes all items from the current error log.	Empty
createNode	Creates a new node based on a valid class name.	Object
isCompatibleNS	Determines if a specified namespace is functionally equivalent, that is compatible, with the namespace of this model. It determines if the two namespaces are equivalent, even though the strings that represent them may not be identical.	Boolean

textNode class

The `textNode` class represents objects that store textual data directly instead of using the `#text` object derived from the node class.

Class hierarchy

Parent class	Current class	Objects derived from this class	
node	`textNode`	certificate connectString delete handler insert oid operation password reason ref	rootElement select soapAction soapAddress speak toolTip update uri user wsdlAddress

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

3 Scripting Objects

This section provides an alphabetical list of all objects supported in this scripting environment. For each object, there is a brief description of the associated properties and methods, along with links to detailed descriptions of the properties and methods.

In addition, each object has an accompanying table that shows the parent and child object hierarchy in relation to the current object. This parent and child hierarchy is meant to provide a mechanism for quickly determining the scripting syntax required to reference a particular object.

arc

The `arc` object describes an arc or an ellipse.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto value	`arc`	edge fill

Parent class

node class

Properties

Name	Description	Type	Access
circular	Enables you to convert an arc into a circle.	String	Read /Write
hand	Describes the justification of a line or edge.	String	Read /Write
startAngle	Specifies the angle where the beginning of the arc renders.	String	Read /Write
sweepAngle	Specifies the length of the arc as an angle.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

area

The `area` object represents the grouping of other container objects on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	area pageArea proto subform	`area`	desc extras

Parent class

container class

Properties

Name	Description	Type	Access
colSpan	Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

None

assist

The `assist` object supplies additional information about a container for users of interactive form applications.

It provides a means to specify the toolTip and behavior for a spoken prompt.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	draw exclGroup field proto	`assist`	speak toolTip

Parent class

node class

Properties

Name	Description	Type	Access
role	Specifies the role played by the parent container.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

barcode

The `barcode` object supplies the information required to display a barcode. This information includes the type of barcode and a set of options that varies from one type of barcode to another.

LiveCycle Designer ES can support two types of barcodes: hardware and software. However, an XFA application is not required to support any particular set of barcodes. Hardware barcodes are displayed by particular printers. The set of supported barcodes may vary depending on the display device, because some printers have built-in support for particular barcodes. Software barcodes are drawn stroke by stroke by the XFA application itself. When displaying on a screen, which is not accessible to barcode readers, an XFA application may also revert to displaying just a placeholder rather than an accurate barcode.

For each type of barcode there are usually two separate specifications, one for the barcode itself and one for the barcode's placement in relation to the physical page and to surrounding printed matter. The creator of the form design is responsible for ensuring that the barcode is placed correctly on the page. The XFA application is responsible for correctly rendering the barcode using the user data. The user data must be compatible with the barcode; that is, it must conform to the allowed character set and string length.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`barcode`	extras

Parent class

node class

Properties

Name	Description	Type	Access
charEncoding	Specifies the character encoding of the value that is encoded into a barcode.	String	Read /Write
checksum	Specifies an algorithm for the checksum to insert into the barcode.	String	Read /Write
dataColumnCount	Specifies an optional number of data columns to encode for supported barcodes. This property applies to two-dimensional (2D) barcodes only.	String	Read /Write
dataLength	Specifies the maximum number of characters for this instance of the barcode. This property applies to one-dimensional barcodes only.	String	Read /Write
dataPrep	Defines preprocessing that is applied to the data written in the barcode.	String	Read /Write
dataRowCount	Specifies an optional number of data rows to encode for supported barcodes. This property applies to 2D barcodes only.	String	Read /Write
endChar	Specifies an optional ending control character to append to barcode data.	String	Read /Write
errorCorrectionLevel	Specifies an optional error correction level to apply to supported barcodes. This property applies to 2D barcodes only.	String	Read /Write
moduleHeight	Determines the height of a set of bars used to encode one character of supplied text.	String	Read /Write
moduleWidth	Specifies different aspects of a barcode depending on the class of barcodes being used.	String	Read /Write
printCheckDigit	Specifies whether to print the check digits in the human-readable text.	String	Read /Write
rowColumnRatio	An optional ratio of rows to columns for supported 2D barcodes.	String	Read /Write
startChar	Specifies an optional starting control character to add to the beginning of the barcode data.	String	Read /Write
textLocation	Specifies the location of any text associated with the barcode.	String	Read /Write
truncate	Truncates the right edge of the barcode for supported formats.	String	Read /Write
type	Specifies the pattern used by an object.	String	Read /Write

Name	Description	Type	Access
upsMode	Represents the mode in a UPS Maxicode barcode.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
wideNarrowRatio	Specifies a ratio of wide bar to narrow bar in supported barcodes.	String	Read /Write

Methods

None

bind

The `bind` object controls the behavior of its parent object during merge operations.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model sourceSet Model	command exclGroup field proto subform	`bind`	picture

Parent class

node class

Properties

Name	Description	Type	Access
contentType	Specifies the type of content in the referenced document, expressed as a MIME type.	String	Read /Write
match	Controls the role played by enclosing an object in a data-binding (merge) operation.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write
transferEncoding	Specifies the encoding of binary content in the referenced document.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

bindItems

The `bindItems` object identifies a set of data nodes for binding.

The application of the `bindItems` object is a binding operation. The links between the list items and the referenced data are active. Any change to the data causes an immediate update to the list items.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	field proto	`bindItems`	ref

Parent class

node class

Properties

Name	Description	Type	Access
connection	Specifies the name of the associated connection control in the connection set.	String	Read /Write
labelRef	Resolves a data value for each data node in the set identified by the `ref` object.	String	Read /Write
valueRef	Resolves a data value for each data node in the set identified by the `ref` object.	String	Read /Write

Methods

None

bookend

The `bookend` object stores properties that identify optional subforms that bookend the contents of the parent subform.

The leader property identifies an optional `subform` or `subformSet` that is laid out first, before the contents of the parent container. The trailer property identifies an optional `subform` or `subformSet`

object that is laid out last, after the contents of the parent container. In this way, these properties bookend the contents of the parent container. This is true regardless of how many `contentArea` or `pageArea` objects the parent container spans.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform subformSet	`bookend`	none

Parent class

node class

Properties

Name	Description	Type	Access
leader	Specifies the `subform` or `subformSet` object to place at the top of a content or page area.	String	Read /Write
trailer	Specifies the `subform` or `subformSet` object to place at the bottom of a content or page area.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

boolean

The `boolean` object describes a single unit of data content representing a boolean logical value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model sourceSet Model	desc exObject extras items proto value variables	`boolean`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Boolean	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	Boolean	Read /Write

Methods

None

border

The `border` object describes the border surrounding an object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	checkButton choiceList dateTimeEdit draw exclGroup field imageEdit numericEdit passwordEdit proto signature subform textEdit	`border`	corner edge extras fill margin

Parent class

node class

Properties

Name	Description	Type	Access
break	Describes the constraints on moving to a new page or content area after rendering an object.	String	Read /Write
hand	Describes the justification of a line or edge.	String	Read /Write

Name	Description	Type	Access
presence	Specifies an object's visibility.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

break

The `break` object describes the constraints on moving to a new page or content area before or after rendering an object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform subformSet	`break`	extras

Parent class

node class

Properties

Name	Description	Type	Access
after	Specifies the constraints on moving to a new page or content area after rendering the subform.	String	Read /Write
afterTarget	Specifies the explicit destination page or content area for the after property.	String	Read /Write
before	Specifies the constraints on moving to a new page or content area before rendering the subform.	String	Read /Write
beforeTarget	Specifies the explicit destination page or content area for the before property.	String	Read /Write
bookendLeader	Specifies a subform to place into the current content area or page before any other content.	String	Read /Write
bookendTrailer	Identifies a subform to place into the current content area or page after any other content.	String	Read /Write

Name	Description	Type	Access
overflowLeader	Specifies the subform to place at the top of the content area or page when it is entered as a result of an overflow.	String	Read /Write
overflowTarget	Specifies the explicit content area that will be the transition target when the current content area or page area overflows.	String	Read /Write
overflowTrailer	Specifies the subform to place at the bottom of the content area or page when it overflows.	String	Read /Write
startNew	Determines whether it is necessary to start a new content area or page even when the current content area or page has the required name.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

breakAfter

The `breakAfter` object describes the conditional constraints on moving to a new page or content area after laying down the parent container. The breakAfter object is invoked after laying out the parent subform. The leaders or trailers are laid down before and after any jump that the breakAfter object mandates.

An optional script object associated with the `breakAfter` object determines whether it is respected. This `script` object defaults to the true condition, which means that `breakAfter` objects with no `script` object are always invoked.

The `breakAfter` object is functionally equivalent to the deprecated syntax of break.after and afterTarget.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform subformSet	`breakAfter`	script

Parent class

node class

Properties

Name	Description	Type	Access
leader	Specifies the `subform` or `subformSet` object to place at the top of a content or page area.	String	Read /Write
startNew	Determines whether it is necessary to start a new content area or page even when the current content area or page has the required name.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write
targetType	Specifies the constraints on moving to a new page or content area before laying out the parent subform.	String	Read /Write
trailer	Specifies the `subform` or `subformSet` object to place at the bottom of a content or page area.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

breakBefore

The `breakBefore` object describes the conditional constraints for moving to a new page or content area before laying down the parent container. The breakBefore object is invoked before laying out the parent subform. The leaders and trailers are laid down before and after any jump that the breakBefore object mandates.

An optional script object associated with the `breakBefore` object determines whether it is respected. This `script` object defaults to the true condition, which means that `breakBefore` objects with no `script` object are always invoked.

The breakBefore object is functionally equivalent to the deprecated syntax of break.before and beforeTarget.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform subformSet	`breakBefore`	script

Parent class

node class

Properties

Name	Description	Type	Access
leader	Specifies the `subform` or `subformSet` object to place at the top of a content or page area.	String	Read /Write
startNew	Determines whether it is necessary to start a new content area or page even when the current content area or page has the required name.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write
targetType	Specifies the constraints on moving to a new page or content area before laying out the parent subform.	String	Read /Write
trailer	Specifies the `subform` or `subformSet` object to place at the bottom of a content or page area.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

button

The `button` object describes a push-button control.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`button`	extras

Parent class

node class

Properties

Name	Description	Type	Access
highlight	Specifies the visual appearance of a button when activated by a user. All values support two states (up and down) except `push` which supports three states (up, down, and rollover).	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

calculate

The `calculate` object controls the calculation of a field's value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	exclGroup field proto subform	`calculate`	extras message script

Parent class

node class

Properties

Name	Description	Type	Access
override	When used with the calculate object, the `override` property indicates whether the field allows overrides to occur and disables or enables calculations. When used with the value object, the `override` property indicates whether a calculation override has occurred.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

caption

The `caption` object describes a descriptive label associated with a form design object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	draw exclGroup field proto	`caption`	extras font margin para value

Parent class

node class

Properties

Name	Description	Type	Access
placement	Specifies the placement of the caption.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write
reserve	A measurement value that specifies the height or width of the caption.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

certificate

The `certificate` object holds a certificate.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	encrypt issuers proto signing	`certificate`	none

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

certificates

The `certificates` object holds a collection of certificate filters.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter proto	`certificates`	issuers keyUsage oids signing subjectDNs

Parent class

node class

Properties

Name	Description	Type	Access
credentialServerPolicy	Specifies whether checking the certificate status is required when a digital signature is signed. The certificate status can be checked against a certificate revocation list (CRL) or an Online Certificate Status Protocol (OCSP) response.	String	Read /Write
url	Specifies the URL for this object.	String	Read /Write
urlPolicy	(urlPolicy)Specifies the type of URL represented by the certificates object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

checkButton

The `checkButton` object that describes a check box or radio button control.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`checkButton`	border extras margin

Parent class

node class

Properties

Name	Description	Type	Access
allowNeutral	Specifies whether the check box or radio button can support an additional third state that represents a neutral value.	String	Read /Write
mark	Indicates the shape to use when filling a Check Box object.	String	Read /Write
shape	Specifies whether the check box or radio button displays with a square or round outline.	String	Read /Write
size	A measurement specifying the size of the check box or radio button outline representing either the height and width for a check box, or the diameter for a radio button.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

choiceList

The `choiceList` object that describes a list of options. The list of options is specified by one or more sibling objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`choiceList`	border extras margin

Parent class

node class

Properties

Name	Description	Type	Access
commitOn	Specifies when a user's selections are propagated to the data model.	String	Read /Write
open	Determines when the choice list is presented by interactive applications.	String	Read /Write
textEntry	Determines if a user can type a value into a drop-down list.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

color

The `color` object describes a unique color on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	corner edge fill linear pattern proto radial stipple	`color`	extras

Parent class

node class

Properties

Name	Description	Type	Access
cSpace	Specifies the color space.	String	Read /Write
use	Invokes a prototype.	String	Read /Write

Name	Description	Type	Access
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

comb

The `comb` object describes a comb field, where each letter of the field is divided by a black vertical line that spans the distance between the top and bottom edges of the field.

Only single- line comb fields can be created. If a `textEdit` object is a multiline field or a rich-text field, the presence of a comb child object will be considered an error and should produce a warning. The `maxChars` property on the `textEdit` object determines the number of combs to create.

The `comb` object is available for only dynamic or interactive PDF generation forms. Static PDF forms, and all other output formats, will ignore this object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto textEdit (`textEdit.comb` is reserved for future use)	`comb`	none

Parent class

node class

Properties

Name	Description	Type	Access
numberOfCells	Indicates the number of cells drawn for a comb field. This is not affected by the number of characters in the field's value.	Integer	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

command

The `command` object specifies a single command to execute against the data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	source	`command`	delete insert query update

Parent class

node class

Properties

Name	Description	Type	Access
timeout	Specifies the number of seconds to attempt a query.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

connect

The `connect` object describes the relationship between its containing object and a connection to a web service, schema, or data description. Connections are defined outside the form design in a separate packet with its own schema.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model sourceSet Model	exclGroup field proto source subform	`connect`	connectString password picture user

Parent class

node class

Properties

Name	Description	Type	Access
connection	Specifies the name of the associated connection control in the connection set.	String	Read /Write
delayedOpen	Specifies the number of seconds to delay opening the data source after a connection is made.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write
timeout	Specifies the number of seconds to attempt a query.	String	Read /Write
usage	Specifies the contexts in which to use the connection.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

connectionSet

The `connectionSet` object is the root object of the connectionSet model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	none	`connectionSet`	wsdlConnection xsdConnection

Parent class

model class

Properties

None

Methods

None

connectString

The `connectString` object specifies the connection string to use to connect to the database.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	connect	`connectString`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

contentArea

The `contentArea` object describes a region within a page area eligible for receiving content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	pageArea proto	`contentArea`	desc extras

Parent class

container class

Properties

Name	Description	Type	Access
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Name	Description	Type	Access
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

None

corner

The `corner` object describes the appearance of a vertex between two edges.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	border proto rectangle	`corner`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
inverted	Specifies whether the corner appears convex (it joins the edges tangentially) or is inverted and appears concave (it joins the edges at right angles).	String	Read /Write
join	Specifies the shape of the corner.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write
radius	Specifies the radius of the corner.	String	Read /Write
stroke	Specifies the appearance of a line.	String	Read /Write
thickness	Specifies the thickness or weight of the line.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

dataGroup

The `dataGroup` object is the parent of a list of XML data nodes within an XML data file. The nodes enclosed within the dataGroup object are either actual data values or other XML data objects, such as dataGroup objects. Subforms, as they appear in XML data files, are an example of data groups.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Data Model	dataGroup	`dataGroup`	dataGroup dataValue

Parent class

node class

Properties

None

Methods

None

dataModel

The `dataModel` object is the root object of the data model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Data Model	None	`dataModel`	dataWindow

Parent class

model class

Properties

None

Methods

None

dataValue

The `dataValue` object represents a container object that stores a value or values. For example, a dataValue object would be a field on a form.

Note: A `dataValue` object can have additional `dataValue` child objects that store additional data. Typically this is not the case.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Data Model	dataGroup	`dataValue`	dataValue

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Varies	Read /Write
contains	Determines whether a data value should be included in value of the parent object or as a property of the parent.	String	Read /Write
contentType	Specifies the type of content in the referenced document, expressed as a MIME type.	String	Read /Write
isNull	Indicates whether the current data value is the null value.	Boolean	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

dataWindow

The `dataWindow` object represents the range of records from the source data currently loaded into the data model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Data Model	dataModel	`dataWindow`	dataGroup dataValue

Parent class

object class

Properties

Name	Description	Type	Access
currentRecordNumber	Returns the current record number within the range of records contained by the current dataWindow object.	Integer	Read
isDefined	Indicates whether a valid data window is currently defined.	Boolean	Read
recordsAfter	Returns the number of records in the data window following the current record.	Integer	Read
recordsBefore	Returns the number of records that are in the data window prior to the current record.	Integer	Read

Methods

Name	Description	Returns
gotoRecord	Moves the current record of the data window to a particular record within the range of records in the data.	Empty
isRecordGroup	Indicates if a particular dataGroup object is also a single record.	Boolean
moveCurrentRecord	Repositions the current record to another location within the range of records.	Empty
record	Returns a record in a position relative to the current record.	Object

date

The `date` object describes a calendar date.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`date`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

dateTime

The `dateTime` object represents a date and time value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`dateTime`	None

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

dateTimeEdit

The `dateTimeEdit` object describes a control intended to aid in the selection of date and time.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`dateTimeEdit`	border comb extras margin

Parent class

node class

Properties

Name	Description	Type	Access
hScrollPolicy	Specifies whether a field can scroll horizontally.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

decimal

The `decimal` object represents a number with a fixed number of digits after the decimal.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`decimal`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Double	Read /Write
fracDigits	Specifies the maximum number of digits (inclusively) following the decimal point to capture and store.	String	Read /Write
leadDigits	Specifies the maximum number of digits (inclusively) preceding the decimal point to capture and store.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

defaultUi

The `defaultUi` object controls the depiction of objects whose appearance is delegated to the application.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`defaultUi`	extras

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

delete

The `delete` object specifies the delete current record operation from the data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	command	`delete`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

delta

The `delta` object permits the user restore the form state using script and holds information on the state to be restored.

For example, you can use the delta object to persist state information across edit sessions when working with certified forms.

A `delta` object can be queried at any time. Use the `delta` and deltas objects to restore any property on a form when subform.restoreState is set to `manual`.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	None	`delta`	None

Parent class

None

Properties

Name	Description	Type	Access
currentValue	Returns the correctly typed object for the property.	Dependent on property	Read
savedValue	Returns a typed object, but you cannot assign this value. If the property is not saved the value is the same as the currentValue.	Dependent on property	Read
target	Specifies the object upon which the event is acting.	String	Read

Methods

Name	Description	Returns
restore	Updates the property's current value with the saved value.	Null

deltas

The `deltas` object manages a list of delta objects.

The `deltas` object excludes any property that was set using a complex binding. It also excludes properties when the current value of the delta is the same as the saved or default value.

Use the delta and `deltas` objects to restore any property on a form when subform.restoreState is set to `manual`.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	None	`deltas`	None The `deltas` object is a list of delta objects.

Parent class

list class

Properties

None

Methods

None

desc

The `desc` object describes human-readable metadata.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	area contentArea draw exclGroup field pageArea proto subform subformSet	`desc`	boolean date dateTime decimal exData float image integer text time

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

Name	Description	Returns
metadata	Collects a comprehensive Extensible Metadata Platform (XMP) metadata packet for the document.	String

digestMethod

The `digestMethod` object lists an array of acceptable digest algorithms to use while signing. The valid values for PDF 1.7 are SHA1, SHA256, SHA384, SHA512 and RIPEMD160.

This object applies only if the digital credential that is signing contains RSA public/private keys. If it contains DSA public/private keys, then the digest algorithm is always SHA1 and this object is ignored. The default value, if not specified, is implementation-specific.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	digestMethods	`digestMethod`	None

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

digestMethods

The `digestMethods` object contains a list of acceptable `digestMethod` object values. If the credential contains RSA public/private keys, the valid values are SHA1, SHA256, SHA384, SHA512, RIPEMD160. If the credential contains DSA public/private keys, the only valid value is SHA1.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter	`digestMethods`	digestMethod

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read \Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

draw

The `draw` object contains non-interactive form design content. Within LiveCycle Designer ES, for example, the draw object describes the text, static image, circle, line, and rectangle objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	area pageArea proto subform	`draw`	assist border caption desc extras font margin para setProperty traversal ui value

Parent class

container class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
anchorType	Specifies the location of the container's anchor point when it is placed by using a positioned layout strategy.	String	Read /Write
colSpan	Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.	String	Read /Write
h	A measurement of the height for the layout.	String	Read /Write
hAlign	Specifies the horizontal text alignment.	String	Read /Write
locale	Specifies the language, currency, and time/date formatting to use for the content of the object.	String	Read /Write
maxH	Specifies the maximum height for layout purposes.	String	Read /Write
maxW	Specifies the maximum width for layout purposes.	String	Read /Write
minH	Specifies the minimum height for layout purposes.	String	Read /Write
minW	Specifies the minimum width for layout purposes.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write

Name	Description	Type	Access
rawValue	Specifies the unformatted value of the current object.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
rotate	Rotates the object around its anchor point by the specified angle.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
vAlign	Specifies the vertical text alignment.	String	Read /Write
w	A measurement specifying the width for the layout.	String	Read /Write
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

None

dSigData

The `dSigData` object describes a unit of XML digital signature data.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	TBD	`dSigData`	None

Parent class

node class

Properties

None

Methods

None

edge

The `edge` object describes an arc, line, or one side of a border or rectangle.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	arc border line proto rectangle	`edge`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
cap	Specifies the rendered termination of the stroke.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write
stroke	Specifies the appearance of a line.	String	Read /Write
thickness	Specifies the thickness or weight of the line.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

encoding

The `encoding` object corresponds to the PDFL subFilters element. The valid values for Adobe are adbe.x509.rsa_sha1, adbe.pkcs7.detached, and adbe.pkcs7.sha1, but other security handlers can define their own values.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	encodings	`encoding`	None

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

encodings

The `encodings` object contains a list of acceptable `encoding` object values.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter submit	`encodings`	encoding

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read/ Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

encrypt

The `encrypt` object encrypts the form data when it is submitted. It contains a certificate object that holds a public key for the encryption scheme. The encryption method used depends on the value of the format property.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	submit	`encrypt`	certificate

Parent class

node class

Properties

Name	Description	Type	Access
format	Determines the format in which to submit the data.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

event

The `event` object causes a script to execute or data to be submit whenever a particular event occurs.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	exclGroup field proto subform	`event`	extras execute script submit

Parent class

node class

Properties

Name	Description	Type	Access
activity	Specifies the name of the event.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

eventPseudoModel

The `eventPseudoModel` object is the root object of the event model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Event Model	None	`eventPseudoModel`	None

Parent class

object class

Properties

Name	Description	Type	Access
change	Specifies the value that a user types or pastes into a field immediately after they perform the action.	String	Read /Write
commitKey	Describes how the current value of a form field was set by the user.	Integer	Read /Write
fullText	Represents the full (untruncated) value that a user pastes into a form field.	String	Read /Write
keyDown	Determines whether a user is pressing an arrow key to make a selection. This property is available only for list boxes and drop-down lists.	Boolean	Read /Write
modifier	Determines whether the modifier key (for example, Ctrl on Microsoft Windows®) is held down when a particular event executes.	Boolean	Read /Write
newContentType	Specifies the content type of the newText property.	String	Read /Write
newText	Specifies the content of the field after it changes in response to user actions.	String	Read /Write
prevContentType	Specifies the content type of the value specified for the prevText property.	String	Read /Write

Name	Description	Type	Access
prevText	Specifies the content of the field before it changes in response to the actions of a user.	String	Read /Write
reenter	Specifies whether the `enter` event is occurring for the first time. The `enter` event occurs each time a user clicks in a field.	Boolean	Read /Write
selEnd	Specifies the index position of the last character of the text selection stored in the prevText property during a change event.	Integer	Read /Write
selStart	Specifies the index position of the first character of the text selection stored in the prevText property during a change event.	Integer	Read /Write
shift	Specifies whether the Shift key is held down during a particular event.	Boolean	Read /Write
soapFaultCode	Specifies any fault code that occurs when a user attempts to execute a web service connection.	String	Read /Write
soapFaultString	Specifies the descriptive message that corresponds to a particular web service connection fault code.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write

Methods

Name	Description	Returns
emit	Notifies the form event manager that an event has occurred.	Empty
reset	Resets all of the properties within the XML form event model.	Empty

exclGroup

The `exclGroup` object describes a mutual exclusion relationship between a set of containers.

An exclusion group is used to cause a set of radio buttons boxes to be mutually exclusive. When a user activates one member of the set, the other members are automatically deactivated. For example, if the set consists of radio buttons, clicking one button causes the other buttons to be deactivated.

Each member of the exclusion group is associated with an `on` value and an `off` value. When a member is activated, it assumes the `on` value. When it is deactivated, it assumes the `off` value. The `on` value for each member of a particular exclusion group must be unique.

Selecting one member of the exclusion group in the form causes each member's value to be set to its `on` or `off` value, as appropriate. Similarly, assigning the `on` value to a member of the exclusion group causes the other members to be deactivated.

Alternatively, a value may be assigned to the exclusion group itself. In this case, each member is activated only if the value matches the `on` value for that member.

Hierarchy of objects

Model	Parent objects	Current object	Child objects	
Form Model	area pageArea proto subform	`exclGroup`	assist bind border calculate caption connect desc	event extras field margin para setProperty traversal validate

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
access	Controls user access to the contents of a container.	String	Read /Write
accessKey	Specifies an accelerator key that is used by an interactive application to move the input focus to a particular field element.	String	Read /Write
anchorType	Specifies the location of the container's anchor point when it is placed by using a positioned layout strategy.	String	Read /Write
borderColor	Specifies the border color value for this field.	String	Read /Write
borderWidth	Specifies the border width for this field.	String	Read /Write
colSpan	Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.	String	Read /Write
fillColor	The background color value for this field.	String	Read /Write
h	A measurement of the height for the layout.	String	Read /Write
hAlign	Specifies the horizontal text alignment.	String	Read /Write
layout	Specifies the layout strategy to be used by this object.	String	Read /Write
mandatory	Specifies the nullTest value for the field.	String	Read /Write

Name	Description	Type	Access
mandatoryMessage	Specifies the mandatory message string for this field.	String	Read /Write
maxH	Specifies the maximum height for layout purposes.	String	Read /Write
maxW	Specifies the maximum width for layout purposes.	String	Read /Write
minH	Specifies the minimum height for layout purposes.	String	Read /Write
minW	Specifies the minimum width for layout purposes.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write
rawValue	Specifies the unformatted value of the current object.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
transient	Specifies whether the processing application must save the value of the exclusion group as part of a form submission or save operation.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
validationMessage	Specifies the validate message string for this field.	String	Read /Write
vAlign	Specifies the vertical text alignment.	String	Read /Write
w	A measurement specifying the width for the layout.	String	Read /Write
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

Name	Description	Returns
execCalculate	Executes the calculate script of the field.	Empty
execEvent	Executes the event script of the object.	Empty
execInitialize	Executes the initialize script of the field.	Empty
execValidate	Executes the validate script of the field.	Empty
selectedMember	Returns the selected member of an exclusion group.	Object

exData

The `exData` object describes a foreign data type.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`exData`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
contentType	Specifies the type of content in the referenced document, expressed as a MIME type.	String	Read /Write
href	Specifies a reference to an external file or resource.	String	Read /Write
maxLength	Specifies the maximum (inclusive) allowable length of the content or `-1` to indicate that no maximum length is imposed.	String	Read /Write
transferEncoding	Specifies the encoding of binary content in the referenced document.	String	Read /Write

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

execute

The `execute` object controls where an event is handled.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	event proto	`execute`	None

Parent class

node class

Properties

Name	Description	Type	Access
connection	Specifies the name of the associated connection control in the connection set.	String	Read /Write
executeType	Specifies whether to import new data into the existing form or merge new data with the original form design to create a new form.	String	Read /Write
runAt	Specifies what application can execute the script.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

exObject

The `exObject` object describes a single program or implementation-dependent foreign object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	area exObject proto subform ui	`exObject`	extras

Parent class

node class

Properties

Name	Description	Type	Access
archive	Specifies the URI location of an archive file that may contain program code related to the `exObject` object.	String	Read /Write
classId	Specifies a URI name or location for the program code represented by the object.	String	Read /Write
codeBase	Specifies a URI location that can be used to assist the resolution of a relative classId property.	String	Read /Write
codeType	Specifies an identifier corresponding to a MIME type that identifies the program code represented by the object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

extras

The `extras` object acts as an enclosure around one or more sets of custom properties. The content of this object may be used by custom applications.

Hierarchy of objects

Model	Parent objects			Current object	Child objects
Form Model sourceSet Model	area barcode border break button calculate caption checkButton choiceList color contentArea corner dateTimeEdit defaultUi draw edge	event exclGroup exObject extras field fill font format imageEdit keep linear margin numericEdit occur pageArea pageSet	passwordEdit pattern proto radial signature solid stipple subform subformSet template textEdit traversal traverse ui validate	`extras`	boolean date dateTime decimal exData extras float image integer text time

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

field

The `field` object describes a single interactive container capable of capturing and presenting data content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects	
Form Model	area exclGroup pageArea proto subform	`field`	assist bind bindItems border calculate caption connect desc event extras	font format items margin para setProperty traversal ui validate value

Parent class

container class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Varies	Read /Write
access	Controls user access to the contents of a container.	String	Read /Write
accessKey	Specifies an accelerator key that is used by an interactive application to move the input focus to a particular field element.	String	Read /Write
anchorType	Specifies the location of the container's anchor point when it is placed by using a positioned layout strategy.	String	Read /Write
borderColor	Specifies the border color value for this field.	String	Read /Write
borderWidth	Specifies the border width for this field.	String	Read /Write
colSpan	Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.	String	Read /Write
editValue	Specifies the edit value for the field.	String	Read /Write
fillColor	The background color value for this field.	String	Read /Write
fontColor	The foreground color value for the field.	String	Read /Write
formatMessage	Specifies the format validation message string for this field.	String	Read /Write

Name	Description	Type	Access
formattedValue	Specifies the formatted value for the field.	String	Read /Write
h	A measurement of the height for the layout.	String	Read /Write
hAlign	Specifies the horizontal text alignment.	String	Read /Write
locale	Specifies the language, currency, and time/date formatting to use for the content of the object.	String	Read /Write
mandatory	Specifies the nullTest value for the field.	String	Read /Write
mandatoryMessage	Specifies the mandatory message string for this field.	String	Read /Write
maxH	Specifies the maximum height for layout purposes.	String	Read /Write
maxW	Specifies the maximum width for layout purposes.	String	Read /Write
minH	Specifies the minimum height for layout purposes.	String	Read /Write
minW	Specifies the minimum width for layout purposes.	String	Read /Write
parentSubform	Specifies the parent subform (page) of this field.	Object	Read
presence	Specifies an object's visibility.	String	Read /Write
rawValue	Specifies the unformatted value of the current object.	Varies	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
rotate	Rotates the object around its anchor point by the specified angle.	String	Read /Write
selectedIndex	The index of the first selected item.	Integer	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
validationMessage	Specifies the validate message string for this field.	String	Read /Write

Name	Description	Type	Access
vAlign	Specifies the vertical text alignment.	String	Read /Write
w	A measurement specifying the width for the layout.	String	Read /Write
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

Name	Description	Returns
addItem	Adds new items to the current form field. For example, this method adds new items to a drop-down list.	Empty
boundItem	Gets the bound value of a specific display item of a drop-down list or list box.	String
clearItems	Removes all the items from the field. For example, it removes all the items contained within a drop-down list or a list box.	Empty
deleteItem	Deletes the specified item.	Boolean
execCalculate	Executes the calculate script of the field.	Empty
execEvent	Executes the event script of the object.	Empty
execInitialize	Executes the initialize script of the field.	Empty
execValidate	Executes the validate script of the field.	Empty
getDisplayItem	Retrieves the item display text for the specified item index.	String
getItemState	Returns the selection state of the specified item.	Boolean
getSaveItem	Retrieves the data value for the specified item index.	String
setItemState	Sets the selection state of the specified item.	Empty

fill

The `fill` object applies a color and optional rendered designs to the region enclosed by an object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	arc border font proto rectangle	`fill`	color extras linear pattern radial solid stipple

Parent class

node class

Properties

Name	Description	Type	Access
presence	Specifies an object's visibility.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

filter

The `filter` object describes the criteria for filtering signed certificates. The signed certificates are used to generate data signatures that follow the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto signature signData	`filter`	certificates digestMethods encodings handler mdp reasons timeStamp

The `mdp`, `reasons`, and `timestamp` child objects are valid only if the parent object is `signature`. If the parent object is `signData`, LiveCycle Designer ES ignores these child objects and does not generate them.

Parent class

node class

Properties

Name	Description	Type	Access
addRevocationInfo	Specifies whether the certificate status is checked when a digital signature is signed. The certificate status can be checked against a certificate revocation list (CRL) or an Online Certificate Status Protocol (OCSP) response.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

float

The `float` object describes a floating point value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`float`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Double	Read /Write
use	Invokes a prototype.	String	Read /Write

Name	Description	Type	Access
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	Double	Read /Write

Methods

None

font

The `font` object describes a font used on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	caption draw field proto	`font`	extras fill

Parent class

node class

Properties

Name	Description	Type	Access
baselineShift	Specifies a positive measurement that shifts a font up from the baseline or a negative measurement that shifts a font down from the baseline.	String	Read /Write
lineThrough	Specifies the activation of a single or double line extending through the text (also known as strikethrough).	String	Read /Write
lineThroughPeriod	Controls the appearance of the line extending through the text (also known as strikethrough).	String	Read /Write
overline	Specifies the activation and type of overlining.	String	Read /Write
overlinePeriod	Controls the appearance of overlining.	String	Read /Write
posture	Specifies the posture of the font.	String	Read /Write
size	A measurement specifying the size of the check box or radio button outline representing either the height and width for a check box, or the diameter for a radio button.	String	Read /Write

Name	Description	Type	Access
typeface	Specifies the name of the typeface.	String	Read /Write
underline	Specifies the activation and type of underlining.	String	Read /Write
underlinePeriod	Controls the appearance of underlining.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
weight	Controls the weight of the font typeface.	String	Read /Write

Methods

None

form

The `form` object is the root object for the form model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	None	`form`	desc extras

Parent class

model class

Properties

None

Methods

Name	Description	Returns
execCalculate	Executes the calculate script of the field.	Empty
execInitialize	Executes the initialize script of the field.	Empty
execValidate	Executes the validate script of the field.	Empty
formNodes	Returns a list of all form model objects that are bound to a specified data object.	Object

Name	Description	Returns
recalculate	Forces a specific set of scripts located on calculate events to execute. The specific events can be either pending calculate events or all calculate events.	Empty
remerge	Forces the remerging of the data model and template model to re-create the form model. After the remerge is complete, any layout model processing must be redone if necessary for the completed form.	Empty

format

The `format` object encloses input formatting and output formatting information, such as the picture clause.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	field proto	`format`	extras picture

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

handler

The `handler` object controls which signature handler is used for a data-signing operation, according to the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter proto	`handler`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
version	Indicates the version number of the current application.	String	Read

Methods

None

hostPseudoModel

The `hostPseudoModel` object is the root object of the host model. Use the host properties and methods at run time.

Examples of hosts include Acrobat and XFAPresentationAgent (server). Some hosts may not support all properties and methods. For example, XFAPresentationAgent does not support `xfa.host.`messageBox.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Host Model	None	`hostPseudoModel`	None

Parent class

object class

Properties

Name	Description	Type	Access
appType	Specifies the name of the client application in which a form currently exists.	String	Read /Write
calculationsEnabled	Specifies whether calculate scripts will execute.	Boolean	Read /Write
currentPage	Sets the currently active page of a document at run time.	Integer	Read /Write
language	Returns the language of the running host application.	String	Read
numPages	Returns the number of pages in the current document.	Integer	Read

Name	Description	Type	Access
platform	Returns the platform of the machine running the script.	String	Read
title	Sets and gets the title of the document. It is available only for client applications.	String	Read /Write
validationsEnabled	Specifies whether the validation scripts will execute.	Boolean	Read
variation	Indicates the packaging of the application that is running the script.	String	Read
version	Indicates the version number of the current application.	String	Read

Methods

Name	Description	Returns
beep	Causes the system to play a sound. It is available only for client applications.	Empty
documentCountInBatch	Determines the number of documents in the current batch.	Integer
documentInBatch	Determines the ordinal number of the current document within the batch.	Integer
exportData	Exports the data from the current form in either XDP or XML format to a file.	Empty
getFocus	Finds and returns the form object that currently has the input focus.	Object
gotoURL	Retrieves the specified URL. It is available only for client applications.	Empty
importData	Imports data to the current form from a specified file.	Empty
messageBox	Displays a dialog box on the screen. It is available only for client applications.	Integer
openList	Opens the drop-down list specified by the reference syntax expression.	Empty
pageDown	Moves to the next page of a form. Use the pageDown method at run time.	Empty
pageUp	Moves to the previous page of a form. Use the pageUp method at run time.	Empty
print	Prints a specific number of pages from a document. It is available only for client applications.	Empty
resetData	Resets the fields to their default values within a document.	Empty

Name	Description	Returns
response	Displays a dialog box containing a question and an entry field for the user to reply to the question. The return value is a string containing the user's response. If the user presses the cancel button on the dialog box, the response is null.	String
setFocus	Sets the keyboard focus to the form object specified by the reference syntax expression.	Empty

image

The `image` object describes a single image on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`image`	None

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read
aspect	Specifies how the image is to map to the nominal content region of the image's container.	String	Read /Write
contentType	Specifies the type of content in the referenced document, expressed as a MIME type.	String	Read /Write
href	Specifies a reference to an external file or resource.	String	Read /Write
transferEncoding	Specifies the encoding of binary content in the referenced document.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read

Methods

None

imageEdit

The `imageEdit` object encloses controls intended to aid in the manipulation of image content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`imageEdit`	border extras margin

Parent class

node class

Properties

Name	Description	Type	Access
data	Indicates whether the image provided to the widget should be represented as a reference or should be embedded.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

insert

The `insert` object specifies the insert current record operation from the data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	command	`insert`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

instanceManager

The `instanceManager` object manages the instance creation, removal, and movement of form model objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	subform	`instanceManager`	occur

Parent class

node class

Properties

Name	Description	Type	Access
count	Specifies the current number of subform instances on a form.	String	Read /Write
max	Specifies the maximum number of occurrences for the enclosing container, or `-1` to set no upper boundary for occurrences.	String	Read /Write
min	Specifies the minimum number of occurrences for the enclosing container.	String	Read /Write

Methods

Name	Description	Returns
addInstance	Adds a new instance of a subform or subform set to the form model.	Object
insertInstance	Inserts a new instance of a subform or subform set into a form.	Object
moveInstance	Moves a `subform` object within a set of subform instances.	Empty
removeInstance	Removes a specified subform or subform set from the form model.	Empty
setInstances	Adds or removes specified subforms or subform sets from the form model.	Empty

integer

The `integer` object describes an integer value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model sourceSet Model	desc exObject extras items proto value variables	`integer`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Integer	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	Integer	Read /Write

Methods

None

issuers

The `issuers` object describes a collection of issuer certificates that are acceptable for data signing according to the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	certificates proto	`issuers`	none

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

items

The `items` object supplies a column of choices for a list box or a check box.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	field proto	`items`	boolean date dateTime decimal exData float image integer text time

Parent class

node class

Properties

Name	Description	Type	Access
presence	Specifies an object's visibility.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write
save	Determines whether the values in a particular column represent both display and bound values, or if the data in the column represents bound values only.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

keep

The `keep` object describes the constraints involved in keeping subforms together within a page or content area.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform	`keep`	extras

Parent class

node class

Properties

Name	Description	Type	Access
intact	Specifies the constraints on keeping a subform intact within a content area or page.	String	Read /Write
next	Specifies the constraints on keeping a subform together with the next subform within a content area or page.	String	Read /Write
previous	Specifies the constraints on keeping a subform together with the previous subform within a content area or page.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

keyUsage

The keyUsage object describes the key usage settings that are required for the signing certificate. It is constructed with a character that is used to represent each key usage type. The first through ninth

characters, from left to right, represent the required value for these properties: `digitalSignature`, `nonRepudiation`, `keyEncipherment`, `dataEncipherment`, `keyAgreement`, `keyCertSign`, `crlSign`, `encipherOnly`, `decipherOnly`. Any additional characters are ignored.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	certificates	`keyUsage`	None

Parent class

node class

Properties

Name	Description	Type	Access
crlSign	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
dataEncipherment	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
decipherOnly	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
digitalSignature	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
encipherOnly	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
keyAgreement	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
keyCertSign	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
keyEncipherment	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
nonRepudiation	Specifies an acceptable key usage extension that must be present in the signing certificate.	String	Read /Write
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

layoutPseudoModel

The `layoutPseudoModel` object is used to query parameters that are only known after the form is laid out such as which page a form design object lies on, the total number of pages, how many pages an object spans, or the orientation of the form design object.

Hierarchy of objects

Model	Parent objects	This object	Child objects
Layout Model	None	`layoutPseudoModel`	None

Parent class

object class

Properties

Name	Description	Type	Access
ready	Specifies whether the form layout process is complete and scripting tasks can begin.	Boolean	Read

Methods

Name	Description	Returns
absPage	Determines the page of the form that a given form design object first appears on.	Integer
absPageCount	Determines the page count of the current form.	Integer
absPageCountInBatch	Determines the page count of the current batch.	Integer
absPageInBatch	Determines which page within the batch contains the form object.	Integer
absPageSpan	Determines the number of pages that a specified form object spans.	Integer
h	Determines the height of a given form design object.	Double
page	Determines the page number that contains a given form design object. If the object spans multiple pages, this method returns the first page the object occurs on.	Integer
pageContent	Retrieves types of form design objects from a specified page of a form.	Object
pageCount	Determines the number of pages of the current form.	Integer
pageSpan	Determines the number of logical pages a given form design object spans.	Integer
relayout	Reapplies the layout options to the current form.	Empty
relayoutPageArea	Replaces the layout of the pageArea object content with a new layout.	Empty
sheet	Determines the sheet number that contains the form object.	Integer

Name	Description	Returns
sheetCount	Determines the number of sheets in the current form.	Integer
sheetCountInBatch	Determines the sheet count of the current batch.	Integer
sheetInBatch	Determines which sheet within the batch contains the form object.	Integer
w	Determines the width of a given form design object.	Double
x	Determines the x coordinate of a given form design object.	Double
y	Determines the y coordinate of a given form design object.	Double

line

The `line` object describes a single rendered line on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto value	`line`	edge

Parent class

node class

Properties

Name	Description	Type	Access
hand	Describes the justification of a line or edge.	String	Read /Write
slope	Specifies the orientation of the line.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

linear

The `linear` object describes a linear gradient fill on a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	fill proto	`linear`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

manifest

The `manifest` object contains a list of references to all the nodes that are included in a document signature.

When the `manifest` objects is a child of the `signature` object, the document signature can protect a collection of nodes instead of the entire form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto signature signData	`manifest`	extras

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	Boolean	Read /Write
action	Identifies the form nodes that are protected by a document signature.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

Name	Description	Returns
evaluate	Gets the list of objects referred to in the manifest.	Object

map

The `map` object specifies data mappings from the column names of a data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	query	`map`	None

Parent class

node class

Properties

Name	Description	Type	Access
bind	Specifies the name of a unique binding ID where columns from the data source specified by the from property are bound.	String	Read /Write
from	Specifies the original column name in the data source.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

margin

The `margin` object specifies margin values for a form design object.

Hierarchy of objects

Model	Parent objects		Current object	Child objects
Form Model	border caption checkButton choiceList dateTimeEdit draw exclGroup	field imageEdit numericEdit passwordEdit proto signature subform textEdit	`margin`	extras

Parent class

node class

Properties

Name	Description	Type	Access
bottomInset	Specifies the size of the bottom inset.	String	Read /Write
leftInset	Specifies a the size of the left inset.	String	Read /Write
rightInset	Specifies the size of the right inset.	String	Read /Write
topInset	A measurement specifying the size of the top inset.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

mdp

The `mdp` object provides support for Modify Detection Prevention Plus (MDP+) digital signatures. Acrobat 8.0 and later supports MDP+ digital signature for XFA-based forms.

In LiveCycle Designer ES, MDP+ signatures are implemented with a Document Signature Field. Document signature fields enable you to specify a collection of form objects that are protected by the document signature. Using document signatures prevents the form or a portion of the form from being modified.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter	`mpd`	None

Parent class

node class

Properties

Name	Description	Type	Access
permissions	Specifies the access permissions granted for a form that includes an author signature.	String	Read /Write
signatureType	Specifies how a form with a document signature is saved as certified PDF document.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

medium

The `medium` object describes a physical medium upon which to render.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto pageArea	medium	None

Parent class

node class

Properties

Name	Description	Type	Access
imagingBBox	Specifies a region within the medium that is available for rendering with four comma separated measurements representing the measurements for x, y, width, and height.	String	Read /Write
long	Specifies the length of the long edge of the medium. The length specified by the `long` property must be greater than the length specified by the short property.	String	Read /Write

Name	Description	Type	Access
orientation	Specifies the orientation of the medium.	String	Read /Write
short	Specifies the length of the short edge of the `medium` object.	String	Read /Write
stock	Specifies the name of a standard paper size.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

message

The `message` object holds one or more sub-objects containing validation failure messages.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	calculate proto validate	`message`	text

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

numericEdit

The `numericEdit` object describes a control intended to aid in the manipulation of numeric content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`numericEdit`	border comb extras margin

Parent class

node class

Properties

Name	Description	Type	Access
hScrollPolicy	Specifies whether a field can scroll horizontally.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

occur

The `occur` object describes the constraints over the number of allowable instances for its enclosing container.

Modify the `occur` object on the `template:ready` event. However, the `template:ready` event is not accessible in the user interface. You cannot modify the `occur` object at the `form:ready` event, because this event occurs too late in the form processing.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	pageArea pageSet proto subform subformSet	`occur`	extras script (`occur.script` is reserved for future use)

Parent class

node class

Properties

Name	Description	Type	Access
initial	Specifies the initial number of occurrences for the enclosing container.	String	Read /Write
max	Specifies the maximum number of occurrences for the enclosing container, or `-1` to set no upper boundary for occurrences.	String	Read /Write
min	Specifies the minimum number of occurrences for the enclosing container.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

oid

The `oid` object describes an Object Identifier (OID) of the certificate policies that must be present in the signing certificate.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	oids proto	`oid`	none

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

oids

The `oids` object describes a collection of Object Identifiers (OIDs) that apply to signing data according to the W3C XML-Signature standards.

This object is only applicable if it has a sibling `issuers` object that is not empty.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	certificates proto	`oid s`	none

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

operation

The `operation` object represents a specific operation provided by a particular WSDL address. Each operation is a single data connection.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	wsdlConnection	`operation`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
input	Specifies an input message associated with a particular WSDL connection operation.	String	Read /Write
output	Specifies the output message associated with a particular WSDL connection operation.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

overflow

The `overflow` object stores properties that are used when a parent subform overflows the current contentArea.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform subformSet	`overflow`	none

Parent class

node class

Properties

Name	Description	Type	Access
leader	Specifies the `subform` or `subformSet` object to place at the top of a content or page area.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write
trailer	Specifies the `subform` or `subformSet` object to place at the bottom of a content or page area.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

packet

The `packet` object stores unrecognized objects; that is, those that do not conform to any of the other XML Form Object Models. This object provides a way to copy, move, or retrieve the information in these unrecognized objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
XFA Model	dataGroup	`packet`	None

Parent class

node class

Properties

Name	Description	Type	Access
content	Specifies the content of the object.	String	Read /Write

Methods

Name	Description	Returns
getAttribute	Gets a specified property value.	String
removeAttribute	Removes the specified property.	Empty
setAttribute	Sets the value of a specified property.	Empty

pageArea

The `pageArea` object describes a rendering surface.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	pageSet proto	`pageArea`	area contentArea desc draw exclGroup extras field medium occur subform

Parent class

container class

Properties

Name	Description	Type	Access
blankOrNotBlank	Specifies whether the page area is intended to be blank and therefore may result in special treatment by the output device.	String	Read /Write
initialNumber	Supplies the initial page number to the first page in a group of consecutive pages that use the same pageSet.	String	Read /Write
numbered	Specifies whether the page area is considered a numbered page area.	String	Read /Write
oddOrEven	Specifies whether a page is odd or even for pagination within a set of pages.	String	Read /Write
pagePosition	Specifies a page's position within a set of pages.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

pageSet

The `pageSet` object describes a set of related page area objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model		`pageSet`	extras occur

Parent class

container class

Properties

Name	Description	Type	Access
relation	Specifies the relationship among the members of the set.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

para

The `para` object specifies the default paragraph and alignment properties to be applied to the content of an enclosing container.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	caption draw exclGroup field proto subform	`para`	None

Parent class

node class

Properties

Name	Description	Type	Access
hAlign	Specifies the horizontal text alignment.	String	Read /Write
lineHeight	Specifies the line height to apply to the paragraph content.	String	Read /Write
marginLeft	Specifies the size of the left indentation of the paragraph.	String	Read /Write
marginRight	Specifies the size of the right indentation of the paragraph.	String	Read /Write

Name	Description	Type	Access
preserve	Specifies widow/orphan-style constraints on the overflow behavior of the content within the enclosing container.	String	Read /Write
radixOffset	Specifies an offset value for the anchor of the paragraph.	String	Read /Write
spaceAbove	Specifies the amount of vertical spacing and the maximum font leading for the first line of the paragraph.	String	Read /Write
spaceBelow	Specifies the amount of vertical spacing and the maximum font leading for the first line of the paragraph.	String	Read /Write
tabDefault	Specifies the distance between default tab stops.	String	Read /Write
tabStops	Specifies a space-separated list of tab stop locations.	String	Read /Write
textIndent	Specifies the horizontal positioning of the first line relative to the remaining lines in a paragraph.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
vAlign	Specifies the vertical text alignment.	String	Read /Write

Methods

None

password

The `password` object specifies the password for the data source (if required for connection).

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	connect	`password`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

passwordEdit

The `passwordEdit` object describes a control intended to aid in the manipulation of password content. Typically, the user interface will obscure any visual representation of the content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`passwordEdit`	border extras margin

Parent class

node class

Properties

Name	Description	Type	Access
hScrollPolicy	Specifies whether a field can scroll horizontally.	String	Read /Write
passwordChar	Specifies the character the form displays for each password character a user enters.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

pattern

The `pattern` object describes a fill pattern for a form design object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	fill proto	`pattern`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

picture

The `picture` object describes input mask and output formatting information.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	bind connect format proto ui validate	`picture`	None

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write

Name	Description	Type	Access
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

proto

The `proto` object describes a set of reusable object definitions.

Hierarchy of objects

Model	Parent objects	Current object	Child objects			
Form Model	subform	`proto`	arc area assist barcode bind bindItems boolean border break breakAfter breakBefore button calculate caption checkButton choiceList color connect contentArea corner	date dateTime dateTimeEdit decimal defaultUi desc draw edge event exclGroup exData execute exObject extras field fill float font format	image imageEdit integer items keep line linear margin medium message numericEdit occur pageArea pageSet para passwordEdit pattern picture radial rectangle	script setProperty signature solid speak stipple subform subformSet submit template text textEdit time toolTip traversal traverse ui validate value variables

Parent class

node class

Properties

None

Methods

None

query

The `query` object represents a specific query of a particular data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	command	`query`	recordSet select

Parent class

node class

Properties

Name	Description	Type	Access
commandType	Specifies the type of command used by a data query.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

radial

The `radial` object describes a radial gradient fill.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	fill proto	`radial`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

reason

The `reason` object contains an acceptable reason for signing data per the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto reasons	`reason`	none

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

reasons

The `reasons` object contains acceptable reasons for signing data per the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter proto	`reasons`	none

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

recordSet

The `recordSet` object contains a number of records based on a specific query of the data source. These records can be viewed, reorganized, added, and removed.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	query	`recordSet`	None

Parent class

node class

Properties

Name	Description	Type	Access
bofAction	Specifies the action to perform if the current record is the first record in the record set.	String	Read /Write
cursorLocation	Indicates the location of the cursor library to use with the record set.	String	Read /Write
cursorType	Specifies the type of cursor to use when opening the record set.	String	Read /Write
eofAction	Specifies the action to perform if the current record is the last record in the record set.	String	Read /Write
lockType	Specifies the type of locking functionality to use with the data source.	String	Read /Write
max	Specifies the maximum number of occurrences for the enclosing container, or `-1` to set no upper boundary for occurrences.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

rectangle

The `rectangle` object describes a single rendered rectangle.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto value	`rectangle`	corner edge fill

Parent class

node class

Properties

Name	Description	Type	Access
hand	Describes the justification of a line or edge.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

ref

The `ref` object contains a reference syntax expression that identifies a node to be included in an XML digital signature.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	bindItems manifest proto setProperty	`ref`	none

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

rootElement

The `rootElement` object specifies the XML element within an XML Schema data connection to use as the root of any data file used within the form.

Hierarchy of objects

Model	Parent Objects	Current Object	Child Objects
connectionSet Model	xsdConnection	`rootElement`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

script

The `script` object contains a script written in FormCalc or JavaScript.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	breakAfter breakBefore calculate event occur (`occur.script` is reserved for future use) proto traverse validate variables	`script`	None

Parent class

node class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
binding	Identifies the type of application to which the script is directed.	String	Read /Write
contentType	Specifies the type of content in the referenced document, expressed as a MIME type.	String	Read /Write
runAt	Specifies what application can execute the script.	String	Read /Write
stateless	Determines whether a script's variables persist from one invocation to the next.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

select

The `select` object contains the select statement query information to use with the current data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	query	`select`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

setProperty

The `setProperty` object modifies a property of its parent object. A parent object can contain any number of `setProperty` objects.

The target property is a reference syntax expression that describes a single property of the parent object. This property identifies the node for which the value is to be set to the value identified by the ref object and connection property. For example, the target specified to set the `toolTip` for a field would be `access.toolTip`.

Within the parent container, there are no restrictions on which properties the `setProperty` object can target. However, the `setProperty` object cannot target the properties of nested containers.

The application of the `setProperty` object is a template operation. The reference is resolved and the data value is applied to the `target` property when generating the form as a result of a merge. There is no permanent link between the data node and the property. Subsequent changes to the data are not propagated to the target property unless another merge occurs.

Note: Using the `setProperty` object to target bind related properties, such as the `bind` object or `#name`, is unlikely to be useful, because the setProperty application occurs after the merge process has occurred.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	draw exclGroup field proto subform	`setProperty`	ref

Parent class

node class

Properties

Name	Description	Type	Access
connection	Specifies the name of the associated connection control in the connection set.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write

Methods

None

signature

The `signature` object determines which other objects are signed by a signature.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`signature`	border extras filter manifest margin

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

signatureProperties (deprecated)

The `signatureProperties` object holds the properties of an XML-signature data signature. Objects inserted within this object are inserted into the XML-Signature as XMP data.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model		`signturePropertie s`	none

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

signaturePseudoModel

The `signaturePseudoModel` object is the root object of the signature model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Signature Model	None	`signaturePseudoModel`	None

Parent class

object class

Methods

Name	Description	Returns
clear	Removes a given signature.	Boolean
enumerate	Enumerates all the XML signatures found in the document.	Object

Name	Description	Returns
sign	Signs a given node list and places the signature in the target location.	Boolean
verify	Checks the validity of a signature.	Integer

signData

The `signData` object controls the creation of a data signature as specified by the W3C XML-Signature standard.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	event proto submit	`signData`	filter manifest ref

Parent class

node class

Properties

Name	Description	Type	Access
operation	Indicates which signature operation to perform or when a link was used.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

signing

The `signing` object describes a collection of signing certificates that are acceptable for data signing according to the W3C XML-Signature standards.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	certificates proto	`signing`	none

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

soapAction

The `soapAction` object contains a fully qualified SOAP action.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	wsdlConnection	`soapAction`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

soapAddress

The `soapAddress` object stores the fully qualified location of the SOAP end point. This location must be specified in RFC 2396 standard format.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	wsdlConnection	`soapAddress`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

solid

The `solid` object describes a solid fill style of a form design object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	fill proto	`solid`	extras

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

source

The `source` object represents an external data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	sourceSet	`source`	connect

Parent class

node class

Properties

Name	Description	Type	Access
db	Specifies the name of a database available from the provider.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

Name	Description	Returns
addNew	Appends a new record to the record set.	Empty
cancel	Cancels any changes made to the current or new row of a record set object, or the field collection of a record object, prior to calling the update method.	Empty
cancelBatch	Cancels a pending batch update.	Empty
close	Closes a connection to a data source.	Empty
delete	Deletes the current record from the record set.	Empty
first	Moves to the first record in the record set, and populates the data model with the record data.	Empty
hasDataChanged	Determines whether the current record data has changed.	Boolean
isBOF	Determines if the current location is at the beginning of the record set. The bofAction property must be set to `stayBOF`.	Boolean
isEOF	Determines if the current location is at the end of the record set. The eofAction property must be set to `stayEOF`.	Boolean
last	Moves to the last record in the record set, and populates the data model with the record data.	Empty
next	Moves to the next record in the record set, and populates the data model with the record data.	Empty

Name	Description	Returns
open	Connects to the data source and populates the data model with the results of the current record.	Empty
previous	Moves to the previous record in the record set, and populates the data model with the record data.	Empty
requery	Updates the current data binding by re-executing the query on which the object data is based. Calling this method is equivalent to calling the close and open methods in succession.	Empty
resync	Refreshes the current record set or data connection.	Empty
update	Updates the current record in the record set.	Empty
updateBatch	Writes all pending batch updates to the data source.	Empty

sourceSet

The `sourceSet` object is the root object of the sourceSet model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	None	`sourceSet`	source

Parent class

model class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

speak

The `speak` property plays an audible prompt describing the contents of a container object, such as a field or subform. This object is ignored by non-interactive form applications.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	assist proto	`speak`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
disable	Inhibits the audible prompt.	String	Read /Write
priority	Alters the search path for text to speak. Whichever object is named in this property moves to the front of the search path. The other objects retain their relative order.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

stipple

The `stipple` object describes a stippling effect for a form object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	fill proto	`stipple`	color extras

Parent class

node class

Properties

Name	Description	Type	Access
rate	Specifies the percentage of stipple color that is stippled over a solid background color.	String	Read /Write

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

subform

The `subform` object describes a single subform capable of enclosing other containers.

Hierarchy of objects

Model	Parent objects	Current object	Child objects			
Form Model	area	`subform`	assist bind bookend border break	breakAfter breakBefore calculate desc extras	instanceManager keep margin occur overflow	para setProperty traversal validate variables

Parent class

container class

Properties

Name	Description	Type	Access
allowMacro	Specifies whether to permit the processing application to optimize output by generating a printer macro for all of the subform's draw content.	String	Read /Write
anchorType	Specifies the location of the container's anchor point when it is placed by using a positioned layout strategy.	String	Read /Write
colSpan	Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.	String	Read /Write
columnWidths	Specifies the widths for columns of a table.	String	Read /Write
h	A measurement of the height for the layout.	String	Read /Write
hAlign	Specifies the horizontal text alignment.	String	Read /Write

Name	Description	Type	Access
instanceIndex	Calculates the index of a subform or subform set based on where it is located relative to other instances of the same form object.	Integer	Read /Write
layout	Specifies the layout strategy to be used by this object.	String	Read /Write
locale	Specifies the language, currency, and time/date formatting to use for the content of the object.	String	Read /Write
maxH	Specifies the maximum height for layout purposes.	String	Read /Write
maxW	Specifies the maximum width for layout purposes.	String	Read /Write
minH	Specifies the minimum height for layout purposes.	String	Read /Write
minW	Specifies the minimum width for layout purposes.	String	Read /Write
presence	Specifies an object's visibility.	String	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
restoreState	Restores the form nodes of a form to their original state, including resetting the visual properties of fields such as changes to border colors.	String	Read /Write
scope	Controls participation of the subform in data binding and reference syntax expressions. It is valid only on the root subform.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
validationMessage	Specifies the validate message string for this field.	String	Read /Write
vAlign	Specifies the vertical text alignment.	String	Read /Write
w	A measurement specifying the width for the layout.	String	Read /Write

Name	Description	Type	Access
x	Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write
y	Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.	String	Read /Write

Methods

Name	Description	Returns
execCalculate	Executes the calculate script of the field.	Empty
execEvent	Executes the event script of the object.	Empty
execInitialize	Executes the initialize script of the field.	Empty
execValidate	Executes the validate script of the field.	Empty

subformSet

The `subformSet` object describes a set of related subform objects.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	area subform	`subformSet`	bookend break breakAfter breakBefore desc extras instanceManager occur overflow subform

Parent class

container class

Properties

Name	Description	Type	Access
instanceIndex	Calculates the index of a subform or subform set based on where it is located relative to other instances of the same form object.	Integer	Read /Write
relation	Specifies the relationship among the members of the set.	String	Read /Write

Name	Description	Type	Access
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

subjectDN

The `subnectDN` object describes the attributes for a subject Distinguished Name (DN) that must be present within the signing certificate for it to be acceptable for signing. It is an array of dictionaries, where each dictionary contains key value pairs that specify the subject DN. The certificate must contain all the attributes specified in the dictionary, but it can contain additional attributes. The key can be any legal attribute identifier.

For more information about the various Subject Distinguished attributes and their types, refer to RFC 3280.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	subjectDNs	`subjectDN`	None

Parent class

node class

Properties

Name	Description	Type	Access
delimiter	Separates the attributes in the Subject DN string.	String	Read /Write

Methods

None

subjectDNs

The `subjectDNs` object describes the collection of key value pairs that is used to specify the subject DN.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	certificates	`subjectDNs`	subjectDN

Parent class

node class

Properties

Name	Description	Type	Access
type	Specifies the pattern used by an object.	String	Read/ Write

Methods

None

submit

The `submit` object describes how to submit data to a host.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	event proto	`submit`	encrypt signData

Parent class

node class

Properties

Name	Description	Type	Access
embedPDF	Determines whether PDF file will be included as part of the data.	String	Read /Write
format	Determines the format in which to submit the data.	String	Read /Write
target	Specifies the object upon which the event is acting.	String	Read /Write
textEncoding	Specifies the encoding of text content in the document.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
xdpContent	Controls what subset of the data is submitted. This property is used only when the format property is `xdp`.	String	Read /Write

Methods

None

template

The `template` object describes a template. One such object exists for each template and all other objects that are descendants of the template object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto	`template`	extras

Parent class

model class

Properties

None.

Methods

Name	Description	Returns
createNode	Creates a new node based on a valid class name.	Object
execCalculate	Executes the calculate script of the field.	Empty
execInitialize	Executes the initialize script of the field.	Empty
execValidate	Executes the validate script of the field.	Empty
formNodes	Returns a list of all form model objects that are bound to a specified data object.	Object
recalculate	Forces a specific set of scripts located on calculate events to execute. The specific events can be either pending calculate events or all calculate events.	Empty
remerge	Forces the remerging of the data model and template model to re-create the form model. After the remerge is complete, any layout model processing must be redone if necessary for the completed form.	Empty

text

The `text` object describes a single unit of data content representing a plain text value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model sourceSet Model	desc exObject extras items message proto value variables	`text`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
maxChars	Specifies the maximum number of characters that this text value can enclose.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

textEdit

The `textEdit` object encloses controls intended to aid in the manipulation of text content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto ui	`textEdit`	border comb (`textEdit.comb` is reserved for future use) extras margin

Parent class

node class

Properties

Name	Description	Type	Access
allowRichText	Specifies whether the text can include styling (also known as rich text).	String	Read /Write
hScrollPolicy	Specifies whether a field can scroll horizontally.	String	Read /Write
multiLine	Specifies whether the text may span multiple lines.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write
vScollPolicy	Specifies whether a field can scroll vertically.	String	Read /Write

Methods

None

time

The `time` object describes a single unit of data representing a time value.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	desc exObject extras items proto value variables	`time`	None

Parent class

content class

Properties

Name	Description	Type	Access
{default}	Represents the actual value stored by an object.	String	Read /Write
use	Invokes a prototype.	String	Read /Write

Name	Description	Type	Access
usehref	Invokes an external prototype.	String	Read /Write
value	Specifies the value of the current object.	String	Read /Write

Methods

None

timeStamp

The `timeStamp` object appends a time stamp to a document signature. A time stamp specifies the date and time when a document was signed and removes any doubt about when the document was signed.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	filter	`timestamp`	None

Parent class

node class

Properties

Name	Description	Type	Access
server	Specifies the URL for a time stamp server.	String	Read /Write
type	Specifies the pattern used by an object.	String	Read/ Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

toolTip

The `toolTip` object supplies text for a tool tip on a form. This object is ignored by non-interactive form applications.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	assist proto	`tooltip`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

traversal

The `traversal` object links its container to other objects in sequence.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	draw exclGroup field proto subform	`traversal`	extras

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

traverse

The `traverse` object declares a single link from its container to another object in a unidirectional chain of links.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto traversal	`traverse`	extras script

Parent class

node class

Properties

Name	Description	Type	Access
operation	Indicates which signature operation to perform or when a link was used.	String	Read /Write
ref	Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

ui

The `ui` object encloses the user interface description of a form object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	draw field proto	`ui`	extras imageEdit picture

Parent class

node class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

update

The `update` object specifies the update current record operation from the data source.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	command source	`update`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

uri

The `uri` object stores a fully qualified URI for a specific xmlConnection or xsdConnection object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	xmlConnection xsdConnection	`uri`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

user

The `user` object specifies the user id for the data source (if required for connection).

Hierarchy of objects

Model	Parent objects	Current object	Child objects
sourceSet Model	connect	`user`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

validate

The `validate` object controls validation of user-supplied data on a form.

The `validate` object can be activated multiple times during the life of a form.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	exclGroup field proto subform	`validate`	extras message picture script

Parent class

node class

Properties

Name	Description	Type	Access
formatTest	Controls validation against the display picture clause.	String	Read /Write
nullTest	Controls whether a field is mandatory on a form or if it can be left empty.	String	Read /Write
scriptTest	Controls validation by the enclosed script.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

value

The `value` object encloses a single unit of data content.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	caption draw field proto	`value`	None

Parent class

node class

Properties

Name	Description	Type	Access
override	When used with the calculate object, the `override` property indicates whether the field allows overrides to occur and disables or enables calculations. When used with the value object, the `override` property indicates whether a calculation override has occurred.	Boolean	Read /Write
relevant	Controls whether a form object is included when the form is printed.	String	Read /Write
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

variables

The `variables` object is used to hold document variables.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
Form Model	proto subform	`variables`	None

Parent class

container class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

wsdlAddress

The `wsdlAddress` object contains the original URL of the WSDL referenced in the wsdlConnection object.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	wsdlConnection	`wsdlAddress`	None

Parent class

textNode class

Properties

Name	Description	Type	Access
use	Invokes a prototype.	String	Read /Write
usehref	Invokes an external prototype.	String	Read /Write

Methods

None

wsdlConnection

The `wsdlConnection` object identifies a unique WSDL web services connection.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	connectionSet	`wsdlConnection`	operation soapAction soapAddress wsdlAddress

Parent class

node class

Properties

Name	Description	Type	Access
dataDescription	Specifies the name of a data connection description to use with a particular type of web services connection.	String	Read /Write

Methods

Name	Description	Returns
execute	Executes a connection.	Boolean

xfa

The `xfa` object is the root node for the xfa model.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
XFA Model	None	`xfa`	packet

Parent class

model class

Properties

Name	Description	Type	Access
this	Retrieves the current node, which is the starting node when using the resolveNode and resolveNodes methods.	Object	Read
timeStamp	Specifies the date/time stamp for this node.	String	Read /Write
uuid	Specifies the Universally Unique Identifier (UUID) for this object.	String	Read /Write

None

Methods

None

xmlConnection

The `xmlConnection` object is used to store a sample XML data connection.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	connectionSet	`xmlConnection`	uri

Parent class

node class

Properties

Name	Description	Type	Access
dataDescription	Specifies the name of a data connection description to use with a particular type of web services connection.	String	Read /Write

Methods

None

xsdConnection

The `xsdConnection` object stores an XML Schema data connection entry.

Hierarchy of objects

Model	Parent objects	Current object	Child objects
connectionSet Model	connectionSet	`xsdConnection`	rootElement uri

Parent class

node class

Properties

Name	Description	Type	Access
dataDescription	Specifies the name of a data connection description to use with a particular type of web services connection.	String	Read /Write

Methods

None

4 Scripting Properties

This section provides an alphabetical list of all properties supported in this scripting environment.

Note: All properties have read/write access unless otherwise specified.

#text

A string of text.

Syntax

Reference_Syntax.#text.value = "*text*"

Values

Type	Values
String	• Any valid string.

Version

XFA 2.1

Examples

JavaScript

```
TextField1.caption.value.#text = "This is a caption.";
```

FormCalc

```
TextField1.caption.value.#text = "This is a caption."
```

{default}

Represents the actual value stored by an object.

The type and possible values differ depending on the object.

Syntax

Reference_Syntax = "*value*"

Values

Type	Values
Varies	Values differ from object to object.

Applies to

Model	Object
Data Model	dataValue
Form Model	boolean date dateTime decimal draw exclGroup exDatafield float image integer picture text time
sourceSet Model	boolean integer text

Also applies to objects derived from the textNode class.

Version

XFA 2.1

access

Controls user access to the contents of a container.

Syntax

```
Reference_Syntax.access = "open | protected | readOnly"
```

Values

Type	Values
String	• `open` (default) Allows updating of a container's contents and navigation into and out of the container without restriction. In interactive forms, you can modify the container's content and tab or otherwise navigate into it. The container produces events. • `protected` The processing application prevents the user from making any direct changes to the container's content. Indirect changes such as calculations can occur. The container does not participate in the tabbing sequence, though an application may allow the selection of text for clipboard copying. Protected containers do not generate any events. • `readOnly` The application does not allow a user to make direct changes to the container's content, but indirect changes such as calculations can occur. The container participates in the tabbing sequence and allows users to view the content. The user can select the container's content for clipboard copying. The container generates a subset of events (those not associated with the user making direct changes to the content).

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.access = "readOnly";
```

FormCalc

```
TextField1.access = "readOnly"
```

See also

- "Referencing objects" on page 420
- "Changing the background color" on page 428
- "Disabling all form fields" on page 434

accessKey

Specifies an accelerator key that is used by an interactive application to move the input focus to a particular field element.

Syntax

Reference_Syntax.accessKey = "*character*"

Values

Type	Values
String	The value of this attribute is a single character. When the user synchronously presses the platform-specific modifier key and the single character, the form's focus shifts to this field. On Windows systems, the modifier key is the ALT key and on Mac OS systems, it is the OPTION key. For example, if the form author sets the accessKey of a field to `f` and a Windows user presses Alt+f, the focus shifts to that field. When designing forms that include accelerator keys, form designers should instruct the users about the availability of the accelerator keys.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.2

Examples

JavaScript

```
TextField1.accessKey = "f";
```

FormCalc

```
TextField1.accessKey = "f"
```

action

Identifies the form nodes that are protected by a document signature.

Syntax

Reference_Syntax.action = "include | exclude | all"

Values

Type	Values
String	• `include` (default) The document signature protects all the fillable form nodes in the specified collection. This option requires at least one valid ref child object whose text value is a reference syntax expression identifying the nodes that are protected by the document signature. • `exclude` The document signature protects all the fillable form nodes that are not in the specified collection. This option requires at least one valid ref child object whose text value is a reference syntax expression identifying the nodes that are protected by the document signature. • `all` The document signature protects all the fillable form nodes.

Applies to

Model	Object
Form Model	manifest

Version

XFA 2.4

activity

Specifies the name of the event.

The accompanying ref property must specify an object that can generate the named event.

Syntax

```
Reference_Syntax.activity = "change | click | docClose | docReady | enter |
exit | full | initialize | mouseDown | mouseEnter | mouseExit | mouseUp |
postExecute | postPrint | postSave | preExecute | prePrint | preSave |
preSubmit | ready"
```

Values

Type	Value
String	• `change` Occurs when the user changes the field value. Examples of when this occurs are: • With each key-stroke • When text is pasted • When a new choice is selected • When a check button is clicked • `click` (default) Occurs when the user clicks in the field. Most systems define click as pressing and releasing the mouse button while not moving the pointer beyond a very small threshold. • `docClose` Executes at the very end of processing a form, if, and only if, all form validations complete with no errors. This event comes too late to modify a saved document. The purpose is to provide the ability to generate an exit status or completion message. • `docReady` Executes prior to the rendering of the document, but after data binding of the data takes place. • `enter` For a field, occurs when the field gains keyboard focus. For a subform or exclusion group, occurs when some field within the subform or exclusion group gains keyboard focus, that is, keyboard focus moves from outside the object to inside it. • `enter` For a field, occurs when the field gains keyboard focus. For a subform or exclusion group, occurs when some field within the subform or exclusion group gains keyboard focus, that is, keyboard focus moves from outside the object to inside it. • `exit` For a field, occurs when the field loses keyboard focus. For a subform or exclusion group, occurs when all fields within the subform or exclusion group lose keyboard focus, that is, focus moves from inside the object to outside it. • `full` Initiates when the form filler attempts to enter more than the maximum allowed amount of content into a field. • `initialize` Executes after data binding is complete. A separate event is generated for each instance of the subform in the form model.

Type	Value
	• `mouseDown` Occurs when the user presses the mouse button in the field, but before the button is released. • `mouseEnter` Occurs when the user drags the pointer over the field without necessarily pressing the button. • `mouseExit` Occurs when the user drags the pointer out of the field without necessarily pressing the button. • `mouseUp` Occurs when the user releases the mouse button in the field. • `postExecute` Occurs when data is sent to a web service via WSDL, just after the reply to the request has been received and the received data is marshalled in a connectionData object underneath $datasets. A script triggered by this event has the chance to examine and process the received data. After execution of this event, the received data is deleted. • `postPrint` Occurs just after the rendered form has been sent to the printer, spooler, or output destination. • `postSave` Occurs just after the form has been written out in PDF or XDP format. Does not occur when the data model or some other subset of the form is exported to XDP. • `preExecute` Occurs when a request is sent to a web service via WSDL. A script triggered by this event has the chance to examine and alter the data before the request is sent. If the script is marked to be run only at the server, the data is sent to the server with an indication that it should run the associated script before performing the rest of the processing. • `preSave` Occurs just before the form data is written out in PDF or XDP format. Does not occur when the data model or some other subset of the form is exported to XDP. XSLT postprocessing, if enabled, occurs after this event.

Type	Value
	• `preSubmit` Occurs when data is submitted to the host via the HTTP protocol. A script triggered by this event can examine and alter the data before it is submitted. If the script is marked to run at the server, the data is sent to the server, with an indication that it should run the associated script before performing the rest of the processing. • `ready` Occurs when the model has finished loading.

Applies to

Model	Object
Form Model	event

Version

XFA 2.1

Examples

JavaScript

```
TextField1.event.activity = "mouseEnter";
```

FormCalc

```
TextField1.event.activity = "mouseEnter"
```

addRevocationInfo

Specifies whether the certificate status is checked when a digital signature is signed. The certificate status can be checked against a certificate revocation list (CRL) or an Online Certificate Status Protocol (OCSP) response.

The signing party must have access to the Internet to retrieve the CRL or OCSP response from the appropriate server.

The `addRevocationInfo` property does not have a default value so that Acrobat can override it if the value is not specified.

Syntax

`Reference_Syntax``.addRevocationInfo = "required | optional | none"`

Values

Type	Values
String	• `Required` Checking the certificate status is required. • `Optional` Checking the certificate status is optional. • `None` A CRL or OCSP response is not included in the digital signature.

Applies to

Model	Object
Form Model	filter

Version

XFA 2.5

after

Specifies the constraints on moving to a new page or content area after rendering the subform.

Syntax

Reference_Syntax.after = "auto | contentArea | pageArea | pageEven | pageFront | pageOdd"

Values

Type	Values
String	The behaviors described below can be further refined by optionally specifying a destination page or content area via the afterTarget property. • `auto` (default) The determination of a transition to a new page or content area will be delegated to the processing application. No transition to a new page or content area will be forced. • `contentArea` Rendering will transition to the next available content area. • `pageArea` Rendering will transition to a new page. • `pageBack` When duplexing, rendering will transition to the next available back surface, potentially causing an intervening page surface to be printed. If duplexing is not in effect, rendering will transition to a new page. • `pageEven` Rendering will transition to the next available even-numbered page, potentially causing intervening numbered or unnumbered pages to be printed. This behavior does not require duplexing. • `pageFront` When duplexing, rendering will transition to the next available front surface, potentially causing an intervening page surface to be printed. If duplexing is not in effect, rendering will transition to a new page. • `pageOdd` Rendering will transition to the next available odd numbered page, potentially causing intervening numbered or unnumbered pages to be printed. This behavior does not require duplexing.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.after = "pageOdd";
```

FormCalc

```
Subform1.break.after = "pageOdd"
```

afterTarget

Specifies the explicit destination page or content area for the after property.

Syntax

Reference_Syntax.afterTarget = "auto | contentArea | pageArea | pageEven | pageFront | pageOdd"

Values

Type	Values
String	The value of this property is expected to be compatible with the value of the after property. For instance, it would be considered an error for the after property to reference a page area and the afterTarget property to reference a content area, or vice versa.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.afterTarget = "pageEven";
```

FormCalc

```
Subform1.break.afterTarget = "pageEven"
```

aliasNode

Specifies the object that is represented by the alias for this model.

Syntax

Reference_Syntax.aliasNode = "*object*"

Values

Type	Values
Object	The object within the model referenced by the reference syntax for that model. In the case of the form model, the alias node would be the form object. For more information about reference syntax expressions, see "About referencing objects in calculations and scripts" on page 58.

Applies to

model class

Version

XFA 2.1

Examples

JavaScript
```
xfa.aliasNode = "form";
```

FormCalc
```
xfa.aliasNode = "form"
```

all

Returns a collection of like-named, in-scope nodes.

If the node has no name, a like-class named collection is returned.

Syntax
Reference_Syntax.all = "*object(s)*"

Values

Type	Values
Object	An object or a collection of objects.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript
```
Subform1.all;
```

FormCalc
```
Subform1.all
```

allowMacro

Specifies whether to permit the processing application to optimize output by generating a printer macro for all of the subform's draw content.

Syntax
Reference_Syntax.allowMacro = "1 | 0"

Values

Type	Values
String	• `1` (default) The processing application is permitted to utilize a printer macro for this subform. • `0` The processing application cannot utilize a printer macro for this subform.

Applies to

Model	Object
Form Model	subform

Version

XFA 2.1

Examples

JavaScript

```
Subform1.allowMacro = "0";
```

FormCalc

```
Subform1.allowMacro = "0"
```

allowNeutral

Specifies whether the check box or radio button can support an additional third state that represents a neutral value.

Syntax

Reference_Syntax`.allowNeutral = "0 | 1"`

Values

Type	Values
String	• `0` (default) The check box or radio button supports two states representing true or false. • `1` The check box or radio button supports three states. These are true, false, or neutral.

Applies to

Model	Object
Form Model	checkButton

Version

XFA 2.1

Examples

JavaScript

```
CheckBox1.resolveNode("ui.#checkButton").allowNeutral = "1";
```

FormCalc

```
CheckBox1.ui.#checkButton.allowNeutral = "1"
```

allowRichText

Specifies whether the text can include styling (also known as rich text).

Note: The `allowRichText` property only relays styling information to the application interface. The setting of this property in no way restricts a user from inputting plain text markup that includes styling information. For example, regardless of the setting of this property, a user could type:

```
<b>hello</b>
```

Syntax

Reference_Syntax`.allowRichText = "0 | 1"`

Values

Type	Values
String	• `0` (default) Text styling is invalid. This is the default when the `textEdit` object does not contain an `exData` object. • `1` Text styling is valid. This is the default when the `textEdit` object does contain an `exData` object.

Applies to

Version

XFA 2.1

Model	Object
Form Model	textEdit

Examples

JavaScript

```
TextField1.resolveNode("ui.#textEdit").allowRichText = "1";
```

FormCalc

```
TextField1.ui.#textEdit.allowRichText = "1"
```

anchorType

Specifies the location of the container's anchor point when it is placed by using a positioned layout strategy.

Syntax

Reference_Syntax.anchorType = "topLeft | topCenter | topRight | middleLeft | middleCenter | middleRight | bottomLeft | bottomCenter | bottomRight"

Values

Type	Values
String	• `topLeft` (default) Top left corner of the container. • `topCenter` Center of the top edge of the container. • `topRight` Top right corner of the container. • `middleLeft` Middle of the left edge of the container. • `middleCenter` Middle of the container. • `middleRight` Middle of the right edge of the container. • `bottomLeft` Bottom left corner of the container. • `bottomCenter` Center of the bottom edge of the container. • `bottomRight` Bottom right corner of the container.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.anchorType = "bottomRight";
```

FormCalc

```
TextField1.anchorType = "bottomRight"
```

appType

Specifies the name of the client application in which a form currently exists.

The `appType` property calls the `viewerType` property from the Acrobat JavaScript object model and returns the corresponding value for the client application in which the form exists. For example, in the context of a PDF form viewed in Adobe Reader®, this property returns `Reader`.

For more information on the `viewerType` property, and the values it returns, see the *JavaScript for Acrobat API Reference* at http://www.adobe.com/go/learn_lc_AcrobatDeveloper.

Syntax

```
Reference_Syntax.appType
```

Values

Type	Values
String	A valid string representing the name of the current hosting client application.

Applies to

Model	Object
Form Model	draw exclGroup field subform
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.appType;
```

FormCalc

```
xfa.host.appType
```

archive

Specifies the URI location of an archive file that may contain program code related to the `exObject` object.

Syntax

Reference_Syntax`.archive = "`*URI*`"`

Values

Type	Values
String	A fully qualified URI value.

Applies to

Model	Object
Form Model	exObject

Version

XFA 2.1

aspect

Specifies how the image is to map to the nominal content region of the image's container.

Syntax

Reference_Syntax`.aspect = "fit | none | actual | width | height"`

Values

Type	Values
String	• `fit` (default) The application scales the image proportionally to the maximum size of the container's content region. • `none` The application scales the image to the size of entire container's content region. This may result in different scale values being applied to the image's X and Y coordinates. • `actual` The image is rendered using the dimensions stored in the image content. The extent of the container's region does not affect the sizing of the image. • `width` The application scales the image proportionally to the width of the container's content region. The image might be taller or shorter than the content region. • `height` The application scales the image proportionally to the height of the container's content region. The image might be wider or narrower than the content region.

Applies to

Model	Object
Form Model	image

Version

XFA 2.1

Examples

JavaScript

```
ImageField1.resolveNode("value.#image").aspect = "actual";
```

FormCalc

```
ImageField1.value.#image.aspect = "actual"
```

baselineShift

Specifies a positive measurement that shifts a font up from the baseline or a negative measurement that shifts a font down from the baseline.

Syntax

Reference_Syntax.baselineShift = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.baselineShift = "-5pt";
```

FormCalc

```
TextField1.font.baselineShift = "-5pt"
```

before

Specifies the constraints on moving to a new page or content area before rendering the subform.

Syntax

```
Reference_Syntax.before = "auto | contentArea | pageArea | pageBack | pageEven
| pageFront | pageOdd"
```

Values

Type	Values
String	The behaviors described below can be further refined by optionally specifying a destination page or content area using the beforeTarget property. The startNew property also modifies some of these behaviors: • `auto` (default) The determination of a transition to a new page or content area is delegated to the processing application. No transition to a new page or content area is forced. • `contentArea` Rendering transitions to the next available content area. See also the startNew property. • `pageArea` Rendering transitions to a new page. See also the startNew property. • `pageBack` When duplexing, rendering transitions to the next available back surface, potentially causing an intervening page surface to print. If duplexing is not in effect, rendering transitions to a new page. Note that `pageBack`, unlike `pageEven`, is not affected by page numbering. • `pageEven` Rendering transitions to the next available even numbered page, potentially causing intervening numbered or unnumbered pages to print. This behavior does not require duplexing. • `pageFront` When duplexing, rendering transitions to the next available front surface, potentially causing an intervening page surface to be printed. If duplexing is not in effect, rendering will transition to a new page. Note that `pageFront`, unlike `pageOdd`, is not affected by page numbering. • `pageOdd` Rendering transitions to the next available odd numbered page, potentially causing intervening numbered or unnumbered pages to print. This behavior does not require duplexing.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.before = "contentArea";
```

FormCalc
```
Subform1.break.before = "contentArea"
```

beforeTarget

Specifies the explicit destination page or content area for the before property.

Syntax
Reference_Syntax.beforeTarget = "auto | contentArea | pageArea | pageEven | pageFront | pageOdd"

Values

Type	Values
String	The value of the beforeTarget property is expected to be compatible with the value of the before property. For instance, it would be considered an error for the before property to have a value of pageArea and the beforeTarget property to reference a content area, or vice versa.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript
```
Subform1.break.beforeTarget = "#contentArea_ID";
```

FormCalc
```
Subform1.break.beforeTarget = "#contentArea_ID"
```

bind

Specifies the name of a unique binding ID where columns from the data source specified by the from property are bound.

Syntax
Reference_Syntax.bind = "*string*"

Values

Type	Values
String	A valid string representing a binding ID.

Applies to

Model	Object
sourceSet Model	map

Version

XFA 2.1

binding

Identifies the type of application to which the script is directed.

Syntax

Reference_Syntax.binding = "XFA | *Application_type*"

Values

Type	Values
String	• `XFA` (default) The script is to be applied by standard application. • Any other valid application type. Any value other than `XFA` signifies that the script may be ignored by standard applications.

Applies to

Model	Object
Form Model	script

Version

XFA 2.1

Examples

JavaScript

```
TextField1.resolveNode("#event.#script").binding = "XFA";
```

FormCalc

```
TextField1.#event.#script.binding = "XFA"
```

blank (deprecated)

Specifies whether the page area is intended to be blank and therefore may result in special treatment by the output device.

Syntax

Reference_Syntax.blank = "0 | 1"

Values

Type	Values
String	• `0` (default) The page area is not intended to be blank, and any content is rendered. • `1` The page area is intended to be blank, and may be subject to special treatment by the output device. For example, a printer may charge the user on a per-printed-page basis. The user does not wish to be charged for blank backsides of printed pages on a duplexed job. This property permits the blank backsides of the document to be marked blank with the result that the processing application must not render any content on the backside and the printer may receive special instructions to ensure that the blank backside is not counted towards the user's charges.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.1

Examples

The reference syntax expression will vary, depending on the object from which it is invoked.

JavaScript

```
xfa.form.form1.pageSet.Page1.blank;
```

FormCalc

```
xfa.form.form1.pageSet.Page1.blank
```

blankOrNotBlank

Specifies whether the page area is intended to be blank and therefore may result in special treatment by the output device.

Syntax

Reference_Syntax.blankOrNotBlank = "any | blank | notBlank"

Values

Type	Values
String	• `any` (default) Matches any blank or non-blank page. • `blank` Matches a page which is inserted by a break-to-even page while on an even page, or a break-to-odd page while on an odd page. • `notBlank` Matches any page inserted either to hold content or to meet minimum occurrence rules.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.5

Examples

The reference syntax expression will vary, depending on the object from which it is invoked.

JavaScript

```
xfa.form.form1.pageSet.Page1.blankOrNotBlank = "notBlank";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.blankOrNotBlank = "notBlank"
```

bofAction

Specifies the action to perform if the current record is the first record in the record set.

Syntax

Reference_Syntax`.bofAction = "moveLast | stayEOF"`

Values

Type	Values
String	• `moveLast` (default) Moves the current record position to a point after the last record. • `stayEOF` The current record will always be the last record in the record set.

Applies to

Model	Object
sourceSet Model	recordSet

Version

XFA 2.1

bookendLeader

Specifies a subform to place into the current content area or page before any other content.

If both the `bookendLeader` and bookendTrailer properties are supplied, the two subforms surround the content like bookends.

Syntax

Reference_Syntax.bookendLeader = "*string*"

Values

Type	Values
String	A valid string representing the name or fully qualified reference syntax expression of a subform.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.bookendLeader = "xfa.form.form1.Subform2";
```

FormCalc

```
Subform1.break.bookendLeader = "xfa.form.form1.Subform2"
```

bookendTrailer

Identifies a subform to place into the current content area or page after any other content.

If both bookendLeader and `bookendTrailer` properties are supplied, the two subforms surround the content like bookends.

Syntax

Reference_Syntax.bookendTrailer = "*string*"

Values

Type	Values
String	A valid string representing the name or fully qualified reference syntax expression of a subform.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.bookendTrailer = "xfa.form.form1.Subform2";
```

FormCalc

```
Subform1.break.bookendTrailer = "xfa.form.form1.Subform2"
```

borderColor

Specifies the border color value for this field.

A border must be defined before you can change the color by scripting.

Syntax

Reference_Syntax.borderColor = " *[0-255], [0-255], [0-255]*"

Values

Type	Values
String	For the color-space of SRGB, the component values must be r,g,b, where r is the red component value, g is the green component value, and b is the blue component value. Each component value must be in the range 0 through 255, inclusive. 255 represents maximum display intensity. For example, 255,0,0 specifies the color red. The default is dependent upon the context of where the color is used; the default color is determined by the object enclosing the color object.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.borderColor = "125,154,125";
```

FormCalc

```
TextField1.borderColor = "125,154,125"
```

borderWidth

Specifies the border width for this field.

Syntax

Reference_Syntax.borderWidth = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.borderWidth = "0.05in";
```

FormCalc

```
TextField1.borderWidth = "0.05in"
```

bottomInset

Specifies the size of the bottom inset.

Syntax

Reference_Syntax.bottomInset = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	margin

Version

XFA 2.1

Examples

JavaScript

```
Subform1.margin.bottomInset = "1in";
```

FormCalc

```
Subform1.margin.bottomInset ="1in"
```

break

Describes the constraints on moving to a new page or content area after rendering an object.

Syntax

```
Reference_Syntax.break = "close | open"
```

Values

Type	Values
String	• `close` (default) • `open`

Applies to

Model	Object
Form Model	border

Version

XFA 2.1

Examples

JavaScript

```
Subform1.border.break = "open";
```

FormCalc

```
Subform1.border.break = "open"
```

calculationsEnabled

Specifies whether calculate scripts will execute.

Syntax

```
Reference_Syntax.calculationsEnabled= "0 | 1"
```

Values

Type	Values
Boolean	• `1` (default) The calculate scripts execute. • `0` The calculate scripts do not execute.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.calculationsEnabled = "1";
```

FormCalc

```
xfa.host.calculationsEnabled = "1"
```

cap

Specifies the rendered termination of the stroke.

Strokes that form an enclosed area do not have a rendered termination. In particular, all rectangle and border edges, as well as all 360-degree arc edges, are not considered to have any termination. Arcs with sweep angles less than 360 degrees and lines do have terminations at both endpoints.

Syntax

`Reference_Syntax``.cap = "square | butt | round"`

Values

Type	Values
String	• `square` (default) The stroke terminates by rendering the end of the edge squarely beyond the edge's endpoint a distance equal to one-half the edge's thickness. • `butt` The stroke terminates by rendering the end of the edge squarely across the endpoint. • `round` The stroke terminates by rendering the end of the edge with a semi-circle at the edge's endpoint, having a radius equal to one-half the edge's thickness.

Applies to

Model	Object
Form Model	edge

Version

XFA 2.1

Examples

JavaScript

```
Line1.resolveNode("value.#line.edge").cap = "round";
```

FormCalc

```
Line1.value.#line.edge.cap = "round"
```

change

Specifies the value that a user types or pastes into a field immediately after they perform the action.

Syntax

```
Reference_Syntax.change
```

Values

Type	Values
String	Any valid string value appropriate for a particular form field.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.change;
```

FormCalc

```
xfa.event.change
```

charEncoding

Specifies the character encoding of the value that is encoded into a barcode.

The value of the barcode field is serialized into a sequence of bytes according to the specified character encoding. Then it is compressed if the dataPrep property requires it and encrypted if the `encrypt` object is present. Finally, it is encoded according to the symbology.

Note: The value of this property is case-insensitive and must match one of the following values.

Syntax

Reference_Syntax.charEncoding = "UTF-8 | none | ISO-8859-1 | ISO-8859-2 | SO-8859-7 | SHift-JIS | KSC-5601 | Big-Five | GB-2312 | UTF-16 | UCS-2 | fontSpecific"

Values

Type	Values
String	• UTF-8 (default) The characters are encoded using Unicode code points as defined by Unicode, and UTF-8 serialization as defined by ISO/IEC 10646. • none No special encoding is specified. The characters are encoded using the ambient encoding for the operating system. • ISO-8859-1 The characters are encoded using ISO-8859-1, also known as Latin-1. • ISO-8859-2 The characters are encoded using ISO-8859-2. I • SO-8859-7 The characters are encoded using ISO-8859-7. • Shift-JIS The characters are encoded using JIS X 0208, more commonly known as Shift-JIS. • KSC-5601 The characters are encoded using the Code for Information Interchange (Hangul and Hanja). • Big-Five The characters are encoded using Traditional Chinese (Big-Five). There is no official standard for Big-Five and several variants are in use. The Adobe form object model uses the variant implemented by Microsoft® as code. • GB-2312 The characters are encoded using Simplified Chinese. • UTF-16 The characters are encoded using Unicode code points as defined by Unicode, and UTF-16 serialization as defined by ISO/IEC 10646.

Type	Values
	• `UCS-2` The characters are encoded using Unicode code points as defined by Unicode, and UCS-2 serialization as defined by ISO/IEC 10646. • `fontSpecific` The characters are encoded in a font-specific way. Each character is represented by one 8-bit byte.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.4

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").charEncoding = "UCS-2";
```

FormCalc

```
Code11BarCode1.ui.#barcode.charEncoding = "UCS-2"
```

checksum

Specifies an algorithm for the checksum to insert into the barcode.

The calculation of the checksums is based on the barcode data.

The template model allows any one of the choices listed below. However, some barcode formats either require a particular checksum or never allow a checksum. For such barcodes, the `checksum` property is ignored. Some of the remaining barcode formats support only a limited subset of these choices. For such barcodes, the template model does not specify an unsupported choice.

Syntax

`Reference_Syntax``.checksum = "none | auto | 1mod10 | 2mod10 | 1mod10_1mod11"`

Values

Type	Values
String	• `none` (default) Do not insert a checksum. • `auto` Insert the default checksum for the barcode format. • `1mod10` Insert a 1 modulo 10 checksum. • `2mod10` Insert a 2 modulo 10 checksum. • `1mod10_1mod11` Insert a 1 modulo 10 checksum followed by a 1 modulo 11 checksum. **Note:** 1 modulo 10, 2 modulo 10, and 1 modulo 11 are barcode standards. Refer to documentation on those standards for more information on those barcodes.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").checksum  = "2mod10";
```

FormCalc

```
Code11BarCode1.ui.#barcode.checksum  = "2mod10"
```

circular

Enables you to convert an arc into a circle.

Syntax

```
Reference_Syntax.circular = "0 | 1"
```

Values

Type	Values
Boolean	• `0` (default) Do not adjust the arc to a circular path. • `1` Adjust the arc to a circular path. **Note:** You can convert an arc into a circle even if the content area where the arc is located is not square. If necessary, the size of the circle is adjusted to match the size of the content area.

Applies to

Model	Object
Form Model	arc

Version

XFA 2.1

Examples

JavaScript

```
Circle1.resolveNode("value.#arc").circular = "1";
```

FormCalc

```
Circle1.value.#arc.circular = 1
```

classAll

Returns a collection of like-class, in-scope nodes.

Note: This property is read only.

Syntax

Reference_Syntax.classAll = "*objects*"

Values

Type	Values
Object	A set of objects derived from the same class as the current object and also within the same scope.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
Subform1.classAll;
```

FormCalc

```
Subform1.classAll
```

classId

Specifies a URI name or location for the program code represented by the object.

Syntax

```
Reference_Syntax.classId = "URI"
```

Values

Type	Values
String	Any fully qualified URI value.

Applies to

Model	Object
Form Model	exObject

Version

XFA 2.1

classIndex

Returns the position of this object in its collection of like-class, in-scope objects.

Note: This property is read only.

Syntax

```
Reference_Syntax.classIndex = "integer"
```

Values

Type	Values
Integer	An integer representing the 0 based index position of the current object in relation to the set of objects in the same scope that derive from the same class.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
Subform1.classIndex;
```

FormCalc

```
Subform1.classIndex
```

className

Determines the name of the class of this object.

Note: This property is read only.

Syntax

Reference_Syntax.className = "*string*"

Values

Type	Values
String	A valid string representing the name of the class of the particular object.

Applies to

object class

Version

XFA 2.1

Examples

JavaScript

```
Subform1.className;
```

FormCalc

```
Subform1.className
```

codeBase

Specifies a URI location that can be used to assist the resolution of a relative classId property.

Syntax

Reference_Syntax.codeBase = "*URI*"

Values

Type	Values
String	A fully qualified URI value.

Applies to

Model	Object
Form Model	exObject

Version

XFA 2.1

codeType

Specifies an identifier corresponding to a MIME type that identifies the program code represented by the object.

Syntax

Reference_Syntax.codeType = "*MIME-type*"

Values

Type	Values
String	A valid MIME-type identifier. For example `application/java`.

Applies to

Model	Object
Form Model	exObject

Version

XFA 2.1

colSpan

Specifies the number of columns spanned by this object when used inside a subform with a layout type of row.

Syntax

Reference_Syntax.colSpan = "1 | *integer*"

Values

Type	Values
String	• `1` (default) • Any valid integer value.

Applies to

Model	Object
Form Model	area draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
StaticText1.colSpan = "1";
```

FormCalc

```
StaticText1.colSpan = "1"
```

columnWidths

Specifies the widths for columns of a table.

The `columnWidth` property is ignored unless the layout property is set to `table`.

Syntax

`Reference_Syntax``.columnWidth = "`*`measurement`* | `-1 [, [, `*`measurement`* | `-1 ] ]"`

Values

Type	Values
String	The value of this property is a set of space-separated tokens. Each token must be a valid measurement or `-1`. The presence of a measurement causes the corresponding column to be set to that width. The presence of `-1` causes the corresponding column to grow to the width of the widest content for that column across all rows of the table.

Applies to

Model	Object
Form Model	subform

Version

XFA 2.1

Examples

JavaScript

```
Subform1.columnWidths = ".5in 1.5in";
```

FormCalc

```
Subform1.columnWidths = ".5in 1.5in"
```

commandType

Specifies the type of command used by a data query.

Syntax

Reference_Syntax.commandType = "unknown | text | table | storedProc"

Values

Type	Values
String	• unknown (default) • text An explicit SQL query string that is not saved under a name in the database. • table A table stored in the database. • storedProc A query, such as a SQL query, created to query one or more tables in the database and then saved as a named query within the database.

Applies to

Model	Object
sourceSet Model	query

Version

XFA 2.1

Examples

In these examples, Titles represents the data connection name.

JavaScript

```
xfa.sourceSet.Titles.nodes.item(1).query.setAttribute("text", "commandType");
```

FormCalc

```
xfa.sourceSet.Titles.nodes.item(1).query.setAttribute("text", "commandType")
```

commitKey

Describes how the current value of a form field was set by the user.

Syntax

Reference_Syntax.commitKey = "0 | 1 | 2 | 3"

Values

Type	Values
Integer	• `0` (default) The value was not set (for example, if the user presses the escape key prior to the form field losing focus). • `1` The value is set when a user left-clicks outside the field. • `2` The value is set when a user presses the enter key. • `3` The value is set when a user tabs to a new field.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.commitKey = "2";
```

FormCalc

```
xfa.event.commitKey = "2"
```

commitOn

Specifies when a user's selections are propagated to the data model.

Syntax

```
Reference_Syntax.commitOn = "select | exit"
```

Values

Type	Values
String	• `select` The selected data is written to the data model when a user selects a choice list entry with a keyboard or mouse. **Note:** Having a choice list commit data as soon as selections are made may be important in forms that contain non-XFA interactive features, such as Acrobat annotations or hypertext links. People filling out such forms may erroneously believe that selecting an item from a choice list followed by clicking a non-XFA interactive feature is the same as exiting the check list. In fact, the check list remains the field in focus. • `exit` The selected data is not written to the data model until the field loses focus. This is the recommended setting for choice lists that support multiple selections (open="multiSelect").

Applies to

Model	Object
Form Model	choiceList

Version

XFA 2.2

Examples

JavaScript

```
DropDownList1.resolveNode("ui.#choiceList").commitOn = "exit";
```

FormCalc

```
DropDownList1.ui.#choiceList.commitOn = "exit"
```

connection

Specifies the name of the associated connection control in the connection set.

The connection named by this property must point to a web service.

Syntax

Reference_Syntax.connection = "*string*"

Values

Type	Values
String	A valid string representing the name of the associated connection object in the connection set. If this property is missing or empty the connection name defaults to the name of the containing subform.

Applies to

Model	Object
Form Model	bindItems connect execute setProperty
sourceSet Model	connect

Version

XFA 2.4

Examples

JavaScript

```
TextField1.resolveNode("#connect").connection = "DataConnection";
```

FormCalc

```
TextField1.#connect.connection = "DataConnection"
```

contains

Determines whether a data value should be included in value of the parent object or as a property of the parent.

Syntax

Reference_Syntax.contains = "data | metaData"

Values

Type	Values
String	• `data` (default) Value is included in the value of the parent object • `metaData` Value is a property of the parent object.

Applies to

Model	Object
Data Model	dataValue

Version

XFA 2.1

content

Specifies the content of the object.

Syntax

Reference_Syntax.content = "*string*"

Values

Type	Values
String	A valid string representing the content of the object. For packets that contain XML content, this should return an empty string.

Applies to

Model	Object
XFA Model	packet

Version

XFA 2.1

Examples

JavaScript

```
xfa.packet.content = "";
```

FormCalc

```
xfa.packet.content = ""
```

contentType

Specifies the type of content in the referenced document, expressed as a MIME type.

Syntax

Reference_Syntax.contentType = "text/plain | application/x-formcalc | *Mime-type*"

Values

Type	Values
String	The following values are allowed for documents containing text: • `text/plain` (default) Unadorned text. The application may accept content that does not conform strictly to the requirements of the MIME type. • `application/x-formcalc` A FormCalc script. • Any valid MIME-type. Support for other text types, such as `text/html` as well as scripting types such as `application/x-ecmascript` is implementation-defined. When the referenced document is an image, a suitable MIME-type must be supplied for this property to tell the application that the content is an image. However, the application is free to override the supplied value if upon examining the image data it determines that the image data is of a different type. Which image types are supported is implementation-defined.

Applies to

Model	Object
Data Model	dataValue
Form Model	exData image script
sourceSet Model	bind

Version

XFA 2.1

Examples

JavaScript

```
ImageField1.resolveNode("value.#image").contentType =
"application/x-formcalc";
```

FormCalc

```
ImageField1.value.#image.contentType = "application/x-formcalc"
```

context (deprecated)

Specifies the current object, which is the starting object for the resolveNode and resolveNodes methods.

Syntax

Reference_Syntax.content = "*object*"

Values

Type	Values
Object	The current object.

Applies to

model class

Version

XFA 2.1

count

Specifies the current number of subform instances on a form.

Syntax

Reference_Syntax.count = "*integer*"

Values

Type	Values
Integer	• *integer* An integer greater than or equal to 0 indicating the number of subform instances on the form.

Applies to

Model	Object
Form Model	instanceManager

Version

XFA 2.5

Examples

JavaScript

```
Subform1.instanceManager.count;
```

FormCalc

```
Subform1.instanceManager.count
```

credentialServerPolicy

Specifies whether checking the certificate status is required when a digital signature is signed. The certificate status can be checked against a certificate revocation list (CRL) or an Online Certificate Status Protocol (OCSP) response.

Syntax

Reference_Syntax.credentialServerPolicy = "Optional | Required"

Values

Type	Values
String	• `Optional` (default) Including the CRL or OCSP response is optional. • `Required` Including the CRL or OCSP response is required.

Applies to

Model	Object
Form Model	certificates

Version

XFA 2.5

crlSign

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.crlSign = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

cSpace

Specifies the color space.

The default color space, and currently the only space permitted, is `SRGB`.

Syntax
Reference_Syntax`.cSpace = "SRGB"`

Values

Type	Values
String	`SRBG` (default) **Note:** This is the only supported value.

Applies to

Model	Object
Form Model	color

Version

XFA 2.1

Examples

JavaScript
```
TextField1.border.edge.color.cSpace = "SRGB";
```

FormCalc
```
TextField1.border.edge.color.cSpace = "SRGB"
```

currentPage

Sets the currently active page of a document at run time.

Page values are 0-based, so the first page of a document returns a value of `0`.

The `currentPage` property is available when `layout:ready` executes on a client. However, it is not available when `layout:ready` executes on the server because the property will not execute until the form layout executes.

Syntax
`Reference_Syntax.currentPage = "`*`integer`*`"`

Values

Type	Values
Integer	A valid integer representing a specific page of the document.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.currentPage = "2";
```

FormCalc

```
xfa.host.currentPage = "2"
```

See also

"Working with page numbers and page counts" on page 426

currentRecordNumber

Returns the current record number within the range of records contained by the current dataWindow object.

Syntax

Reference_Syntax.currentRecordNumber = "*integer*"

Values

Type	Values
Integer	Any valid integer value.

Applies to

Model	Object
Data Model	dataWindow

Examples

JavaScript

```
xfa.dataWindow.currentRecordNumber = "2"; // The third record
```

FormCalc

```
xfa.dataWindow.currentRecordNumber = "2" // The third record
```

currentValue

Returns the correctly typed object for the property.

Syntax

Reference_Syntax.currentValue = "*typed object*"

Values

Type	Values
Depends on the type of the property	The correctly typed object for the property.

Applies to

Model	Object
Form Model	delta

Version

XFA 2.1

cursorLocation

Indicates the location of the cursor library to use with the record set.

Syntax

Reference_Syntax.cursorLocation = "client | server"

Values

Type	Values
String	• client (default) Cursor library is located on the client computer. • server Cursor library is located on the server computer.

Applies to

Model	Object
sourceSet Model	recordSet

Version

XFA 2.1

cursorType

Specifies the type of cursor to use when opening the record set.

Syntax

Reference_Syntax.cursorType = "forwardOnly | keyset | dynamic | static | unspecified"

Values

Type	Values
String	• `forwardOnly` (default) Identical to a static cursor, except that scrolling occurs only in a forward direction. This improves performance when you need to make only one pass through a record set. • `keyset` Similar to a dynamic cursor, except that records that other users add are not visible. Data changes by other users are visible. • `dynamic` Additions, changes, and deletions by other users are visible, and all types of movement through the record set are permitted, except for bookmarks, if the provider does not support them. • `static` A static copy of a set of records that can be used to find data or generate reports. Additions, changes, or deletions by other users are not visible. • `unspecified` The type of cursor is not specified.

Applies to

Model	Object
sourceSet Model	recordSet

Version

XFA 2.1

data

Indicates whether the image provided to the widget should be represented as a reference or should be embedded.

The `data` property affects the object behavior when the form is filled.

Syntax

`Reference_Syntax``.data = "link | embed"`

Values

Type	Values
String	• `link` The image is represented as a URI reference. If the user provides the widget with a URI, the href attribute of the container's image object is updated to reflect the new URI. If the image object was previously loaded with an embedded image, that image is removed from the object. • `embed` The image is embedded in the container's image object. If the user provides the widget with a URI, the image referenced by the URI is embedded as the content of the image object.

Applies to

Model	Object
Form Model	imageEdit

Version

XFA 2.1

Examples

JavaScript

```
TextField1.resolveNode("ui.#imageEdit").data = "embed";
```

FormCalc

```
TextField1.ui.#textEdit.data = "embed"
```

dataColumnCount

Specifies an optional number of data columns to encode for supported barcodes. This property applies to two-dimensional (2D) barcodes only.

The form design must supply this property in conjunction with the dataRowCount property to specify a fixed row and column barcode, otherwise the parser must use the rowColumnRatio property to determine the row and column count. The template must not supply the dataColumnCount property unless the dataRowCount property is also supplied. When these properties are used the size of the barcode is fixed. If the supplied data does not fill the barcode it is padded out with padding symbols.

Syntax

Reference_Syntax.dataColumnCount = "*string*"

Values

Type	Values
String	A valid string representing the number of data columns to encode.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").dataColumnCount = "3";
```

FormCalc

```
Code11BarCode1.ui.#barcode.dataColumnCount = "3"
```

dataDescription

Specifies the name of a data connection description to use with a particular type of web services connection.

Syntax

Reference_Syntax.dataDescription = "*string*"

Values

Type	Values
String	A valid string representing the name of a data description to use while exporting data.

Applies to

Model	Object
connectionSet Model	wsdlConnection xmlConnection xsdConnection

Version

XFA 2.1

dataEncipherment

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.dataEncipherment = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

dataLength

Specifies the maximum number of characters for this instance of the barcode. This property applies to one-dimensional barcodes only.

For software barcodes, when the moduleWidth property is not specified, the `dataLength` property must be supplied by the form design. For hardware barcodes, this property is ignored.

The data being displayed is not validated. For software barcodes, the application allows the data to overflow the assigned region of the field. For hardware barcodes, the result of an overflow depends on the printer.

Note: There is no corresponding minimum length restriction. Some barcode formats have a fixed number of symbols and must be filled out with padding characters. Others allow a variable number of symbols and must terminate after the last symbol.

Syntax

Reference_Syntax.dataLength = "*string*"

Values

Type	Values
String	A valid string representing the maximum number of characters for this barcode instance. Each barcode type has its own default length value.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").dataLength = "10";
```

FormCalc

```
Code11BarCode1.ui.#barcode.dataLength = "10"
```

dataPrep

Defines preprocessing that is applied to the data written in the barcode.

It does not affect the data in the object models, nor does it affect what the user sees when the field has focus in interactive contexts.

Note: Recommended for 2D barcodes only.

Syntax

Reference_Syntax.dataPrep = "none | flateCompress"

Values

Type	Values
String	• none (default) Uses the data as supplied. • flateCompress Writes a header consisting of a byte with decimal value 257, followed by another byte with decimal value 1. It then writes the data compressed with the Flate algorithm, as defined by the Internet Engineering Task Force (IETF) in RFC1951. It does not use a predictor algorithm. Do not specify this option with a type that cannot encode arbitrary binary data.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").dataPrep = "flateCompress";
```

FormCalc

```
Code11BarCode1.ui.#barcode.dataPrep = "flateCompress"
```

dataRowCount

Specifies an optional number of data rows to encode for supported barcodes. This property applies to 2D barcodes only.

The form design can supply this property in conjunction with the dataColumnCount property to specify a fixed row and column barcode. Otherwise the rowColumnRatio property plus the actual length of the data being inserted determine the row and column count. The dataRowCount property cannot be present unless the dataColumnCount property is also present. When these properties are used the size of the barcode is fixed. If the supplied data does not fill the barcode the remaining cells are padded out with padding symbols.

Syntax

Reference_Syntax.dataRowCount = "*string*"

Values

Type	Values
String	A valid string representing the number of data rows to encode.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").dataRowCount = "2";
```

FormCalc

```
Code11BarCode1.ui.#barcode.dataRowCount = "2"
```

db

Specifies the name of a database available from the provider.

Syntax

Reference_Syntax.db = "*string*"

Values

Type	Values
String	A valid string representing the database name.

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

decipherOnly

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.decipherOnly = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

delayedOpen

Specifies the number of seconds to delay opening the data source after a connection is made.

Syntax

Reference_Syntax.delayedOpen = "*string*"

Values

Type	Values
String	A valid string representing the number of seconds.

Applies to

Model	Object
sourceSet Model	command

Version

XFA 2.1

Examples

In these examples, `Titles` represents the data connection name.

JavaScript

```
xfa.sourceSet.Titles.connect.delayedOpen = "5";
```

FormCalc

```
xfa.sourceSet.Titles.connect.delayedOpen = "5"
```

delimiter

Separates the attributes in the Subject DN string.

Syntax

Reference_Syntax.delimiter = ", | *string*"

Values

Type	Values
String	• , (default) • A valid string that separates the attributes in the Subject DN string.

Applies to

Model	Object
Form Model	subjectDN

Version

XFA 2.5

digitalSignature

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.digitalSignature = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

disable

Inhibits the audible prompt.

Syntax

Reference_Syntax`.disable = "1 | 0"`

Values

Type	Values
String	• `1` (default) An audible prompt is produced if the field is not hidden or invisible. • `0` There is not be an audible prompt.

Applies to

Model	Object
Form Model	speak

Version

XFA 2.1

Examples

JavaScript

```
TextField1.assist.speak.disable = "0";
```

FormCalc

```
TextField1.assist.speak.disable = "0"
```

editValue

Specifies the edit value for the field.

Syntax

Reference_Syntax.editValue = "*string*"

Values

Type	Values
String	A valid string representing the edit value for the field.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

embedPDF

Determines whether PDF file will be included as part of the data.

Syntax

Reference_Syntax.embedPDF = "0 | 1"

Values

Type	Values
String	• 0 (default) The PDF file is sent as part of in the data. • 1 The PDF file is not sent as part of the data. A URI is sent in its place.

Applies to

Model	Object
Form Model	submit

Version

XFA 2.1

Examples

JavaScript

```
Button1.resolveNode("#event.#submit").embedPDF = "1";
```

FormCalc

```
Button1.#event.#submit.embedPDF = "1"
```

encipherOnly

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.encipherOnly = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

endChar

Specifies an optional ending control character to append to barcode data.

The `endChar` property is ignored by the parser if the barcode pattern does not support the specified control character.

Syntax

Reference_Syntax.endChar = "*character*"

Values

Type	Values
String	A valid control character.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").endChar = "*";
```

FormCalc

```
Code11BarCode1.ui.#barcode.endChar = "*"
```

eofAction

Specifies the action to perform if the current record is the last record in the record set.

Syntax

Reference_Syntax.eofAction = "moveLast | stayEOF | addNew"

Values

Type	Values
String	• `moveLast` (default) Moves the current record position to a point after the last record. • `stayEOF` The current record will always be the last record in the record set. • `addNew` Adds a new record to the record set.

Applies to

Model	Object
sourceSet Model	recordSet

Version

XFA 2.1

errorCorrectionLevel

Specifies an optional error correction level to apply to supported barcodes. This property applies to 2D barcodes only.

Note: For barcode types that accept this property, the parser ignores the checksum.

Syntax
Reference_Syntax.errorCorrectionLevel = "0 | *integer*"

Values

Type	Values
String	• 0 (default) • For PDF417, the valid values are integers in the range 0 through 8, inclusive.

Applies to

Model	Object
Form Model	barcode

Version
XFA 2.1

Examples

JavaScript
```
Code11BarCode1.resolveNode("ui.#barcode").errorCorrectionLevel = "5";
```

FormCalc
```
Code11BarCode1.ui.#barcode.errorCorrectionLevel = "5"
```

executeType

Specifies whether to import new data into the existing form or merge new data with the original form design to create a new form.

Syntax
Reference_Syntax.executeType = "import | remerge"

Values

Type	Values
String	• import (default) Imports data into the current form without merging that data with the form design. • remerge Merges the data in the connectionData dataset with the form design. The merge process creates dynamic subforms, if necessary, depending on the data returned by the web service.

Applies to

Model	Object
Form Model	execute

Version

XFA 2.1

Examples

JavaScript

```
Button1.resolveNode("#event.#execute").executeType = "remerge";
```

FormCalc

```
Button1.#event.#execute.executeType = "remerge"
```

fillColor

The background color value for this field.

A fill color must be defined before you can change the color.

Syntax

```
Reference_Syntax.fillColor = "[0-255], [0-255], [0-255]"
```

Values

Type	Values
String	For the color-space of SRGB, the component values must be r,g,b, where r is the red component value, g is the green component value, and b is the blue component value. Each component value must be in the range 0 through 255, inclusive. 255 represents maximum display intensity. For example, 255,0,0 specifies the color red. The default is dependent upon the context of where the color is used; the default color is determined by the object enclosing the color object.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.fillColor = "150,130,33";
```

FormCalc

```
TextField1.fillColor = "150,130,33"
```

See also

"Changing the background color" on page 428

fontColor

The foreground color value for the field.

Syntax

Reference_Syntax.fontColor = "*string*"

Values

Type	Values
String	A valid string that represents the font color.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.fontColor = "150,130,33";
```

FormCalc

```
TextField1.fontColor = "150,130,33"
```

format

Determines the format in which to submit the data.

Syntax

Reference_Syntax.format = "pdfEnvelope | xmlEnvelope"

Values

Type	Values
String	For the `encrypt` object: • `pdfEnvelope` Adds the contents being submitted to a PDF document as an encrypted attachment. • `xmlEnvelope` Encrypts the contents being submitted using W3C XML encryption and contains them within an XML envelope.
String	For the `submit` object: • `xdp` (default) The data is packaged in XDP format. • `formdata` The data is packaged in URL-encoded format as described in Uniform Resource Locators (URL). • `pdf` The data is packaged in PDF as described in the Adobe PDF Specifications.

Applies to

Model	Object
Form Model	encrypt submit

Version

XFA 2.1

Examples

JavaScript

```
Button1.resolveNode("#event.#submit").format = "pdf"
```

FormCalc

```
Button1.#event.#submit.format = "pdf"
```

formatMessage

Specifies the format validation message string for this field.

Syntax

Reference_Syntax.formatMessage = "*string*"

Values

Type	Values
String	A valid string representing the format validation message.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.formatMessage = "Please use the format: LASTNAME, FIRSTNAME";
```

FormCalc

```
TextField1.formatMessage = "Please use the format: LASTNAME, FIRSTNAME"
```

formattedValue

Specifies the formatted value for the field.

Syntax

Reference_Syntax.formattedValue = "*string*"

Values

Type	Values
String	A valid string representing the value of the field with formatting, including picture formats and symbols.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
TextField2.rawValue = TextField1.formattedValue;
```

FormCalc

```
TextField2 = TextField1.formattedValue
```

See also

"Getting or setting object values" on page 425

formatTest

Controls validation against the display picture clause.

The `formatTest` property can be used for validations. For more information, see "Validation" on page 40.

Syntax

Reference_Syntax`.formatTest = "warning | disabled | error"`

Values

Type	Values
String	• `disabled` Do not perform any test. The form object is permitted to have a value that does not conform to the picture clause. The field can be left with a non-conforming value and it will not invalidate the form. • `error` Emit a message and refuse to accept data that does not fit the picture clause. The form object must conform to a picture clause. • `warning` (default) Emit a message if the data does not fit the picture clause, but allow the user to proceed to the next field. The message must inform the user that the form object should have a value that conforms to the picture clause. It must provide two choices: • `dismiss`: The user understands the message and wants to return to the form to satisfy this constraint. • `override`: The user understands the message, but chooses to contravene this constraint.

Applies to

Model	Object
Form Model	validate

Version

XFA 2.1

Examples

Set the validation pattern if has not already been defined.

JavaScript

```
TextField1.validate.picture.value = "A9A 9A9";
TextField1.validate.formatTest = "error";
```

FormCalc

```
TextField1.validate.picture = "A9A 9A9"
TextField1.validate.formatTest = "error"
```

fracDigits

Specifies the maximum number of digits (inclusively) following the decimal point to capture and store.

Syntax

Reference_Syntax.fracDigits = "2 | *integer*"

Values

Type	Values
String	• 2 (default) • A string representing any valid integer value.

Applies to

Model	Object
Form Model	decimal

Version

XFA 2.1

Examples

The numeric field data type should be set to decimal.

JavaScript

```
NumericField1.resolveNode("value.#decimal").fracDigits = "3";
```

FormCalc

```
NumericField1.value.#decimal.fracDigits = "3"
```

from

Specifies the original column name in the data source.

Syntax

Reference_Syntax.from = "*string*"

Values

Type	Values
String	A valid string representing the name of the column in the data source where data will be mapped from.

Applies to

Model	Object
sourceSet Model	map

Version

XFA 2.1

fullText

Represents the full (untruncated) value that a user pastes into a form field.

Fields may truncate pasted text if it exceeds the allowable content region. The `fullText` property stores the untruncated value in memory for use with scripting operations.

The value of the newContentType determines the content type of this property.

Syntax

```
Reference_Syntax.fullText = "string"
```

Values

Type	Values
String	Any valid string value.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.fullText;
```

FormCalc

```
xfa.event.fullText
```

h

A measurement of the height for the layout.

When height is specified as a measurement, that value overrides any growth range allowed by the minH property and the maxH property. When this property is omitted or set to an empty string, the growth range is set by the minH property and the maxH property.

Syntax

```
Reference_Syntax.h = "0in | measurement"
```

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.h = "2in";
```

FormCalc

```
TextField1.h = "2in"
```

hAlign

Specifies the horizontal text alignment.

Syntax

```
Reference_Syntax.hAlign = "left | center | right | justifyAll | justify |
radix"
```

Values

Type	Values
String	• `left` (default) Align with the left edge of the available region. • `center` Center horizontally within the available region. • `right` Align with the right edge of the available region. • `justifyAll` Spread-justify all lines to fill the available region. • `justify` Left-align the last line and spread-justify the rest.

Applies to

Model	Object
Form Model	draw exclGroup field para subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.hAlign = "right";
```

FormCalc

```
TextField1.para.hAlign = "right"
```

hand

Describes the justification of a line or edge.

Syntax

Reference_Syntax.hand = "even | left | right"

Values

Type	Values
String	• `even` (default) Center the displayed line on the underlying vector or arc. • `left` Position the displayed line immediately to the left of the underlying vector or arc, when following that line from its start point to its end point. • `right` Position the displayed line immediately to the right of the underlying vector or arc, when following that line from its start point to its end point.

Applies to

Model	Object
Form Model	arc border line rectangle

Version

XFA 2.1

Examples

JavaScript

```
Line1.resolveNode("value.#line").hand = "left";
```

FormCalc

```
Line1.value.#line.hand = "left"
```

highlight

Specifies the visual appearance of a button when activated by a user. All values support two states (up and down) except `push` which supports three states (up, down, and rollover).

Syntax

`Reference_Syntax``.highlight="none | inverted | push | outline"`

Values

Type	Values
String	• `push` (default) • `none` • `inverted` • `outline` **Note:** Buttons that are set to highlight mode "push" can assign different captions to the alternate button states (down and rollover).

Applies to

Model	Object
Form Model	button

Version

XFA 2.5

Examples

JavaScript

```
Button1.resolveNode("ui.#button").highlight = "push";
```

FormCalc

```
Button1.ui.#button.highlight = "push"
```

href

Specifies a reference to an external file or resource.

The transferEncoding property does not apply to external images.

Syntax

Reference_Syntax`.href = "`*URL*`"`

Values

Type	Values
String	A valid HTML reference. For example: • `http://www.adobe.com/data` • `ftp://255.255.0.0/dataFiles`

Applies to

Model	Object
Form Model	exData image

Version

XFA 2.1

Examples

JavaScript

```
ImageField1.resolveNode("value.#image").href = "/E/dev/Logos/adobe.jpg";
```

FormCalc

```
ImageField1.value.#image.href = "/E/dev/Logos/adobe.jpg"
```

hScrollPolicy

Specifies whether a field can scroll horizontally.

Note: This property does not apply to Text Fields that can expand to accommodate data or text.

Syntax

Reference_Syntax.hScrollPolicy = "auto | on | off"

Values

Type	Values
String	• `auto` (default) Single-line fields scroll horizontally and multi-line fields scroll vertically (displaying a vertical scroll bar when necessary). • `on` Vertical and/or horizontal scroll bars appear regardless of whether the text or data overflows the boundaries of the field. • `off` Restricts the user from entering characters in the field beyond what can physically fit within the field width. Note that this restriction does not apply to data with the field.

Applies to

Model	Object
Form Model	dateTimeEdit

Version

XFA 2.5

Examples

JavaScript

```
TextField1.resolveNode("ui.#textEdit").hScrollPolicy = "off";
```

FormCalc

```
TextField1.ui.#textEdit.hScrollPolicy = "off"
```

id

Specifies a generic user-defined XML ID type.

Syntax

Reference_Syntax.id = "*string*"

Values

Type	Values
String	A valid string representing a user-defined XML identification.

Applies to

node class

Version

XFA 2.1

Examples

In these examples, CurrentPageNumber is a floating field. It is one type of object that has an identification.

imagingBBox

Specifies a region within the medium that is available for rendering with four comma separated measurements representing the measurements for x, y, width, and height.

Syntax

Reference_Syntax.bind = "none | *x, y, width, height*"

Values

Type	Values
String	• none (default) The entire area of the paper is available for rendering. • x, y, width, height The content of the subform is not available for manipulation by the user. A user-agent should treat the subform as a pass-through container in sequencing operations, and you must not be permitted to modify the content of the subform. The content of the subform is still modifiable via indirect means such as scripting operations and calculations.

Applies to

Model	Object
Form Model	medium

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.medium = "100, 100, 50, 50";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.medium = "100, 100, 50, 50"
```

index

Returns the position of this node in its collection of like-named, in-scope nodes.

If the node has no name, the position in its like-class named collection is returned.

Syntax

Reference_Syntax.index = "*integer*"

Values

Type	Values
Integer	An integer representing the 0 based index position of the current object relative to objects of the same name within the same scope.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
Subform1.parent.index;
```

FormCalc

```
Subform1.parent.index
```

See also

- "Referencing objects" on page 420
- "Manipulating instances of a subform" on page 424
- "Changing the background color" on page 428

initial

Specifies the initial number of occurrences for the enclosing container.

Syntax

Reference_Syntax.initial = "*1* | *string*"

Values

Type	Values
String	• `1` (default) • A valid string representing any valid integer.

Applies to

Model	Object
Form Model	occur

Version

XFA 2.1

Examples

Modifying the `occur` object on the `form:ready` event is too late in the form life cycle. It needs to be modified on the `template:ready` event. However, the `template:ready` event is not accessible in the user interface.

JavaScript

```
Subform1.occur.initial = "3";
```

FormCalc

```
Subform1.occur.initial = "3"
```

initialNumber

Supplies the initial page number to the first page in a group of consecutive pages that use the same pageSet.

When you use separate numbering runs within a single document, use `initialNumber` to control the initial number of each run. For example you can use i - iv for the table of contents, followed by 1 - 27 for the body of the document.

Syntax

`Reference_Syntax``.initialNumber = "1 | `*`string`*`"`

Values

Type	Values
String	• `1` (default) • A valid string representing any integer.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.initialNumber = "4";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.initialNumber = "4"
```

input

Specifies an input message associated with a particular WSDL connection operation.

Syntax

Reference_Syntax.input = "*string*"

Values

Type	Values
String	A valid string representing an input message.

Applies to

Model	Object
connectionSet Model	operation

Version

XFA 2.1

instanceIndex

Calculates the index of a subform or subform set based on where it is located relative to other instances of the same form object.

Syntax

Reference_Syntax.instanceIndex = "*integer*"

Values

Type	Values
Integer	A valid integer representing the zero-based index of the specified subform or subform set.

Applies to

Model	Object
Form Model	subform subformSet

Version

XFA 2.5

intact

Specifies the constraints on keeping a subform intact within a content area or page.

Syntax

Reference_Syntax.intact = "none | contentArea | pageArea"

Values

Type	Values
String	• none (default) The determination of whether a subform will be rendered intact within a content area or page is delegated to the processing application. It is possible that the subform could be split across a content area or page. This is the default when the parent container's layout is tb, lr-tb, or table. • contentArea The subform is requested to be rendered intact within a content area. This is the default when the parent container's layout is position or row. • pageArea The subform is requested to be rendered intact within a page. **Note:** There is no single default value for this property. Instead it is context-sensitive. When the parent container's layout is tb, lr-tb, or table the default value is none. When the parent container's layout is position or row, the default value is contentArea. The default is computed at the moment the API call to get the value is made or at the moment the layout operation is invoked.

Applies to

Model	Object
Form Model	keep

Version

XFA 2.1

Examples

JavaScript

```
Subform1.keep.intact = "pageArea";
```

FormCalc

```
Subform1.keep.intact = "pageArea"
```

inverted

Specifies whether the corner appears convex (it joins the edges tangentially) or is inverted and appears concave (it joins the edges at right angles).

Syntax

Reference_Syntax.inverted = "0 | 1"

Values

Type	Values
String	• 0 (default) The corner appears convex. • 1 The corner appears concave.

Applies to

Model	Object
Form Model	corner

Version

XFA 2.1

Examples

JavaScript

```
TextField1.border.corner.inverted = "1";
```

FormCalc

```
TextField1.border.corner.inverted = "1"
```

isContainer

Specifies whether this object is a container object.

Note: This property is read only.

Syntax

Reference_Syntax.isContainer = "True | False"

Values

Type	Values
Boolean	• `True` (default) The object is a type of container object. • `False` The object is not a type of container object.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
TextField1.isContainer;
```

FormCalc

```
TextField1.isContainer
```

isDefined

Indicates whether a valid data window is currently defined.

A data window is considered valid if the current record index points to a record within the data. A data window is not defined if there are no records, or if the current record index is beyond the end of the range of records.

Note: This property is read only.

Syntax

Reference_Syntax`.isDefined = "True | False"`

Values

Type	Values
Boolean	• `True` (default) The current data window is defined. • `False` The current data window is not defined.

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript

```
xfa.dataWindow.isDefined;
```

FormCalc

```
$dataWindow.isDefined
```

isNull

Indicates whether the current data value is the null value.

Syntax

Reference_Syntax.isNull = "True | False"

Values

Type	Values
Boolean	• `True` (default) The current data value is the null value. • `False` The current data window is not the null value.

Applies to

node class

Model	Object
Data Model	dataValue

Version

XFA 2.1

Examples

JavaScript

```
TextField1.isNull = "False";
```

FormCalc

```
TextField1.isNull = "False"
```

join

Specifies the shape of the corner.

Syntax
Reference_Syntax.join = "square | round"

Values

Type	Values
String	• square (default) The corner has the shape of a right-angle between the adjoining edges. • round The corner has the shape of a round curve between the adjoining edges.

Applies to

Model	Object
Form Model	corner

Version
XFA 2.1

Examples

JavaScript
```
TextField1.border.corner.join = "round";
```

FormCalc
```
TextField1.border.corner.join = "round"
```

keyAgreement

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax
Reference_Syntax.keyAgreement = "Yes | No | *empty_string*"

Values

Type	Values
String	• Yes (default) The value must be set in the certificate for it to be acceptable. • No The value must not be set in the certificate for it to be acceptable. • "" If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

keyCertSign

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.keyCertSign = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

keyDown

Determines whether a user is pressing an arrow key to make a selection. This property is available only for list boxes and drop-down lists.

Syntax

Reference_Syntax.keyDown = "True | False"

Values

Type	Values
String	• `True` (default) Arrow key was used to make the selection. • `False` Arrow key was not used to make the selection.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.keyDown;
```

FormCalc

```
xfa.event.keyDown
```

keyEncipherment

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.keyEnciphement = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

labelRef

Resolves a data value for each data node in the set identified by the `ref` object.

The data values are then used to populate the label items, such as `<items save='0'>`.

The `labelRef` property is a relative reference syntax expression.

The `labelRef` property is optional. You might want to define a list using only a set of values with no labels. In that case, the rendered object uses labels that default to the actual values.

Syntax

`Reference_Syntax``.labelRef = "`*`string`*`"`

Values

Type	Values
String	A string representing a data value for each data node in the set.

Applies to

Model	Object
Form Model	bindItems

Version

XFA 2.4

language

Returns the language of the running host application.

Syntax

```
Reference_Syntax.language
```

Values

Type	Values
String	A valid string representing the locale language of the host computer.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.language;
```

FormCalc

```
xfa.host.language
```

layout

Specifies the layout strategy to be used by this object.

Syntax

Reference_Syntax`.layout = "position | lr-tb | rl-tb | row | table | tb"`

Values

Type	Values
String	• `position` (default) The content of the control is positioned according to the to the location information expressed on the content objects. • `lr-tb` The content of the object flows from left to right and top to bottom. • `rl-tb` Reserved for future use. The content of the object flows from right to left and top to bottom. • `row` This is an inner object of a table, representing one or more rows. The objects contained in this object are cells of the table and their height and width properties, if any, are ignored. The cells are laid out from right to left and each one is adjusted to the height of the row and the width of one or more contiguous columns. • `table` This is the outer object of a table. Each of its child subforms or exclusion groups must have its layout property set to row. The rows of the table are laid out from top to bottom. • `tb` The content of the object flows from top to bottom.

Applies to

Model	Object
Form Model	exclGroup subform

Version

XFA 2.1

Examples

JavaScript

```
Subform1.layout = "tb";
```

FormCalc

```
Subform1.layout = "tb"
```

See also

- "Referencing objects" on page 420
- "Working with page numbers and page counts" on page 426
- "Disabling all form fields" on page 434

leadDigits

Specifies the maximum number of digits (inclusively) preceding the decimal point to capture and store.

Syntax

Reference_Syntax.leadDigits = "0 | *integer*"

Values

Type	Values
String	• 0 (default) • A valid string representing any integer value.

Applies to

Model	Object
Form Model	decimal

Version

XFA 2.1

Examples

For these examples, the numeric field data type should be set to decimal.

JavaScript

```
NumericField1.resolveNode("value.#decimal").leadDigits = "2";
```

FormCalc

```
NumericField1.value.#decimal.leadDigits = "2"
```

leader

Specifies the `subform` or `subformSet` object to place at the top of a content or page area.

The leader property replaces the deprecated overflowLeader and bookendLeader properties.

Syntax
Reference_Syntax.leader = "*string*"

Values

Type	Values
String	A valid string representing the ID or fully qualified reference syntax expression of a subform or subform set. The default is an empty string.

Applies to

Model	Object
Form Model	bookend breakAfter breakBefore overflow

Version

XFA 2.4

Examples

JavaScript
```
Subform1.leader = "xfa.form.form1.Subform2";
```

FormCalc
```
Subform1.leader = "xfa.form.form1.Subform2"
```

leftInset

Specifies a the size of the left inset.

Syntax
Reference_Syntax.leftInset = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	margin

Version

XFA 2.1

Examples

JavaScript

```
Subform1.margin.leftInset = "0.25in";
```

FormCalc

```
Subform1.margin.leftInset = "0.25in"
```

length

Specifies the number of objects in the list.

Note: This property is read only.

Syntax

Reference_Syntax.length

Values

Type	Values
Integer	A valid integer representing the number of objects.

Applies to

list class

Version

XFA 2.5

Examples

JavaScript

```
// Display the number of child nodes under root node.
xfa.host.messageBox("Number of nodes under rootNode after appending clone: " +
xfa.record.nodes.length);
```

FormCalc

```
// Display the number of child nodes under root node.
xfa.host.messageBox("Number of nodes under rootNode after appending clone: " +
xfa.record.nodes.length)
```

See also

- "Referencing objects" on page 420
- "Creating a node in the data model" on page 422
- "Calculating totals" on page 428
- "Changing the background color" on page 428
- "Populating a drop-down list" on page 430
- "Disabling all form fields" on page 434

lineHeight

Specifies the line height to apply to the paragraph content.

Omitting a value or specifying an empty value indicates that the font size determines the line height.

Syntax

Reference_Syntax.lineHeight = "0pt | *measurement*"

Values

Type	Values
String	• 0pt (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.lineHeight = "20pt";
```

FormCalc

```
TextField1.para.lineHeight = "20pt"
```

lineThrough

Specifies the activation of a single or double line extending through the text (also known as strikethrough).

Syntax

Reference_Syntax.lineThrough = "0 | 1 | 2"

Values

Type	Values
String	• 0 (default) The font renders without a line through the text. • 1 The font renders with a single line through the text. • 2 The font renders with a double line through the text.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.lineThrough = "2";
```

FormCalc

```
TextField1.font.lineThrough = "2"
```

lineThroughPeriod

Controls the appearance of the line extending through the text (also known as strikethrough).

Syntax

Reference_Syntax.lineThroughPeriod = "all | word"

Values

Type	Values
String	• `all` (default) The rendered line shall extend across word breaks. • `word` The rendered line shall be interrupted at word breaks.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.lineThroughPeriod = "word";
```

FormCalc

```
TextField1.font.lineThroughPeriod = "word"
```

locale

Specifies the language, currency, and time/date formatting to use for the content of the object.

The locale affects the representation of data formatted, validated, or normalized by picture clauses. When this property is absent or empty, the default behavior is to inherit the parent object's locale. If the outermost subform does not specify a locale, the default behavior derives from the ambient locale of the operating system. If the operating system does not supply a locale, `en_US` is used.

Syntax

`Reference_Syntax``.locale = "ambient | `*`locale`*`"`

Values

Type	Values
String	• `ambient` (default) The application uses its own ambient locale. • A valid locale name, for example `en_US`. For a complete list of valid locale values, refer to the IETF RFC 1766 and ISO 639/ISO 3166 specifications.

Applies to

Model	Object
Form Model	draw field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.locale = "en_US";
```

FormCalc

```
TextField1.locale = "en_US"
```

lockType

Specifies the type of locking functionality to use with the data source.

Syntax

`Reference_Syntax``.lockType = "unspecified | readOnly | pessimistic | optimistic | batchOptimistic"`

Values

Type	Values
String	• `unspecified` (default) Does not specify a type of lock. • `readOnly` Indicates read-only records. Data cannot be altered. • `pessimistic` Records are locked at the data source immediately after editing. • `optimistic` Records are locked only when a user-instigated update of the data occurs. • `batchOptimistic` Indicates optimistic batch updates. This is required for batch update mode.

Applies to

Model	Object
sourceSet Model	recordSet

Version

XFA 2.1

Examples

In these examples, `Titles` represents the data connection name.

JavaScript

```
xfa.sourceSet.Titles.nodes.item(1).query.recordSet.lockType = "optimistic";
```

FormCalc

```
xfa.sourceSet.Titles.nodes.item(1).query.recordSet.lockType = "optimistic"
```

long

Specifies the length of the long edge of the medium. The length specified by the `long` property must be greater than the length specified by the short property.

Syntax

`Reference_Syntax``.long = "0in | measurement"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	medium

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.medium.long = "4in";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.medium.long = "4in"
```

mandatory

Specifies the nullTest value for the field.

Syntax

```
Reference_Syntax.mandatory = "string"
```

Values

Type	Values
String	A string that represents the null test value.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
Textfield1.mandatory = "error";
```

FormCalc

```
TextField1.mandatory = "error"
```

mandatoryMessage

Specifies the mandatory message string for this field.

Syntax

```
Reference_Syntax.mandatoryMessage = "string"
```

Values

Type	Values
String	A string that represents the mandatory message.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
Textfield1.mandatoryMessage = "This field is required.";
```

FormCalc

```
TextField1.mandatoryMessage = "This field is required."
```

marginLeft

Specifies the size of the left indentation of the paragraph.

Syntax

Reference_Syntax.marginLeft = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.marginLeft = "0.5in";
```

FormCalc

```
TextField1.para.marginLeft = "0.5in"
```

marginRight

Specifies the size of the right indentation of the paragraph.

Syntax

Reference_Syntax.marginRight = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.marginRight = "0.5in";
```

FormCalc

```
TextField1.para.marginRight = "0.5in"
```

mark

Indicates the shape to use when filling a Check Box object.

Syntax

Reference_Syntax.mark = "default | check | circle | cross | diamond | square | star"

Values

Type	Values
String	• `default` (default) The default marks vary depending on the shape of the Checkbox object. A corner to corner for square and a filled circle for round. The new marks are font-based symbols. • `check` • `circle` • `cross` • `diamond` • `square` • `star`

Applies to

Model	Object
Form Model	checkButton

Version

XFA 2.5

Examples

JavaScript

```
CheckBox1.resolveNode("ui.#checkButton").mark = "diamond";
```

FormCalc

```
CheckBox1.ui.#checkButton.mark = "diamond"
```

match

Controls the role played by enclosing an object in a data-binding (merge) operation.

Syntax

```
Reference_Syntax.mark = "once | none | global | dataref"
```

Values

Type	Values
String	• `once` (default) The node representing the enclosing object binds to a node in the Data model in accordance with the standard matching rules. • `none` The node representing the enclosing object is transient. It will not be bound to any node in the Data model. • `global` The containing field is global. If the normal matching rules fail to provide a match for it, the data-binding process looks outside the current record for data to bind to the field. • `dataRef` The containing field binds to the node in the Data model specified by the accompanying ref property.

Applies to

Model	Object
Form Model	bind
sourceSet Model	bind

Version

XFA 2.1

Examples

You should set the field global property before the merge.

JavaScript

```
TextField1.bind.match = "global";
```

FormCalc

```
TextField1.bind.match = "global"
```

max

Specifies the maximum number of occurrences for the enclosing container, or `-1` to set no upper boundary for occurrences.

The `max` property defaults to the value of the min property. In the absence of a min property, the default is `1`.

Syntax

Reference_Syntax.max = "1 | -1 | *integer*"

Values

Type	Values
String	• `1` (default) • `-1` No upper boundary limit. • Any valid integer.

Applies to

Model	Object
Form Model	instanceManager occur
sourceSet Model	recordSet

Version

XFA 2.1

Examples

JavaScript

```
Subform1.occur.max = "3";
```

FormCalc

```
Subform1.occur.max = "3"
```

maxChars

Specifies the maximum number of characters that this text value can enclose.

Syntax

Reference_Syntax.maxChars = "0 | *integer*"

Values

Type	Values
String	• `0` (default) • Any valid integer value. **Note:** If you do not specify a value for this property, or if the value is an empty string, there is no maximum.

Applies to

Model	Object
Form Model	text

maxH

Specifies the maximum height for layout purposes.

If you do not specify a value for this property, there is no upper limit. If you specify a value for the h property, the container cannot grow vertically and this property is ignored.

Syntax

Reference_Syntax.maxH = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.maxH = "3in";
```

FormCalc

```
TextField1.maxH = "3in"
```

maxLength

Specifies the maximum (inclusive) allowable length of the content or -1 to indicate that no maximum length is imposed.

The interpretation of this property is affected by the content type. In this case this property specifies the maximum (inclusive) allowable length of the content in characters. For instance, where the content type is text/plain this property represents the maximum (inclusive) number of characters of plain text content. Similarly, where the content type is text/html this property represents the maximum (inclusive) number of characters of content excluding markup, and insignificant whitespace.

Syntax

Reference_Syntax.maxLength = "-1 | *integer*"

Values

Type	Values
String	• `-1` (default) • Any valid integer value.

Applies to

Model	Object
Form Model	exData

Version

XFA 2.1

maxW

Specifies the maximum width for layout purposes.

If you do not specify a value for this property, there is no maximum. If you specify a value for the w property, the container cannot grow horizontally and this property is ignored.

Syntax

Reference_Syntax`.maxW = "0in | `*measurement*`"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.maxW = "3in";
```

FormCalc

```
TextField1.maxW = "3in"
```

min

Specifies the minimum number of occurrences for the enclosing container.

Syntax

Reference_Syntax.min = "1 | *integer*"

Values

Type	Values
String	• 1 (default) • Any valid integer.

Applies to

Model	Object
Form Model	instanceManager occur

Version

XFA 2.1

Examples

JavaScript

```
Subform1.occur.min = "0";
```

FormCalc

```
Subform1.occur.min = "0"
```

See also

"Manipulating instances of a subform" on page 424

minH

Specifies the minimum height for layout purposes.

If you supply a value for the h property, the container cannot grow vertically and this property is ignored.

Syntax

Reference_Syntax.minH = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.minH = "0.5in";
```

FormCalc

```
TextField1.minH = "0.5in"
```

minW

Specifies the minimum width for layout purposes.

If you supply a value for the w property, the container cannot grow horizontally and this property is ignored.

Syntax

Reference_Syntax.minW = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.minW = "0.5in";
```

FormCalc
```
TextField1.minW = "0.5in"
```

model

Specifies the model for the current object.

Note: This property is read only.

Syntax
Reference_Syntax.model

Values

Type	Values
Object	The root object for the particular XML Form Object Model, such as connectionSet or dataModel.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript
```
xfa.model.name;
```

FormCalc
```
xfa.model.name
```

modifier

Determines whether the modifier key (for example, Ctrl on Microsoft Windows®) is held down when a particular event executes.

Syntax
Reference_Syntax.modifier

Values

Type	Values
Boolean	• `True` (default) Modifier key is held down during event execution. • `False` Modifier key is not held down during event execution.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.modifier;
```

FormCalc

```
xfa.event.modifier
```

moduleHeight

Determines the height of a set of bars used to encode one character of supplied text.

The allowable range of heights varies from one barcode pattern to another. The form design must not specify a height outside the allowable range.

Syntax

Reference_Syntax.moduleHeight = "5mm | *measurement*"

Values

Type	Values
String	• 5mm (default for 2D barcodes) • Any valid measurement. When this property is not supplied, the default behavior depends on the type of barcode. One-dimensional barcodes grow to the height of the enclosing field, limited by the allowable height range. 2D barcodes default to a module height of 5mm.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").moduleHeight = "5mm";
```

FormCalc

```
Code11BarCode1.ui.#barcode.moduleHeight = "5mm"
```

moduleWidth

Specifies different aspects of a barcode depending on the class of barcodes being used.

For one-dimensional software barcodes the parser sets the width of the narrow bars to the value of this property. The width of the wide bars is derived from that of the narrow bars. The allowable range of widths varies from one barcode format to another. The form design must not specify a value outside the allowable range. If `moduleWidth` is supplied, then the dataLength property is ignored. Conversely moduleWidth has no default, so when the dataLength property is not supplied, then `moduleWidth` must be supplied.

For 2D hardware barcodes, `moduleWidth` either has no effect or has the same effect as for a software barcode, depending upon the printer and barcode. The allowable range for the value varies between printers and between barcodes.

For 2D barcodes the value of this property determines the module width. A module is a set of bars encoding one symbol. Usually a symbol corresponds to a character of supplied data. The allowable range of widths varies from one barcode format to another. The form design must not specify a value outside the allowable range.

Syntax

Reference_Syntax.moduleWidth = "0.25mm | *measurement*"

Values

Type	Values
String	• `0.25mm` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").moduleWidth = "25mm";
```

FormCalc

```
Code11BarCode1.ui.#barcode.moduleHeight = "25mm"
```

multiLine

Specifies whether the text may span multiple lines.

The `multiLine` property is useful for clients such as HTML browsers that have two types of text editing interfaces.

Syntax
Reference_Syntax.multiLine = "1 | 0"

Values

Type	Values
String	• 1 (default) The text may span multiple lines. • 0 The text is limited to a single line.

Applies to

Model	Object
Form Model	textEdit

Version

XFA 2.1

Examples

JavaScript
```
TextField1.resolveNode("ui.#textEdit").multiLine = "0";
```

FormCalc
```
TextField1.ui.#textEdit.multiLine = "0"
```

See also

"Concatenating data values" on page 427

name

Specifies an identifier that may be used to specify this object or event in script expressions.

For example, this property specifies the name of the host application, and on an interactive PDF form, it returns Acrobat.

Syntax
Reference_Syntax.name

Values

Type	Values
String	A string up to 255 characters.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.name;
```

FormCalc

```
xfa.host.name
```

See also

- "Referencing objects" on page 420
- "Changing the background color" on page 428

newContentType

Specifies the content type of the newText property.

For example, if newContentType='text/html', newText will contain an XHTML fragment.

Syntax

```
Reference_Syntax.newContentType = "allowRichText | plainTextOnly"
```

Values

Type	Values
String	• `allowRichText` (default) The field supports rich text. • `plainTextOnly` The field does not support rich text. Even if markup is present in the data, it should be passed through rather than interpreted. However, it is not guaranteed whether downstream processing will respond to the markup.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.newContentType = "plainTextOnly";
```

FormCalc

```
xfa.event.newContentType = "plainTextOnly"
```

newText

Specifies the content of the field after it changes in response to user actions.

Syntax

Reference_Syntax.newtext = "*string*"

Values

Type	Values
String	A string up to 255 characters.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
TextField2.rawValue = xfa.event.newText;
```

FormCalc

```
TextField2 = xfa.event.newText
```

See also

- "Referencing objects" on page 420
- "Populating a drop-down list" on page 430

next

Specifies the constraints on keeping a subform together with the next subform within a content area or page.

Syntax

Reference_Syntax.next = "none | contentArea | pageArea"

Values

Type	Values
String	• `none` (default) The determination of whether a subform is rendered in the same content area or page together with the next subform is delegated to the processing application. No special keep constraints will be forced. • `contentArea` The subform is requested to be rendered in the same content area with the next subform. • `pageArea` The subform is requested to be rendered in the same page with the next subform.

Applies to

Model	Object
Form Model	keep

Version

XFA 2.1

Examples

JavaScript

```
Subform1.keep.next = "contentArea";
```

FormCalc

```
Subform1.keep.next = "contentArea"
```

nodes

Returns a list of all child objects of the current object.

Note: This property is read only.

Syntax

Reference_Syntax.nodes

Values

Type	Values
Object	A list of XML Form Object Model objects.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
Subform1.nodes; // Single line example

// This example displays the names of the children of Subform1
var oNodes = this.nodes;
var nodesLength = oNodes.length;

for (var i = 0; i < nodesLength; i++) {
   xfa.host.messageBox(oNodes.item(i).name)
}
```

FormCalc

```
Subform1.nodes // Single line example

// This example displays the names of the children of Subform1
var oNodes = Subform1.nodes
var nodesLength = oNodes.length;

for (var i = 0; i < nodesLength; i++) {
   xfa.host.messageBox(oNodes.item(i).name)
}
```

See also

- "Creating a node in the data model" on page 422
- "Changing the background color" on page 428
- "Populating a drop-down list" on page 430

nonRepudiation

Specifies an acceptable key usage extension that must be present in the signing certificate.

Syntax

Reference_Syntax.nonRepudiation = "Yes | No | *empty_string*"

Values

Type	Values
String	• `Yes` (default) The value must be set in the certificate for it to be acceptable. • `No` The value must not be set in the certificate for it to be acceptable. • `""` If unspecified or specified as an empty string, the certificate's attribute is disregarded.

Applies to

Model	Object
Form Model	keyUsage

Version

XFA 2.5

ns

Returns the namespace for the object.

If the particular object is the root of a model, then this property returns the namespace for the model.

Note: This property is read only.

Syntax

Reference_Syntax.ns

Values

Type	Values
Object	A valid string representing the namespace of the current object, or the namespace of the current model if the root object is the currently selected object.

Applies to

node class

Version

XFA 2.1

nullTest

Controls whether a field is mandatory on a form or if it can be left empty.

The `nullTest` property can be used for validations. For more information about validations, see "Validation" on page 40.

Syntax

Reference_Syntax.nullTest = "disabled | error | warning"

Values

Type	Values
String	• `disabled` (default) Do not perform this test (default). The form object is permitted to have a value of null. The field can be left without a value and it will not negatively impact the validity of the form. This value disables the validation test. • `error` Emit an error message and refuse to accept an empty field. The form object is required to have a non-null value. • `warning` Emit a warning message if the field is empty, but allow the user to proceed to the next field. The message must inform the user that the form object is recommended to have a value, and provide two choices: • `dismiss`: The user understands the form's recommendation and wishes to return to the form and satisfy this constraint. • `override`: The user understands the form's recommendation, but has chosen to contravene this constraint.

Applies to

Model	Object
Form Model	validate

Version

XFA 2.1

Examples

JavaScript

```
TextField1.validate.nullTest = "error";
```

FormCalc

```
TextField1.validate.nullTest = "error"
```

numbered

Specifies whether the page area is considered a numbered page area.

Numbered page areas contribute to the normal incrementing of page numbers, whereas unnumbered pages occur without incrementing page numbering.

Syntax

Reference_Syntax.numbered = "auto | none"

Values

Type	Values
String	• `auto` (default) The page area represents a numbered page area. Therefore the instantiation of the page area contributes to the incrementing of the current page area number. • `none` The page area does not contribute to the incrementing of the current page area numbering.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.numbered = "none";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.numbered = "none"
```

numberOfCells

Indicates the number of cells drawn for a comb field. This is not affected by the number of characters in the field's value.

Syntax

Reference_Syntax.numberOfCells = "0 | *integer*"

Values

Type	Values
Integer	• `0` (default) A single cell is drawn for the comb field, or if the maxChars property is set, the number of cells corresponds to the value of maxChars. • *integer* A valid integer representing the total number of cells drawn for the comb field.

Applies to

Model	Object
Form Model	comb

Version

XFA 2.5

Examples

JavaScript

```
TextField1.resolveNode("ui.#textEdit.comb").numberOfCells = "6";
```

FormCalc

```
TextField1.ui.#textEdit.comb.numberOfCells = "6"
```

numPages

Returns the number of pages in the current document.

Syntax

Reference_Syntax.numPages

Values

Type	Values
Integer	A valid integer representing the total number of pages.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.numPages;
```

FormCalc

```
xfa.host.numPages
```

See also

- "Referencing objects" on page 420
- "Working with page numbers and page counts" on page 426
- "Disabling all form fields" on page 434

oddOrEven

Specifies whether a page is odd or even for pagination within a set of pages.

Syntax

Reference_Syntax.oddOrEven = "any | odd | even"

Values

Type	Values
String	• `any` (default) Matches any page within a document. • `odd` Matches the first page within a document and every other page after that, irrespective of page numbering. • `even` Matches the second page within a document and every other page after that, irrespective of page numbering.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.5

Examples

The reference syntax expression will vary, depending on the object from which it is invoked.

JavaScript

```
xfa.form.form1.pageSet.Page1.oddOrEven = "even";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.oddOrEven = "even"
```

oneOfChild

Retrieves or sets that child object in the case where a parent object can only have one of a particular child object.

Syntax

Reference_Syntax.oneOfChild = "*object*"

Values

Type	Values
Object	The one of child object.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
TextField1.value.oneOfChild;
```

FormCalc

```
TextField1.value.oneOfChild
```

See also

- "Referencing objects" on page 420
- "Concatenating data values" on page 427

open

Determines when the choice list is presented by interactive applications.

Syntax

Reference_Syntax.open = "userControl | onEntry | always | multiSelect"

Values

Type	Values
String	• `userControl` (default) The list drops down when the user clicks on a button or makes some other appropriate gesture. The list disappears when the cursor moves outside the list or some other appropriate user-interface event occurs. • `onEntry` The list drops down on entry into the field. It disappears upon exit from the field. • `always` The list is displayed when the field is visible. • `multiSelect` The user can select multiple entries from the list by pressing the Shift key while making selections. The list of choices is displayed when the field is visible.

Applies to

Model	Object
Form Model	choiceList

Version

XFA 2.1

Examples

JavaScript

```
DropDownList1.resolveNode("ui.#choiceList").open = "always";
```

FormCalc

```
DropDownList1.ui.#choiceList.open = "always"
```

operation

Indicates which signature operation to perform or when a link was used.

Syntax

```
Reference_Syntax.operation = "next | back | down | first | left | right | up"
```

Values

Type	Values
String	For the `signData` object: • `sign` Add an XML signature to the XML data being submitted. This operation does not modify the application's active document. • `verify` Verifies an XML signature. If the verification fails, the submission processes are canceled and the application issues a message indicating why the submission failed. This operation is performed before any signature is created or cleared. • `clear` Removes an XML signature, if it exists, from the XML data being submitted. This operation does not modify the application's active document and is performed before any signature is created.
	For the `traverse` object: • `next` (default) Used when the user presses the Tab key or enters the final character in a fixed-width field. However, the same chain of next links is also traversed by the screen reader when reading the form. Defaults to left-to-right top-to-bottom order. The chain of next links can include boilerplate objects, but these objects cannot accept input focus. Therefore, when advancing focus to the next form object, tabbing continues until an object that accepts input focus is reached. You must ensure that the form design does not present a non-terminating loop. • `back` Used when the user presses Shift+Tab. Defaults to right-to-left bottom-to-top order. • `down` Destination when the user presses the Down Arrow key. Defaults to top-to-bottom order. • `first` This property is used only when the container is a subform or subform set. The link points to the object that gains focus when the container is entered. In effect the container delegates focus via this link. Defaults to the first container that is a child of this container, in top-to-bottom left-to-right order. • `left` Destination when the user presses the Left Arrow key. Defaults to right-to-left order. • `right` Destination when the user presses the Right Arrow key. Defaults to left-to-right order. • `up` Destination when the user presses the Up Arrow key. Defaults to bottom-to-top order.

Applies to

Model	Object
connectionSetModel	wsdlConnection
Form Model	signData traverse

Version

XFA 2.4

orientation

Specifies the orientation of the medium.

Syntax

Reference_Syntax.orientation = "portrait | landscape"

Values

Type	Values
String	• `portrait` (default) The orientation of the medium places the short edge at the top. • `landscape` The orientation of the medium places the long edge at the top.

Applies to

Model	Object
Form Model	medium

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.medium.orientation = "landscape";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.medium.orientation = "landscape"
```

output

Specifies the output message associated with a particular WSDL connection operation.

Syntax

Reference_Syntax.output = "*string*"

Values

Type	Values
String	A valid string representing the output message.

Applies to

Model	Object
connectionSet Model	operation

Version

XFA 2.1

Examples

JavaScript

```
xfa.connectionSet.DataConnection.operation.output = "Connection successful.";
```

FormCalc

```
xfa.connectionSet.DataConnection.operation.output = "Connection successful."
```

overflowLeader

Specifies the subform to place at the top of the content area or page when it is entered as a result of an overflow.

Syntax

```
Reference_Syntax.overflowLeader = "string"
```

Values

Type	Values
String	A valid string representing the name or fully qualified reference syntax expression of a subform.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.overflowLeader = "Subform2";
```

FormCalc

```
Subform1.break.overflowLeader = "Subform2"
```

overflowTarget

Specifies the explicit content area that will be the transition target when the current content area or page area overflows.

Syntax

Reference_Syntax.overflowTarget = "*string*"

Values

Type	Values
String	The name or fully qualified reference syntax expression of a content area.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.overflowTarget = "xfa.form.form1.pageSet.Page1.Content_Main";
```

FormCalc

```
Subform1.break.overflowTarget = "xfa.form.form1.pageSet.Page1.Content_Main"
```

overflowTrailer

Specifies the subform to place at the bottom of the content area or page when it overflows.

The vertical space required for the overflow trailer must be reserved.

Syntax

Reference_Syntax.overflowTrailer = "*string*"

Values

Type	Values
String	A valid string representing the name or fully qualified reference syntax expression of a subform.

Applies to

Model	Object
Form Model	break

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.overflowTrailer = "Subform2";
```

FormCalc

```
Subform1.break.overflowTrailer = "Subform2"
```

overline

Specifies the activation and type of overlining.

Syntax

Reference_Syntax.overline = "0 | 1 | 2"

Values

Type	Values
String	• 0 (default) The font renders without overlining. • 1 The font renders with a single overline. • 2 The font renders with a double overline.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.overline = "0";
```

FormCalc

```
TextField1.font.overline = "0"
```

overlinePeriod

Controls the appearance of overlining.

Syntax
Reference_Syntax.overlinePeriod = "all | word"

Values

Type	Values
String	• `all` (default) The rendered line extends across word breaks. • `word` The rendered line is interrupted at word breaks.

Applies to

Model	Object
Form Model	font

Version
XFA 2.1

Examples

JavaScript
```
TextField1.font.overlinePeriod = "all";
```

FormCalc
```
TextField1.font.overlinePeriod = "all"
```

override

When used with the calculate object, the `override` property indicates whether the field allows overrides to occur and disables or enables calculations. When used with the value object, the `override` property indicates whether a calculation override has occurred.

When there is no accompanying calculate object, this property has no effect and the user can enter a value in the field.

Syntax
Reference_Syntax.override = "error | ignore | disabled | warning"

Values

Type	Values
String	• `error` The calculation is enabled and the user cannot override the calculated value. If the user tries to override the calculated value, the processing application displays an error message. To avoid the need for error messages, form designers can define these fields as read-only. This is the default override value if the `calculate` object is included in the container object. • `ignore` The calculated value is supplied as a default. If the user overrides the value, the processing application allows the override to occur without displaying any warning message to the user. This is the default override value if the `calculate` object is omitted from the container. • `disabled` The calculation is disabled. In an interactive context, the user can enter data in the field. The effect of this override value is independent of user action. The `disabled` value allows an event script to dynamically enable or disable a calculate object. • `warning` The calculation is enabled and the calculated value is recommended over user-input values. If the user overrides the calculated value, the processing application displays a warning message. The message informs the user that the form object should use a calculated value and provides the user with two choices: • `Dismiss` indicates that the user wants to use the calculated value. • `Override` indicates that the user understands the message, but chooses to override the calculated value. The application does not issue any warnings or prompts on subsequent focus gains by the same object.

Applies to

Model	Object
Form Model	calculate value

Version

XFA 2.1

Examples

JavaScript

```
TextField1.calculate.override = "disabled";
```

FormCalc

```
TextField1.calculate.override = "disabled"
```

pagePosition

Specifies a page's position within a set of pages.

Syntax

Reference_Syntax.pagePosition = "any | first | last | rest | only"

Values

Type	Values
String	• any (default) Matches any pages with a contiguous set of pages. • first Matches the first page within a contiguous sequence of pages. • last Matches the last page within a contiguous sequence of pages. • rest Matches any page that is both not the first or the last in a sequence of pages. • only Matches a single page sequence.

Applies to

Model	Object
Form Model	pageArea

Version

XFA 2.5

Examples

The reference syntax expression will vary, depending on the object from which it is invoked.

JavaScript

```
xfa.form.form1.pageSet.Page1.pagePosition = "only";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.pagePosition = "only"
```

parent

Returns the parent object of the current object.

Note: This property is read only.

Syntax

Reference_Syntax.parent

Values

Type	Values
Object	An XML Form Object Model object.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
TextField1.parent;
```

FormCalc

```
TextField1.parent
```

See also

- "Referencing objects" on page 420
- "Manipulating instances of a subform" on page 424
- "Changing the background color" on page 428

parentSubform

Specifies the parent subform (page) of this field.

Syntax

Reference_Syntax.parentSubform = "*string*"

Values

Type	Values
String	A valid string representing the name or fully qualified reference syntax expression of the parent subform object.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.parentSubform;
```

FormCalc
```
TextField1.parentSubform
```

passwordChar

Specifies the character the form displays for each password character a user enters.

Syntax
Reference_Syntax.passwordChar = "* | *character*"

Values

Type	Values
String	• "*" (asterisk) (default) • Any valid single character.

Applies to

Model	Object
Form Model	passwordEdit

Version

XFA 2.1

Examples

JavaScript
```
PasswordField1.resolveNode("ui.#passwordEdit").passwordChar = "*";
```

FormCalc
```
PasswordField1.ui.#passwordEdit.passwordChar = "*"
```

permissions

Specifies the access permissions granted for a form that includes an author signature.

For information about author signatures, see "signatureType" on page 296.

Syntax
Reference_Syntax.permissions = "1 | 2 | 3"

Values

Type	Values
String	• `1` No changes to the document are permitted. Any change to the document invalidates the signature. • `2` (default) The permitted changes are filling in forms, instantiating page templates, and signing. Other changes invalidate the signature. • `3` The permitted changes are those allowed by 2, as well as annotation creation, deletion, and modification. Other changes invalidate the signature.

Applies to

Model	Object
Form Model	mdp

Version

XFA 2.5

placement

Specifies the placement of the caption.

Syntax

Reference_Syntax.placement = "left | right | top | bottom | inline"

Values

Type	Values
String	• `left` (default) Locates the caption to the left of the content. • `right` Locates the caption to the right of the content. • `top` Locates the caption above the content. • `bottom` Locates the caption below of the content. • `inline` Locates the caption inline immediately before to the content.

Applies to

Model	Object
Form Model	caption

Version

XFA 2.1

Examples

JavaScript

```
TextField1.caption.placement = "left";
```

FormCalc

```
TextField1.caption.placement = "left"
```

platform

Returns the platform of the machine running the script.

Note: This property is read only.

Syntax

Reference_Syntax.platform

Values

Type	Values
String	A valid string representing the operating system. For example, in the case of a PDF form in Acrobat, this property returns one of: WIN, MAC, or UNIX.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.platform;
```

FormCalc

```
xfa.host.platform
```

posture

Specifies the posture of the font.

Syntax

Reference_Syntax.posture = "normal | italic"

Values

Type	Values
String	• `normal` (default) The font has a normal posture. • `italic` The font is italicized.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.posture = "italic";
```

FormCalc

```
TextField1.font.posture = "italic"
```

presence

Specifies an object's visibility.

Syntax

Reference_Syntax.presence = "visible | invisible | hidden | simplex | duplex"

Values

Type	Values
String	• `visible` (default) The object is visible. • `invisible` The object is transparent. Although invisible, the object still takes up space. • `hidden` The object is hidden. The form does not display the object and the object does not take up space on the form's layout. • `simplex` The object is printed only when using single-sided printing. • `duplex` The object is printed when using double-sided printing.

Applies to

Model	Object
Form Model	border caption corner draw edge exclGroup field fill items subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.presence = "hidden";
```

FormCalc

```
TextField1.presence = "hidden"
```

See also

"Making an object visible or invisible" on page 432

preserve

Specifies widow/orphan-style constraints on the overflow behavior of the content within the enclosing container.

Syntax

Reference_Syntax.preserve = "0 | *integer* | all"

Values

Type	Values
String	• 0 (default) The content is broken across an overflow boundary. • integer An integer value greater than zero specifies the minimum quantity of content that must transition across the overflow boundary. For instance, specifying an integer value of 2 would prevent a single line of content from being widowed across the overflow boundary; it would result in a minimum of two lines of content transitioning across the overflow boundary. • all Each paragraph of content must be kept intact and therefore cannot be broken across an overflow boundary.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.preserve = "all";
```

FormCalc

```
TextField1.para.preserve = "all"
```

prevContentType

Specifies the content type of the value specified for the prevText property.

For example, if prevContentType='text/html',prevText contains an XHTML fragment.

Syntax

Reference_Syntax.prevContentType = "allowRichText | plainTextOnly"

Values

Type	Values
String	• `allowRichText` (default) The field supports rich text. • `plainTextOnly` The field does not support rich text.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.prevContentType = "plainTextOnly";
```

FormCalc

```
xfa.event.prevContentType = "plainTextOnly"
```

previous

Specifies the constraints on keeping a subform together with the previous subform within a content area or page.

Syntax

Reference_Syntax`.previous = "none | contentArea | pageArea"`

Values

Type	Values
String	• `none` (default) The determination of whether a subform renders in the same content area or page together with the previous subform will be delegated to the processing application. No special constraints are forced. • `contentArea` The subform is requested to be rendered in the same content area with the previous subform. • `pageArea` The subform is requested to be rendered in the same page with the previous subform.

Applies to

Model	Object
Form Model	keep

Version

XFA 2.1

Examples

JavaScript

```
Subform1.keep.previous = "contentArea";
```

FormCalc

```
Subform1.keep.previous = "contentArea"
```

prevText

Specifies the content of the field before it changes in response to the actions of a user.

The `prevText` value can be recalled, similar to an undo feature.

Syntax

`Reference_Syntax``.prevText`

Values

Type	Values
String	A string up to 255 characters.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.prevText;
```

FormCalc

```
xfa.event.prevText
```

See also

- "Referencing objects" on page 420
- "Populating a drop-down list" on page 430

printCheckDigit

Specifies whether to print the check digits in the human-readable text.

The parser ignores this property if the checksum property has a value of `0`, or if the checksum property has a value of `1` and the standard behavior for the barcode type is to not include a checksum.

Syntax

```
Reference_Syntax.printCheckDigit = "0 | 1"
```

Values

Type	Values
String	• `0` (default) Do not print the check digit in the human-readable text, only in the barcode itself. • `1` Append the check digit to the end of the human-readable text.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").printCheckDigit = "1";
```

FormCalc

```
Code11BarCode1.ui.#barcode.printCheckDigit = "1"
```

priority

Alters the search path for text to speak. Whichever object is named in this property moves to the front of the search path. The other objects retain their relative order.

Syntax

```
Reference_Syntax.priority = "custom | caption | name | tooltip"
```

Values

Type	Values
String	• `custom` (default) The search order is `speak`, `tooltip`, `caption`, the container's name. • `caption` The search order is `caption`, `speak`, `tooltip`, the container's name. • `name` The search order is the container's name, `speak`, `tooltip`, `caption`. • `tooltip` The search order is `tooltip`, `speak`, `caption`, the container's name.

Applies to

Model	Object
Form Model	speak

Version

XFA 2.1

Examples

JavaScript

```
TextField1.assist.speak.priority = "tooltip";
```

FormCalc

```
TextField1.assist.speak.priority = "tooltip"
```

radius

Specifies the radius of the corner.

The `radius` property always influences the appearance of round corners, but will also determine the depth of an inverted square corner. Each edge is trimmed from its end points by the corner radius, regardless of the values of the inverted and join properties. In general, this is of no consequence, because the corner will visibly join with the edges at their trim points. However, if the corner specifies a presence if invisible, the trimming of the edges will become apparent, even when the corner is square and not inverted.

Syntax

Reference_Syntax`.radius = "0in | `*measurement*`"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	corner

Version

XFA 2.1

Examples

JavaScript

```
TextField1.border.corner.radius = "0.5in";
```

FormCalc

```
TextField1.border.corner.radius = "0.5in"
```

radixOffset

Specifies an offset value for the anchor of the paragraph.

Syntax

Reference_Syntax.radixOffset = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
NumericField1.para.radixOffset = "0in";
```

FormCalc

```
NumericField1.para.radixOffset = "0in"
```

rate

Specifies the percentage of stipple color that is stippled over a solid background color.

The background color is not specified by the stipple object.

Syntax
Reference_Syntax.rate = "50 | *integer*"

Values

Type	Values
String	• 50 (default) • Any valid integer value between 0 and 100, where 0 results in no visible stippling drawn over the background color and 100 results in a complete obscuring of the background color by filling the area completely with stipple color. Any stipple rate between 0 and 100 results in a varying blend of background color and an overlaid stipple color. For example, a stipple rate of 50 results in an equal blend of background color and stipple color.

Applies to

Model	Object
Form Model	stipple

Version

XFA 2.1

Examples

JavaScript
```
TextField1.border.fill.stipple.rate = "75";
```

FormCalc
```
TextField1.border.fill.stipple.rate = "75"
```

rawValue

Specifies the unformatted value of the current object.

For example, this property can return or set the value of a field.

Syntax
Reference_Syntax.rawValue = "*value*"

Values

Type	Values
Varies	Values differ depending on the referencing object. For example, for objects that require a color value, this property specifies a comma-separated list of values for each color component of the color space in the form r,g,b. Alternatively, the rawValue property of a field object is a string representing the actual value displayed in the field, or the field's bound value.

Applies to

Model	Object
Form Model	draw exclGroup field

Version

XFA 2.1

Examples

JavaScript

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

FormCalc

```
TextField1.rawValue = "Hello"
```

See also

- "Referencing objects" on page 420
- "Creating a node in the data model" on page 422
- "Getting or setting object values" on page 425
- "Working with page numbers and page counts" on page 426
- "Concatenating data values" on page 427
- "Calculating totals" on page 428
- "Populating a drop-down list" on page 430
- "Using radio buttons and check boxes" on page 433
- "Determining that a form has changed" on page 433

ready

Specifies whether the form layout process is complete and scripting tasks can begin.

Note: This property is read only.

Syntax

Reference_Syntax.ready = "True | False"

Values

Type	Values
Boolean	• True (default) Layout process is complete. • False Layout process is not complete.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.ready;
```

FormCalc

```
xfa.layout.ready
```

recordsAfter

Returns the number of records in the data window following the current record.

Note: This property is read only.

Syntax

Reference_Syntax.recordsAfter

Values

Type	Values
Integer	A valid integer value between 0 and the index value of the last record in the source data.

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript

```
xfa.dataWindow.recordsAfter;
```

FormCalc

```
xfa.dataWindow.recordsAfter
```

recordsBefore

Returns the number of records that are in the data window prior to the current record.

Note: This property is read only.

Syntax
Reference_Syntax.recordsBefore

Values

Type	Values
Integer	A valid integer value between 0 and the index value of the first record in the source data.

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript
```
xfa.dataWindow.recordsBefore;
```

FormCalc
```
xfa.dataWindow.recordsBefore
```

reenter

Specifies whether the enter event is occurring for the first time. The enter event occurs each time a user clicks in a field.

The first time a user clicks in a field, an enter event is sent with the reenter property set to false. If the user clicks in the field again or presses the Enter key, another enter event is sent with the reenter property set to true.

Syntax
Reference_Syntax.reenter = "0 | 1"

Values

Type	Values
Boolean	• True The enter event has already occurred. • False The enter event occurs for the first time.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.reenter = "False";
```

FormCalc

```
xfa.event.reenter = "False"
```

ref

Specifies a reference syntax expression defining the node in the data model to which the enclosing container will bind.

Syntax

Reference_Syntax.ref = "*string*"

Values

Type	Values
String	A valid reference syntax expression.

Applies to

Model	Object
Form Model	bind bindItems connect event items signData traverse
sourceSet Model	bind connect

Version

XFA 2.1

relation

Specifies the relationship among the members of the set.

Syntax

Reference_Syntax.relation = "ordered | unordered | choice"

Values

Type	Values
String	• `ordered` (default) Instantiates members in the order in which they are declared in the form design. This has the effect of potentially re-ordering the content to satisfy the document order of the form design. • `unordered` Instantiates the members in data order regardless of the order in which they are declared. This has the effect of potentially re-ordering the set to satisfy the ordering of the content. • `choice` The members are exclusive of each other, and only one member may be instantiated. The determination of which member to instantiate is based upon the data.

Applies to

Model	Object
Form Model	subformSet

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.resolveNode("#subformSet").relation = "unordered";
```

FormCalc

```
xfa.form.form1.#subformSet.relation = "unordered"
```

relevant

Controls whether a form object is included when the form is printed.

Syntax

Reference_Syntax`.relevant = "+print | -print"`

Values

Type	Values
String	• `+print	print` (default for visible objects) Forces a particular object to appear when the form is printed, regardless of the object's presence property setting. • `-print` (default for invisible or hidden objects) Forces an object not to appear when the form is printed, regardless of the object's presence property setting.

Applies to

Model	Object
Form Model	area border contentArea draw exclGroup field pageArea pageSet subform subformSet value

Version

XFA 2.1

Examples

JavaScript

```
Button1.relevant = "-print";
```

FormCalc

```
Button1.relevant = "-print"
```

See also

"Making an object visible or invisible" on page 432

reserve

A measurement value that specifies the height or width of the caption.

The effect of this property is determined by the placement property. When the caption is placed at the left or right, the `reserve` property specifies the height of the caption region. When the caption is placed at the top or bottom, the `reserve` property specifies the width. When the caption is placed inline, the `reserve` property is ignored.

A reserve of 0 sets the caption area to auto-fit. It adjusts the size of the object to fit the caption.

Syntax

`Reference_Syntax``.reserve = "`*`measurement`*`"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	caption

Version

XFA 2.1

Examples

JavaScript

```
TextField1.caption.reserve = "1.5in";
```

FormCalc

```
TextField1.caption.reserve = "1.5in"
```

restoreState

Restores the form nodes of a form to their original state, including resetting the visual properties of fields such as changes to border colors.

Syntax

Reference_Syntax.restoreState = "none | manual | auto"

Values

Type	Values
String	• **`none` (default)** The state and restore information are not saved. • **`manual`** Specific properties are saved and restored using script objects. The state is restored using the delta script object. If the root subform uses this value, the following properties are saved and restored: • The checksum is verified. • All state information is restored using the restore method on the delta script object only if the checksum was valid. • Field values and calculation overrides are restored if the checksum was valid. • **`auto` (default for new forms)** Automatically saves and restores the form to its original state. When opening a certified form, the state will not be restored. On an uncertified form, certification of the document will not be allowed. **Note:** The `auto` setting can not be used for certified documents. If the root subform uses this value, the following properties and saved and restored: • The checksum is verified • After the merge step is complete but prior to calculations being executed, each form node will have its state restored using the saved form model only if the checksum was valid. • The restore method on the delta script object does nothing. • Field values and calculation overrides are restored if the checksum was valid.

Applies to

Model	Object
Form Model	subform

Version

XFA 2.5

Examples

JavaScript

```
Subform1.restoreState = "auto";
```

FormCalc

```
Subform1.restoreState = "auto"
```

rightInset

Specifies the size of the right inset.

Syntax

Reference_Syntax.rightInset = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	margin

Version

XFA 2.1

Examples

JavaScript

```
Subform1.margin.rightInset = "0.25in";
```

FormCalc

```
Subform1.margin.rightInset = "0.25in"
```

role

Specifies the role played by the parent container.

Syntax

Reference_Syntax.role = "*string*"

Values

Type	Values
String	A valid string specifying the role of the parent container. It may be used by speech-enabled XFA processing applications to provide information. For example, it may be assigned values borrowed from HTML, such as `TH` (table headings) and `TR` (table rows).

Applies to

Model	Object
Form Model	assist

Version

XFA 2.2

Examples

JavaScript

```
TextField1.assist.role = "TH";
```

FormCalc

```
TextField1.assist.role = "TH"
```

rotate

Rotates the object around its anchor point by the specified angle.

The angle represents degrees counter-clockwise with respect to the default position. The value must be a non-negative multiple of 90.

Note: The direction of rotation is the same as for positive angles in PostScript®, PDF, and PCL but opposite to that in SVG.

Syntax

Reference_Syntax`.rotate = "0 | `*measurement*`"`

Values

Type	Values
String	• `0` (default) • Any valid angle measurement.

Applies to

Model	Object
Form Model	draw field

Version

XFA 2.1

Examples

JavaScript

```
TextField1.rotate = "90";
```

FormCalc

```
TextField1.rotate = "90"
```

rowColumnRatio

An optional ratio of rows to columns for supported 2D barcodes.

The parser ignores this property if dataRowCount and dataColumnCount properties are specified.

When `rowColumnRatio` is supplied, the barcode grows to the number of rows required to hold the supplied data. If the last row is not filled by the supplied data it is padded out with padding symbols.

Syntax

Reference_Syntax.rowColumnRatio = "*string*"

Values

Type	Values
String	A valid string representing the ratio of rows to columns.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

runAt

Specifies what application can execute the script.

This setting is enforced even if the script is called by another script.

Syntax

Reference_Syntax.runAt = "client | server | both"

Values

Type	Values
String	• `client` (default) The script runs only on the client. • `server` The script runs only on the server. • `both` The script runs on both client and server.

Applies to

Model	Object
Form Model	execute script

Version

XFA 2.1

Examples

JavaScript

```
NumericField1.calculate.script.runAt = "both";
```

FormCalc

```
NumericField1.calculate.script.runAt = "both"
```

save

Determines whether the values in a particular column represent both display and bound values, or if the data in the column represents bound values only.

Syntax

Reference_Syntax`.save = "0 | 1"`

Values

Type	Values
String	• `0` (default) The values supplied by this object are for display only. • `1` The values supplied by this object may be entered into the field. At least one column must have a value of 1. If multiple columns have a value set to 1, then the parser saves the first column first column with a value of 1 that is encountered.

Applies to

Model	Object
Form Model	items

Version

XFA 2.1

Examples

JavaScript

```
DropDownList1.resolveNode("#items").save = "1";
```

FormCalc

```
DropDownList1.#items.save = "1"
```

savedValue

Returns a typed object, but you cannot assign this value. If the property is not saved the value is the same as the currentValue.

Syntax

Reference_Syntax.savedValue = "*typed object*"

Values

Type	Values
Depends on the type of the property	The typed object for the property.

Applies to

Model	Object
Form Model	delta

Version

XFA 2.5

scope

Controls participation of the subform in data binding and reference syntax expressions. It is valid only on the root subform.

By default, a named subform takes part in data binding and can be referenced using a reference syntax expression. This property allows a subform to be given a name but remain transparent to data binding and reference syntax expressions.

Syntax

Reference_Syntax.scope = "name | none"

Values

Type	Values
String	• `name` (default) If the subform has a name it takes part in data binding and reference syntax expressions. Otherwise it does not. • `none` The subform does not take part in data binding and reference syntax expressions, even if it has a name.

Applies to

Model	Object
Form Model	subform

Version

XFA 2.1

Examples

JavaScript

```
Subform1.scope = "none";
```

FormCalc

```
Subform1.scope = "none"
```

scriptTest

Controls validation by the enclosed script.

Scripts specified as part of a validation should make no assumptions as to how the processing application might use the validation results, or when the `validate` object is invoked. In particular, the script should not attempt to provide feedback to a user or alter the state of the form in any way. For more information about validations, see "Validation" on page 40.

Syntax

`Reference_Syntax``.scriptTest = "error | disabled | warning"`

Values

Type	Values
String	• `disabled` Do not perform this test. The form object is permitted to have a value that does not conform to the script. The field can be left with a non-conforming value, and it will not negatively affect the validity of the form. This value disables the validation test. • `error` (default) Emit a message and refuse to accept data that the script reports is erroneous. The form object is required to have a value that conforms to the script. • `warning` Emit a message if the script reports the data is erroneous but allow the user to proceed to the next field. The message must inform the user that the form object is recommended to have a value that conforms to the script's constraints, and provide two choices: • `dismiss`: The user understands the form's recommendation and wishes to return to the form and satisfy this constraint. • `override`: The user understands the form's recommendation, but has chosen to contravene this constraint.

Applies to

Model	Object
Form Model	validate

Version

XFA 2.1

Examples

JavaScript

```
NumericField1.validate.scriptTest = "disabled";
```

FormCalc

```
NumericField1.validate.scriptTest = "disabled"
```

selectedIndex

The index of the first selected item.

Setting this property sets the specified index and deselects any previously selected items. If you want to preserve the multiple selection state, use the getItemState or setItemState methods instead. Specifying an index value of `-1` clears the list. Getting this property returns a value of `-1` when no items are selected.

Syntax

Reference_Syntax.selectedIndex

Values

Type	Values
Integer	A valid integer representing the index value of the first selected item. Specifying an index value of `-1` clears the list. Specifying any other valid value results in only that item being selected.

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

selEnd

Specifies the index position of the last character of the text selection stored in the prevText property during a change event.

Syntax

Reference_Syntax`.selEnd`

Values

Type	Values
Integer	A valid integer representing the 0 based index value of the last character of the text selection. If no text is selected, this property is set to the position of the text entry cursor at the time the change is made. Changing the value of this property changes which characters will be replaced by the value of change and also repositions the text entry cursor.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.selEnd;
```

FormCalc

```
xfa.event.selEnd
```

selStart

Specifies the index position of the first character of the text selection stored in the prevText property during a change event.

Syntax

Reference_Syntax.selStart

Values

Type	Values
Integer	A valid integer representing the 0-based index value of the first character of the text selection. If no text is selected, this property is set to the position of the text entry cursor at the time the change is made. Changing the value of this property changes which characters will be replaced by the value of change and also repositions the text entry cursor.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.selStart;
```

FormCalc

```
xfa.event.selStart
```

server

Specifies the URL for a time stamp server.

Syntax

Reference_Syntax.server = "*string*"

Values

Type	Values
String	A valid string representing the URL for the time stamp server.

Applies to

Model	Object
Form Model	timeStamp

Version

XFA 2.5

shape

Specifies whether the check box or radio button displays with a square or round outline.

Syntax

Reference_Syntax.shape = "square | round"

Values

Type	Values
String	• square (default) The button appears with a square outline. • round The button appears with a round outline.

Applies to

Model	Object
Form Model	checkButton

Version

XFA 2.1

Examples

JavaScript

```
CheckButton1.resolveNode("ui.#checkButton").shape = "square";
```

FormCalc

```
CheckButton.ui.#checkButton.shape = "square"
```

shift

Specifies whether the Shift key is held down during a particular event.

Syntax

Reference_Syntax.shift

Values

Type	Values
Boolean	• `True` (default) The Shift key is pressed during event execution. • `False` The Shift key is not pressed during event execution.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.shift;
```

FormCalc

```
xfa.event.shift
```

short

Specifies the length of the short edge of the `medium` object.

The length specified by the short property must be smaller than the length specified by the long property.

Syntax

`Reference_Syntax``.short = "0in | `*`measurement`*`"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	medium

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.medium.short;
```

FormCalc

```
xfa.form.form1.pageSet.Page1.medium.short
```

signatureType

Specifies how a form with a document signature is saved as certified PDF document.

Syntax

Reference_Syntax`.signatureType = "filler | author"`

Values

Type	Values
String	• `filler` (default) Saves the form as a certified PDF document. • `author` Documents with author signatures are referred to as certified. After the form is saved as a PDF document and opened in Acrobat, the user can click the document signature field to certify the entire document.

Applies to

Model	Object
Form Model	mdp

Version

XFA 2.5

size

A measurement specifying the size of the check box or radio button outline representing either the height and width for a check box, or the diameter for a radio button.

Syntax

Reference_Syntax`.size = "10pt | `*measurement*`"`

Values

Type	Values
String	• `10pt` (default) • Any valid measurement. The values for this property depend on the referencing object: • For the `font` object, this property specifies the size of the font. • For the `checkButton` object, this property specifies either the height or width of a check box or the diameter of a radio button.

Applies to

Model	Object
Form Model	checkButton font

Version

XFA 2.1

Examples

JavaScript

```
CheckBox1.resolveNode("ui.#checkButton").size = "20pt";
```

FormCalc

```
CheckBox1.ui.#checkButton.size = "20pt"
```

slope

Specifies the orientation of the line.

Syntax

Reference_Syntax`.slope = "\ | /"`

Values

Type	Values
String	• `\` (backslash character) (default) The line extends from the top-left to the bottom-right. • `/` (forward slash character) The line extends from the bottom-left to the top-right.

Applies to

Model	Object
Form Model	line

Version

XFA 2.1

Examples

JavaScript

```
Line1.resolveNode("value.#line").slope = "/";
```

FormCalc

```
Line1.value.#line.slope = "/"
```

soapFaultCode

Specifies any fault code that occurs when a user attempts to execute a web service connection.

Syntax

Reference_Syntax.soapFaultCode = "*string*"

Values

Type	Values
String	A valid string representing the SOAP fault code.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

soapFaultString

Specifies the descriptive message that corresponds to a particular web service connection fault code.

Syntax

Reference_Syntax.size = "10pt | *measurement*"

Values

Type	Values
String	A valid string representing the SOAP fault code message.

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

somExpression

Reads the reference syntax expression for this node.

Syntax

Reference_Syntax.somExpression

Values

Type	Values
String	A valid string representing a fully qualified reference syntax expression.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
TextField1.somExpression;
```

FormCalc

```
TextField1.somExpression
```

spaceAbove

Specifies the amount of vertical spacing and the maximum font leading for the first line of the paragraph.

Syntax

Reference_Syntax.spaceAbove = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.spaceAbove = "2pt";
```

FormCalc

```
TextField1.para.spaceAbove = "2pt"
```

spaceBelow

Specifies the amount of vertical spacing and the maximum font leading for the first line of the paragraph.

Syntax

Reference_Syntax.spaceAbove = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.spaceBelow = "2pt";
```

FormCalc

```
TextField1.para.spaceBelow = "2pt"
```

startAngle

Specifies the angle where the beginning of the arc renders.

Syntax

Reference_Syntax.startAngle = "0 | *angle*"

Values

Type	Values
String	• `0` (default) • A value greater than `0` and less than or equal to `360`.

Applies to

Model	Object
Form Model	arc

Version

XFA 2.1

Examples

JavaScript

```
Circle1.resolveNode("value.#arc").startAngle = "12";
```

FormCalc

```
Circle1.value.#arc.startAngle = "12"
```

startChar

Specifies an optional starting control character to add to the beginning of the barcode data.

The `starChar` property is ignored by the parser if the barcode pattern does not support the specified starting control character.

Syntax

`Reference_Syntax``.startChar = "`*`character`*`"`

Values

Type	Values
String	A valid string representing a control character.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").startChar = "*";
```

FormCalc

```
Code11BarCode1.ui.#barcode.startChar = "*"
```

startNew

Determines whether it is necessary to start a new content area or page even when the current content area or page has the required name.

This property has no effect unless the before property has the value `contentArea` or `pageArea`.

Syntax

```
Reference_Syntax.startNew = "0 | 1"
```

Values

Type	Values
String	• `0` (default) Does not start a new content area or page area if the current one has the specified name. • `1` Starts a new content area or page. The name of the content area or page is supplied by the accompanying beforeTarget property.

Applies to

Model	Object
Form Model	break breakAfter breakBefore

Version

XFA 2.1

Examples

JavaScript

```
Subform1.break.startNew = "1";
```

FormCalc

```
Subform1.break.startNew = "1"
```

stateless

Determines whether a script's variables persist from one invocation to the next.

Syntax

```
Reference_Syntax.stateless = "0 | 1"
```

Values

Type	Values
String	• `0` (default) The script's variables do persist (it is stateful). • `1` The script's variables do not persist (it is stateless).

Applies to

Model	Object
Form Model	script

Version

XFA 2.1

Examples

JavaScript

```
TextField1.resolveNode("#event.#script").stateless = "1";
```

FormCalc

```
TextField1.#event.#script.stateless = "1"
```

stock

Specifies the name of a standard paper size.

Syntax

Reference_Syntax`.stock = "letter | `*paper_size*`"`

Values

Type	Values
String	• `letter` (default) • Any valid paper size value.

Applies to

Model	Object
Form Model	medium

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.form1.pageSet.Page1.medium.stock = "A4";
```

FormCalc

```
xfa.form.form1.pageSet.Page1.medium.stock = "A4"
```

stroke

Specifies the appearance of a line.

Syntax

```
Reference_Syntax.stroke = "solid | dashed | dotted | dashDot | dashDotDot
| lowered | raised | etched | embossed"
```

Values

Type	Values
String	• `solid` (default) Solid. • `dashed` A series of rectangular dashes. • `dotted` A series of round dots. • `dashDot` Alternating rectangular dashes and dots. • `dashDotDot` A series of a single rectangular dash followed by two round dots. • `lowered` The line appears to enclose a lowered region. • `raised` The line appears to enclose a raised region. • `etched` The line appears to be a groove lowered into the drawing surface. • `embossed` The line appears to be a ridge raised out of the drawing surface.

Applies to

Model	Object
Form Model	corner edge

Version

XFA 2.1

Examples

JavaScript

```
Line1.resolveNode("value.#line.edge").stroke = "etched";
```

FormCalc

```
Line1.value.#line.edge.stroke = "etched"
```

sweepAngle

Specifies the length of the arc as an angle.

Syntax

Reference_Syntax.sweepAngle = "360 |*angle*"

Values

Type	Values
String	• 360 (default) • A value less than 360 and greater than or equal to 0.

Applies to

Model	Object
Form Model	arc

Version

XFA 2.1

Examples

JavaScript

```
Circle1.resolveNode("value.#arc").sweepAngle = "45";
```

FormCalc

```
Circle1.value.#arc.sweepAngle = "45"
```

tabDefault

Specifies the distance between default tab stops.

By default, no default tab stops are defined.

Syntax

Reference_Syntax.tabDefault = "*string*"

Values

Type	Values
String	A valid string representing the distance between the default tab stops.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.tabDefault = "3in";
```

FormCalc

```
TextField1.para.tabDefault = "3in"
```

tabStops

Specifies a space-separated list of tab stop locations.

Within the region from the left margin to the rightmost tab stop in the list, these tab stop locations replace the default tab stops specified by the tabDefault property. The default tab stops resume to the right of this region.

Each entry in the list of tab stops consists of a keyword specifying the alignment at the tab stop, followed by a space, followed by the distance of the tab stop from the left margin.

Syntax

Reference_Syntax.tabStops = "center | left | rigth | decimal"

Values

Type	Values
String	The tab stop alignment is one of the following values: • `center` Specifies a center-aligned tab stop. • `left` Specifies a left-aligned tab stop. • `right` Specifies a right-aligned tab stop. • `decimal` Specifies a tab-stop that aligns content around a radix point.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.tabStops = "decimal";
```

FormCalc

```
TextField1.para.tabStops = "decimal"
```

target

Specifies the object upon which the event is acting.

Syntax

Reference_Syntax.target = "*URL*"

Values

Type	Values
String	A valid string representing the name of a form design object, a reference syntax expression, or a URL where data is sent.

Applies to

Model	Object
Event Model	eventPseudoModel
Form Model	breakAfter breakBefore overflow setProperty signData submit

Version

XFA 2.4

Examples

JavaScript

```
xfa.event.target;
```

FormCalc

```
xfa.event.target
```

See also

- "Referencing objects" on page 420
- "Saving a form" on page 431

targetType

Specifies the constraints on moving to a new page or content area before laying out the parent subform.

The targetType property replaces the deprecated `break.before` property.

Syntax

`Reference_Syntax``.targetType = "auto | contentArea | pageArea"`

Values

Type	Values
String	The startNew property also modifies some of these behaviors. • `auto` (default) The determination of a transition to a new page or content area is delegated to the processing application. No transition to a new page or content area is forced. • `contentArea` Rendering transitions to the next available content area. See also the startNew property. • `pageArea` Rendering transitions to a new page. See also the startNew property.

Applies to

Model	Object
Form Model	breakAfter breakBefore

Version

XFA 2.4

Examples

JavaScript

```
Subform1.breakBefore.targetType = "pageArea";
```

FormCalc

```
Subform1.breakBefore.targetType = "pageArea"
```

textEncoding

Specifies the encoding of text content in the document.

Syntax

Reference_Syntax.textEncoding = "UTF-8 | UTF-16 | Shift-JIS | Big-Five | GB-2312"

Values

Note: The value of this property is case-insensitive and must match one of the following values.

Type	Values
String	• none (default) No special encoding is specified. The characters are encoded using the ambient encoding for the operating system. • ISO-8859-1 The characters are encoded using ISO-8859-1, also known as Latin-1. • ISO-8859-2 The characters are encoded using ISO-8859-2. I • SO-8859-7 The characters are encoded using ISO-8859-7. • Shift-JIS The characters are encoded using JIS X 0208, more commonly known as Shift-JIS. • KSC-5601 The characters are encoded using the Code for Information Interchange (Hangul and Hanja). • Big-Five The characters are encoded using Traditional Chinese (Big-Five). There is no official standard for Big-Five and several variants are in use. The Adobe form object model uses the variant implemented by Microsoft as code. • GB-2312 The characters are encoded using Simplified Chinese. • UTF-8 The characters are encoded using Unicode code points as defined by Unicode, and UTF-8 serialization as defined by ISO/IEC 10646. • UTF-16 The characters are encoded using Unicode code points as defined by Unicode, and UTF-16 serialization as defined by ISO/IEC 10646.

Type	Values
	• `UCS-2` The characters are encoded using Unicode code points as defined by Unicode, and UCS-2 serialization as defined by ISO/IEC 10646. • `fontSpecific` The characters are encoded in a font-specific way. Each character is represented by one 8-bit byte.

Applies to

Model	Object
Form Model	submit

Version

XFA 2.1

Examples

JavaScript

```
Button1.event.submit.textEncoding = "UCS-2";
```

FormCalc

```
Button1.event.submit.textEncoding = "UCS-2"
```

textEntry

Determines if a user can type a value into a drop-down list.

Syntax

Reference_Syntax.textEntry = "0 | 1"

Values

Type	Values
String	• `0` (default) Prevents the user from typing in the current field. The value is chosen by selecting a value from the drop-down list. • `1` Allows a user to type a value into a drop-down list or select from the drop-down list. This opens up the field value to be anything that the user might type. If the `open` property is set to `multiSelect`, the user is not allowed to enter values in the field.

Applies to

Model	Object
Form Model	choiceList

Version

XFA 2.1

Examples

JavaScript

```
DropDownList1.resolveNode("ui.#choiceList").textEntry = "1";
```

FormCalc

```
DropDownList1.ui.#choiceList.textEntry = "1"
```

textIndent

Specifies the horizontal positioning of the first line relative to the remaining lines in a paragraph.

A negative value indicates a hanging indent whereas a positive value indicates a first-line indent.

Syntax

Reference_Syntax.textIndent = "0in | *measurement*"

Values

Type	Values
String	• `0in` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	para

Version

XFA 2.1

Examples

JavaScript

```
TextField1.para.textIndent = "3in";
```

FormCalc

```
TextField1.para.textIndent = "3in"
```

textLocation

Specifies the location of any text associated with the barcode.

The region available for embedded text, if any, is determined by the barcode format. For most barcode formats it is a single, contiguous region. However, for EAN series barcodes, it is divided into four regions that inherit the typeface property and size property from the enclosing field. The form design must specify a typeface property and size property for the field that will fit into the provided space without overlapping any bars. The typeface property should be non-proportional.

Syntax

```
Reference_Syntax.textLocation = "below | none | above | aboveEmbedded
| belowEmbedded"
```

Values

Type	Values
String	• `below` (default) Places text below the barcode. • `above` Places text above the barcode. • `belowEmbedded` Partially embeds text at the bottom of the barcode aligned with the bottom of the bars. • `aboveEmbedded` Partially embeds text at the top of the barcode aligned with the top of the bars. • `none` Displays no text.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").textLocation = "aboveEmbedded";
```

FormCalc

```
Code11BarCode1.ui.#barcode.textLocation = "aboveEmbedded"
```

thickness

Specifies the thickness or weight of the line.

Syntax

```
Reference_Syntax.thickness = "0.5pt | measurement"
```

Values

Type	Values
String	• `0.5pt` (default) • Any valid measurement.

Applies to

Model	Object
Form Model	corner edge

Version

XFA 2.1

Examples

JavaScript

```
TextField1.border.edge.thickness = "0.2in";
```

FormCalc

```
TextField1.border.edge.thickness = "0.2in"
```

this

Retrieves the current node, which is the starting node when using the resolveNode and resolveNodes methods.

Note: This property is read only.

Syntax

```
this
```

Values

Type	Values
Object	The current object.

Applies to

Model	Object
XFA Model	xfa

Version

XFA 2.1

Examples

JavaScript

```
this
```

FormCalc
```
this
```

See also

- "Referencing objects" on page 420
- "Working with page numbers and page counts" on page 426
- "Changing the background color" on page 428

timeout

Specifies the number of seconds to attempt a query.

Syntax
Reference_Syntax.timeout = "*string*"

Values

Type	Values
String	A valid string representing the number of seconds before the query times out.

Applies to

Model	Object
sourceSet Model	command connect

Version

XFA 2.1

Examples

In these examples, `Titles` represents the data connection name.

JavaScript
```
xfa.sourceSet.Titles.connect.timeout = "10";
```

FormCalc
```
xfa.sourceSet.Titles.connect.timeout = "10"
```

timeStamp

Specifies the date/time stamp for this node.

Syntax
Reference_Syntax.timeStamp = "*string*"

Values

Type	Values
String	A valid string representing a date and time.

Applies to

Model	Object
XFA Model	xfa

Version

XFA 2.1

title

Sets and gets the title of the document. It is available only for client applications.

Syntax

Reference_Syntax.title

Values

Type	Values
String	A valid string representing the title of the current form.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.title;
```

FormCalc

```
xfa.host.title
```

topInset

A measurement specifying the size of the top inset.

Syntax

Reference_Syntax.topInset = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	margin

Version

XFA 2.1

Examples

JavaScript

```
Subform1.margin.topInset "1in";
```

FormCalc

```
Subform1.margin.topInset "1in"
```

trailer

Specifies the `subform` or `subformSet` object to place at the bottom of a content or page area.

The trailer property replaces the deprecated overflowTrailer and bookendTrailer properties.

Syntax

`Reference_Syntax``.trailer = "`*`string`*`"`

Values

Type	Values
String	A valid string representing the ID or fully qualified reference syntax expression of a subform or subform set. The default is an empty string.

Applies to

Model	Object
Form Model	bookend breakAfter breakBefore overflow

Version

XFA 2.4

Examples

JavaScript

```
Subform1.breakBefore.trailer = "Subform2";
```

FormCalc

```
Subform1.breakBefore.trailer = "Subform2"
```

transferEncoding

Specifies the encoding of binary content in the referenced document.

Syntax

Reference_Syntax.transferEncoding = "none | base64"

Values

Type	Values
String	• none (default) The referenced document is not encoded. If the referenced document is specified via a URI then it will be transferred as a byte stream. If the referenced document is inline it must conform to the restrictions on the PCDATA data type. • base64 The binary content is encoded in accordance with the base64 transfer encoding standard.

Applies to

Model	Object
Form Model	exData image
sourceSet Model	bind

Version

XFA 2.1

transient

Specifies whether the processing application must save the value of the exclusion group as part of a form submission or save operation.

Syntax

Reference_Syntax.transient = "0 | 1"

Values

Type	Values
String	• 0 (default) The exclusion group value must be saved. • 1 The exclusion group must not be saved.

Applies to

Model	Object
Form Model	exclGroup

Version

XFA 2.1

truncate

Truncates the right edge of the barcode for supported formats.

The truncation applies only to barcode type PDF417. The parser ignores this property for barcode formats to which it does not apply.

Syntax

Reference_Syntax.truncate = "0 | 1"

Values

Type	Values
String	• 0 (default) Include the right-hand synchronization mark. • 1 Omit the right-hand synchronization mark.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Code11BarCode1.resolveNode("ui.#barcode").truncate = "1";
```

FormCalc

```
Code11BarCode1.ui.#barcode.truncate = "1"
```

type

Specifies the pattern used by an object.

For the radial object, the type property specifies the direction of flow for a color transition.

For the subjectDNs object, the `type` property specifies whether the values provided in the element should be treated as a restrictive or a non-restrictive set.

Syntax

`Reference_Syntax``.type = "toRight | toLeft | toTop | toBottom"`

Values

Type	Values
String	The values for this property depend on the referencing object.
	For the `barcode` object: A string that identifies the barcode pattern. This property must be supplied. The set of supported values for this property is specific to the display device. The following values have been defined for this property as indicating particular barcode types: • `codabar` Codabar, as defined in ANSI/AIM BC3-1995, USS Codabar. • `code2Of5Industrial` Code 2 of 5 Industrial; no official standard. • `code2Of5Interleaved` Code 2 of 5 Interleaved, as defined in ANSI/AIM BC2-1995, USS Interleaved 2-of-5. • `code2Of5Matrix` Code 2 of 5 Matrix; no official standard. • `code2Of5Standard` Code 2 of 5 Standard; no official standard. • `code3Of9` Code 39 (also known as code 3 of 9), as defined in ANSI/AIM BC1-1995, USS Code 39. • `code3Of9extended` Code 39 extended; no official standard. • `code11` Code 11 (USD-8); no official standard. • `code49` Code 49, as defined in ANSI/AIM BC6-1995, USS Code 49. • `code93` Code 93, as defined in ANSI/AIM BC5-1995, USS Code 93. • `code128` Code 128, as defined in ANSI/AIM BC4-1995, Code 128. • `code128A` Code 128 A, as defined in ANSI/AIM BC4-1995, ISS Code 128. • `code128B` Code 128 B, as defined in ANSI/AIM BC4-1995, ISS Code 128. • `code128C` Code 128 C, as defined in ANSI/AIM BC4-1995, ISS Code 128. • `code128SSCC` Code 128 serial shipping container code, as defined in ANSI/AIM BC4-1995, ISS Code 128.

Type	Values
	• `ean8` EAN-8, as defined in ISO/EEC 15420. • `ean8add2` EAN-8 with 2-digit Addendum, as defined in ISO/EEC 15420. • `ean8add5` EAN-8 with 5-digit Addendum, as defined in ISO/EEC 15420. • `ean13` EAN-13, as defined in ISO/EEC 15420. • `ean13pwcd` EAN-13 with price/weight customer data. • `ean13add2` EAN-13 with a 2-digit addendum. • `ean13add5` EAN-13 with a 5-digit addendum. • `fim` United States Postal Service facing identification mark (FIM), as defined in First-Class Mail (USPS-C100). • `logmars` Logistics Applications of Automated Marking and Reading Symbols (logmars) as definied by United States Military Standard MIL-STD-1189B . • `maxicode` UPS Maxicode, as defined in ANSI/AIM BC10-ISS Maxicode. • `msi` Modified Plessey (MSI). May once have had a formal specification, but no longer does. • `pdf417` PDF417, as defined in USS PDF417. • `pdf417macro` PDF417, but allows the data to span multiple PDF417 barcodes. The barcodes are marked so that the bacrode reader knows when it still has additional barcodes to read and can prompt the operator if necessary. • `plessey` Plessey; no official standard. • `postAUSCust2` Australian Postal Customer 2, as defined in Customer Barcoding Technical Specifications. • `postAUSCust3` Australian Postal Customer 3, as defined in Customer Barcoding Technical Specifications. • `postAUSReplyPaid` Australian Postal Reply Paid, as defined in Customer Barcoding Technical Specifications. • `postAUSStandard` Australian Postal Standard, as defined in Customer Barcoding Technical Specifications. • `postUKRM4SCC` United Kingdom RM4SCC (Royal Mail 4-State Customer Code), as defined in the How to Use Mailsort Guide.

Type	Values
	• `postUSDPBC` United States Postal Service Delivery Point barcode, as defined in DMM C840 Barcoding Standards for Letters and Flats. • `postUSStandard` United States Postal Service POSTNET barcode (Zip+4), as defined in DMM C840 Barcoding Standards for Letters and Flats. • `postUSZip` United States Postal Service POSTNET barcode (5 digit Zip), as defined in DMM C840 Barcoding Standards for Letters and Flats. • `qr` QR Code, as defined in ISS - QR Code. • `telepen` Telepen, as defined in USS Telepen. • `ucc128` UCC/EAN 128, as defined in International Symbology Specification - Code 128 (1999). • `ucc128random` UCC/EAN 128 Random Weight, as defined in International Symbology Specification - Code 128 (1999). • `ucc128sscc` UCC/EAN 128 serial shipping container code (SSCC), as defined in International Symbology Specification - Code 128 (1999). • `upcA` UPC-A, as defined in ISO/EEC 15420. • `upcAadd2` UPC-A with 2-digit Addendum, as defined in ISO/EEC 15420. • `upcAadd5` UPC-A with 5-digit Addendum, as defined in ISO/EEC 15420. • `upcApwcd` UPC-A with Price/Weight customer data, as defined in ISO/EEC 15420. • `upcE` UPC-E, as defined in ISO/EEC 15420. • `upcEadd2` UPC-E with 2-digit Addendum, as defined in ISO/EEC 15420. • `upcEadd5` UPC-E with 5-digit Addendum, as defined in ISO/EEC 15420. • `upcean2` UPC/EAN with 2-digit Addendum, as defined in ISO/EEC 15420. • `upcean5` UPC/EAN with 5-digit Addendum, as defined in ISO/EEC 15420.

Type	Values
	For the `digestMethods`, `encodings`, subjectDNs, and `timeStamp` objects: Specifies whether the signing options are restricted to the filter settings. • `optional` (default) The signing options are not restricted to the filter settings. The values provided in the element are optional seed values that the XFA processing application may choose from. The XFA processing application may also supply its own value. • `required` The signing options are restricted to the filter settings. The values provided in the element are seed values that the XFA processing application must choose from.
	For the `linear` object: Specifies the direction of flow for a color transition. • `toRight` (default) The start color appears at the left side of the object and transitions into the end color at the right side. • `toLeft` The start color appears at the right side of the object and transitions into the end color at the left side. • `toTop` The start color appears at the bottom side of the object and transitions into the end color at the top side. • `toBottom` The start color appears at the top side of the object and transitions into the end color at the bottom side.
	For the `radial` object: Specifies the direction of the color transition. • `toEdge` (default) The start color appears at the center of the object and transitions into the end color at the outer edge. • `toCenter` The start color appears at the outer edge of the object and transitions into the end color at the center.

Applies to

Model	Object	
Form Model	barcode handler issuers linear oids	pattern radial reasons signing subjectDNs timeStamp
sourceSet Model	extras	

Version

XFA 2.1

typeface

Specifies the name of the typeface.

Syntax

Reference_Syntax.typeface = "Courier | *typeface*"

Values

Type	Values
String	• `Courier` (default) • Any valid typeface identifier.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.typeface = "Myriad Pro";
```

FormCalc

```
TextField1.font.typeface = "Myriad Pro"
```

underline

Specifies the activation and type of underlining.

Syntax

Reference_Syntax.underline = "0 | 1 | 2"

Values

Type	Values
String	• `0` (default) The font renders without underlining. • `1` The font renders with a single underline. • `2` The font renders with a double underline.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.underline = "2";
```

FormCalc

```
TextField1.font.underline = "2"
```

underlinePeriod

Controls the appearance of underlining.

Syntax

Reference_Syntax.underlinePeriod = "all | word"

Values

Type	Values
String	• `all` (default) The rendered line shall extend across word breaks. • `word` The rendered line shall be interrupted at word breaks.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.underlinePeriod = "word";
```

FormCalc

```
TextField1.font.underlinePeriod = "word"
```

upsMode

Represents the mode in a UPS Maxicode barcode.

Syntax

Reference_Syntax.upsMode = "usCarrier | internationalCarrier | standardSymbol | secureSymbol"

Values

Type	Values
String	• usCarrier (default) United States carrier with postal codes that contain up to nine digits. • internationalCarrier International carrier with alphanumeric postal codes that contain up to six digits. • standardSymbol Non-shipping encoded information up to 90 characters in length. • secureSymbol Non-shipping encoded information up to 74 characters in length (it has more error correction than four).

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.5

url

Specifies the URL for this object.

Syntax

Reference_Syntax.url = "*string*"

Values

Type	Values
String	A valid string representing a URL for this individual node.

Applies to

Model	Object
Form Model	certificates

Version

XFA 2.5

urlPolicy

(urlPolicy)Specifies the type of URL represented by the certificates object.

It is equivalent to the urlType attribute for PDF documents and its values are encoded as Browser, ASSP, or the string the user entered for the urlType key.

Syntax

Reference_Syntax.urlPolicy = "enrollmentServer | roamingCredentialServer | *string*"

Values

Type	Values
String	• enrollmentServer The URL references a web server where a signing party can enroll for a digital certificate. • roamingCredentialServer The URL references web service that holds the digital credentials that a signing party uses to sign a document or data. • A valid string that extends the use of this property with unique values.

Applies to

Model	Object
Form Model	certificates

Version

XFA 2.5

usage

Specifies the contexts in which to use the connection.

Syntax
Reference_Syntax.usage = "exportAndImport | exportOnly | importOnly"

Values

Type	Values
String	• exportAndImport (default) Used during both import and export. • exportOnly Used during export, ignored during import. • importOnly Used during import, ignored during export.

Applies to

Model	Object
Form Model	connect

Version
XFA 2.1

Examples

JavaScript
```
TextField1.connect.usage = "importOnly";
```

FormCalc
```
TextField1.connect.usage = "importOnly"
```

use

Invokes a prototype.

Syntax
Reference_Syntax.use = "*string*"

Values

Type	Values
String	The value of this property is a '#' character followed by the prototype's identifier.

Applies to

Model	Object
connectionSet Model	operation rootElement soapAction soapAddress uri wsdlAddress
Form Model	arc area assist barcode bookend boolean border break breakAfter breakBefore button calculate caption certificate certificates checkButton choiceList color comb
sourceSet Model	boolean command connect connectString

Version

XFA 2.1

usehref

Invokes an external prototype.

Note: The `usehref` property cannot target PDF files, even if the PDF files contain XML Form Object Model objects.

If an object contains both a `use` and a `usehref` property, the `usehref` property has precedence over the `use` property. This precedence allows a different prototype to be used when rendering form designs on legacy systems. Legacy systems will ignore the `usehref` property.

To mitigate security issues, specify HTTPS for the `usehref` URI or ensure that all prototype references occur behind a firewall.

Syntax

Reference_Syntax.usehref = "*string*"

Values

Type	Values

String	A valid string representing an external prototype. The value of this property includes a "#" character and the prototype's identifier:
	`usehref="URL#XML_ID"`
	`usehref="URL#ref(reference_syntax)"`

Applies to

Model	Object
connectionSet Model	operation rootElement soapAction soapAddress uri wsdlAddress
Form Model	arc area assist barcode bookend boolean border break breakAfter breakBefore button calculate caption certificate certificates checkButton choiceList color comb
sourceSet Model	bind boolean command connect

Version

XFA 2.4

uuid

Specifies the Universally Unique Identifier (UUID) for this object.

Syntax

Reference_Syntax.uuid = "*string*"

Values

Type	Values
String	A valid string representing a universally unique identifier for this individual node.

Applies to

Model	Object
XFA Model	xfa

Version

XFA 2.1

validationMessage

Specifies the validate message string for this field.

Syntax

Reference_Syntax.validationMessage = "*string*"

Values

Type	Values
String	A valid string representing a validation message to display to the user.

Applies to

Model	Object
Form Model	exclGroup field

Version

XFA 2.1

Examples

JavaScript

```
NumericField1.validationMessage = "This is the validation message."
```

FormCalc

```
NumericField1.validationMessage = "This is the validation message."
```

validationsEnabled

Specifies whether the validation scripts will execute.

Syntax

Reference_Syntax.validationsEnabled = "0 | 1"

Values

Type	Values
Boolean	• 0 (default) Validation scripts are disabled. • 1 Validation scripts are enabled.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.validationsEnabled = "1";
```

FormCalc

```
xfa.host.validationsEnabled = "1"
```

vAlign

Specifies the vertical text alignment.

Syntax

Reference_Syntax.vAlign = "top | middle | bottom"

Values

Type	Values
String	• `top` (default) Align with the top of the available region. • `middle` Center vertically within the available region. • `bottom` Align with the bottom of the available region.

Applies to

Model	Object
Form Model	draw exclGroup field para subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.vAlign = "top";
```

FormCalc

```
TextField1.vAlign = "top"
```

value

Specifies the value of the current object.

Specifies a comma separated list of values for each color component of the color space.

Syntax

```
Reference_Syntax.value = "various"
```

Values

Type	Values
Varies	Values differ depending on the referencing object. For example, the `value` property of a `field` object is a string representing the actual value displayed in the field, or the field's bound value. Alternatively, for objects that require an color value, this property specifies a comma-separated list of values for each color component of the color space. For the color-space of SRGB, the component values must be r,g,b, where r is the red component value, g is the green component value, and b is the blue component value. Each component value must be in the range 0 through 255, inclusive. 255 represents maximum display intensity. For example, 255,0,0 specifies the color red. The default is dependent upon the context of where the color is used; the default color is determined by the object enclosing the color object.

Applies to

Model	Object
Data Model	dataValue
Form Model	boolean color date dateTime decimal float image integer picture script text time
sourceSet Model	boolean integer text

Also applies to objects derived from the textNode class.

Version

XFA 2.1

Examples

JavaScript

```
// Use the value property to set and get the document variable's value.
TextField1.rawValue = docVar.value;
```

FormCalc

```
// Use the value property to set and get the document variable's value.
TextField1 = docVar.value
```

See also

- "Creating a node in the data model" on page 422
- "Manipulating instances of a subform" on page 424
- "Getting or setting object values" on page 425
- "Concatenating data values" on page 427
- "Changing the background color" on page 428
- "Populating a drop-down list" on page 430

valueRef

Resolves a data value for each data node in the set identified by the `ref` object.

The data values are then used to populate the value items, such as `<items save='1'>`.

The `valueRef` property is a relative reference syntax expression.

Note: This property is read only.

Syntax

`Reference_Syntax``.valueRef = "`*`string`*`"`

Values

Type	Values
String	A valid string representing a data value for each data node in the set.

Applies to

Model	Object
Form Model	bindItems

Version

XFA 2.4

variation

Indicates the packaging of the application that is running the script.

It is available only for client applications.

Note: This property is read only.

Syntax

`Reference_Syntax``.variation`

Values

Type	Values
String	A valid string representing the packaging of the application. For example, in the case of a PDF form in Acrobat, this property returns one of: `Reader`, `Fill-in`, `Business Tools`, or `Full`.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.variation;
```

FormCalc

```
xfa.host.variation
```

version

Indicates the version number of the current application.

Note: This property is read only.

Syntax

Reference_Syntax`.version`

Values

Type	Values
String	A valid string representing the packaging of the application. For example, in Acrobat 6.0.1 this property returns `6.0.1`.

Applies to

Model	Object
Form Model	handler
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.version;
```

FormCalc
```
xfa.host.version
```

vScollPolicy

Specifies whether a field can scroll vertically.

Note: This property does not apply to Text Fields that can expand to accommodate data or text.

Syntax
Reference_Syntax.hScrollPolicy = "auto | on | off"

Values

Type	Values
String	• `auto` (default) Single-line fields scroll horizontally and multi-line fields scroll vertically (displaying a vertical scroll bar when necessary). • `on` Vertical and/or horizontal scroll bars appear regardless of whether the text or data overflows the boundaries of the field. • `off` Restricts the user from entering characters in the field beyond what can physically fit within the field width. Note that this restriction does not apply to data with the field.

Applies to

Model	Object
Form Model	textEdit

Version

XFA 2.5

Examples

JavaScript
```
TextField1.resolveNode("ui.#textEdit").vScrollPolicy = "off";
```

FormCalc
```
TextField1.ui.#textEdit.vScrollPolicy = "off"
```

w

A measurement specifying the width for the layout.

When you specify a width, that value overrides any growth range allowed by the minW property and the maxW property. Omitting this property or specifying an empty string indicates that the minW property and the maxW property define the width for the object.

Syntax

Reference_Syntax.w = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement.

Applies to

Model	Object
Form Model	draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.w = "3in";
```

FormCalc

```
TextField1.w = "3in"
```

weight

Controls the weight of the font typeface.

Syntax

Reference_Syntax.weight = "bold | normal"

Values

Type	Values
String	• bold (default) The typeface is rendered with a bold weight. • normal The typeface is rendered at the default typeface weight.

Applies to

Model	Object
Form Model	font

Version

XFA 2.1

Examples

JavaScript

```
TextField1.font.weight = "normal";
```

FormCalc

```
TextField1.font.weight = "normal"
```

wideNarrowRatio

Specifies a ratio of wide bar to narrow bar in supported barcodes.

The allowable range of ratios varies between barcode formats and also, for hardware barcodes, the output device. The template must not specify a value outside the allowable range. The parser ignores this property for barcode formats which do not allow a variable ratio of wide to narrow bar widths.

Syntax

Reference_Syntax.wideNarrowRatio = "3:1 | wide[:narrow]"

Values

Type	Values
String	• `3:1` (default) Any valid ratio that uses the syntax: • *`wide`*`[:`*`narrow`*`]` where *`wide`* is a positive number representing the numerator of the ratio, and *`narrow`* is an optional positive number representing the denominator of the ratio. If narrow is not supplied it defaults to `1`.

Applies to

Model	Object
Form Model	barcode

Version

XFA 2.1

Examples

JavaScript

```
Barcode1.resolveNode("ui.#barcode").wideNarrowRatio = "5:1";
```

FormCalc

```
Barcode1.ui.#barcode.wideNarrowRatio = "5:1"
```

x

Specifies the X coordinate of the container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.

Containers with flowed content do not use x coordinates.

Syntax

Reference_Syntax.x = "0in | *measurement*"

Values

Type	Values
String	• 0in (default) • Any valid measurement value.

Applies to

Model	Object
Form Model	area contentArea draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.x = "5in";
```

FormCalc

```
TextField.x = "5in"
```

xdpContent

Controls what subset of the data is submitted. This property is used only when the format property is xdp.

Syntax

Reference_Syntax.xdpContent = "*string*"

Values

Type	Values
String	• `datasets pdf xfdf` (default) Submits objects with the tags datasets, pdf, and xfdf to the host. • `tag1 tag2 ... tagN` Submits objects with tags matching any of the specified tags. • `*` (asterisk) Submits all data objects to the host.

Applies to

Model	Object
Form Model	submit

Version

XFA 2.1

Examples

JavaScript

```
Button1.resolveNode("#event.#submit").xdpContent = "*"
```

FormCalc

```
Button1.#event.#submit.xdpContent = "*"
```

y

Specifies the y coordinate of a container's anchor point relative to the top-left corner of the parent container when placed with positioned layout.

Containers with flowed content do not use y coordinates.

Syntax

Reference_Syntax`.y = "0in | `*measurement*`"`

Values

Type	Values
String	• `0in` (default) • Any valid measurement value.

Applies to

Model	Object
Form Model	area contentArea draw exclGroup field subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.y = "5in";
```

FormCalc

```
TextField.y = "5in"
```

5 Scripting Methods

This section provides an alphabetical list of all methods supported in this scripting environment.

Each host, such as Acrobat and Adobe Reader is responsible for implementing the available methods. Some methods do not make sense on a server, such as `beep`. The server does not implement these methods and instead can output an error message if a user tries to call the method.

absPage

Determines the page of the form that a given form design object first appears on.

Syntax

`Reference_Syntax``.absPage( OBJECT` *`param`* `)`

Parameters

param	The fully qualified reference syntax expression of one of the following form design objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the page of the form (0-based). This method returns `-1` if the object specified in *`param`* cannot be found on the form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
TextField2.rawValue = xfa.layout.absPage(this);
```

FormCalc

```
TextField2 = xfa.layout.absPage($)
```

See also

"Working with page numbers and page counts" on page 426

absPageCount

Determines the page count of the current form.

Syntax
Reference_Syntax.absPageCount()

Parameters

None

Returns

An integer representing the number of pages in the current form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
TextField2.rawValue = xfa.layout.absPageCount();
```

FormCalc
```
TextField2 = xfa.layout.absPageCount()
```

See also

"Working with page numbers and page counts" on page 426

absPageCountInBatch

Determines the page count of the current batch.

Syntax
Reference_Syntax.absPageCountInBatch()

Parameters

None

Returns

An integer representing the page count of the current batch.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

absPageInBatch

Determines which page within the batch contains the form object.

Syntax

Reference_Syntax.absPageInBatch(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following form objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the page number that contains the form object.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.5

absPageSpan

Determines the number of pages that a specified form object spans.

Syntax

Reference_Syntax.absPageSpan(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following form objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the number of pages the specified form object spans. This method returns `-1` if the object specified in *param* cannot be found on the form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.absPageSpan(Subform1);
```

FormCalc
```
xfa.layout.absPageSpan(Subform1)
```

addInstance

Adds a new instance of a subform or subform set to the form model.

Syntax
Reference_Syntax.addInstance(BOOLEAN *param*)

Parameters

param (Optional)	Indicates if the new subform or subform set has a corresponding data value in the data model. • `True` (default) Merge the new subform or subform set with the data model. • `False` Do not perform a merge operation.

Returns

The new form object, or null if no object was added.

Applies to

Model	Object
Form Model	instanceManager

Version

XFA 2.1

Examples

JavaScript
```
Subform1.instanceManager.addInstance(1);
```

FormCalc
```
Subform1.instanceManager.addInstance(1)
```

See also

"Manipulating instances of a subform" on page 424

addItem

Adds new items to the current form field. For example, this method adds new items to a drop-down list.

Syntax
Reference_Syntax.addItem(STRING *param1* [, STRING *param2*])

Parameters

param1	A valid string representing the value to display in the field.
param2 (Optional)	A valid string representing the new item's bound value. If empty, the default value is an empty string.

Returns

Empty

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
DropDownList1.addItem("Human Resources");
```

FormCalc

```
DropDownList1.addItem("Human Resources")
```

See also

"Populating a drop-down list" on page 430

addNew

Appends a new record to the record set.

Syntax

Reference_Syntax.addNew()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.addNew();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.addNew();
```

append

Appends a node to the end of the node list.

Syntax

Reference_Syntax.append(OBJECT *param*)

Parameters

param	A valid reference syntax expression representing the node to be appended.

Returns

Empty

Applies to

list class

Version

XFA 2.1

Examples

JavaScript

```
// Append a data group node to another data model node.
xfa.record.nodes.append(oGroupNode);
```

FormCalc

```
// Append a data group node to another data model node.
xfa.record.nodes.append(oGroupNode)
```

See also

"Creating a node in the data model" on page 422

applyXSL

Applies an XSL transformation to the XML representation of the current node. It is equivalent to calling saveXML and transforming the result with the specified XSL document.

Syntax

Reference_Syntax.applyXSL(STRING *param*)

Parameters

param	A valid string representing the XSL transformation input to apply.

Returns

A valid string representing the result of the XSL transformation.

Applies to

node class

Version

XFA 2.1

assignNode

Evaluates the reference syntax expression using the current context and sets the value of the found node. If the node doesn't exist, it can be created.

Syntax

```
Reference_Syntax.assignNode( STRING param1 [, STRING param2 [, INTEGER param3 ] ] )
```

Parameters

param1	A valid string representing a reference syntax expression that points to a particular node.
param2 (Optional)	A valid string representing the value to assign to the node.
param3 (Optional)	An integer value representing the action to take when creating new nodes. The following are the valid parameter values: • 0 If the node exists, the value is updated. If the node doesn't exist, it is created. • 1 If the node exists, an error will be thrown. If the node doesn't exist, it is created. • 2 If the node exists, no action is taken. If the node doesn't exist, it is created. • 3 A new node is always created.

Returns

An object corresponding to the specified node.

Applies to

node class

Version

XFA 2.1

beep

Causes the system to play a sound. It is available only for client applications.

Syntax

Reference_Syntax.beep([INTEGER *param*])

Parameters

param (Optional)	The system code for the appropriate sound. • 0 (Error) • 1 (Warning) • 2 (Question) • 3 (Status) • 4 (Default - The string representing the display value.)

Returns

Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.beep("2");
```

FormCalc

```
xfa.host.beep("2")
```

boundItem

Gets the bound value of a specific display item of a drop-down list or list box.

Syntax

Reference_Syntax.boundItem(STRING *param*)

Parameters

param	A valid string representing the display value that appears in the list box or drop-down list.

Returns

A valid string representing the bound value of a specified display value.

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript

```
DropDownList1.boundItem("Text");
```

FormCalc

```
DropDownList1.boundItem("Text")
```

cancel

Cancels any changes made to the current or new row of a record set object, or the field collection of a record object, prior to calling the update method.

Syntax

Reference_Syntax.cancel()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.cancel();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.cancel()
```

cancelBatch

Cancels a pending batch update.

Syntax
Reference_Syntax.cancelBatch()

Parameters
None

Returns
Empty

Applies to

Model	Object
sourceSet Model	source

Version
XFA 2.1

clear

Removes a given signature.

Syntax
Reference_Syntax.clear(OBJECT *param1* [, BOOLEAN *param2*])

Parameters

param1	input	A valid XML signature node.
param2 (Optional)	input (Optional)	• True (default) Indicates that a dialog box is used for this operation. • False Indicates that a dialog box is not used for this operation.

Returns
True if the signature was removed successfully, and False if the signature was not removed successfully.

Applies to

Model	Object
Signature Model	signaturePseudoModel

Version
XFA 2.1

clearErrorList

Removes all items from the current error log.

Syntax
Reference_Syntax.clearErrorList()

Parameters

None

Returns
Empty

Applies to

model class

Version

XFA 2.1

clearItems

Removes all the items from the field. For example, it removes all the items contained within a drop-down list or a list box.

Syntax
Reference_Syntax.clearItems()

Parameters

None

Returns
Empty

Applies to

Model	Object
Form Model	field

Version

XFA 2.1

Examples

JavaScript
```
DropDownList1.clearItems();
```

FormCalc
```
DropDownList1.clearItems()
```

See also

"Populating a drop-down list" on page 430

clone

Makes a copy of an object.

Syntax

Reference_Syntax.clone(BOOLEAN *param*)

Parameters

param	A Boolean value indicating if cloning should occur recursively. • `True` (Default) Clone the object recursively. • `False` Do not clone the object recursively.

Returns

The duplicate copy of the object.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
xfa.record.NewGroupNode.clone(1);
```

FormCalc

```
xfa.record.NewGroupNode.clone(1)
```

See also

"Creating a node in the data model" on page 422

close

Closes a connection to a data source.

Syntax

Reference_Syntax.close()

Parameters

None

Returns
Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript
```
xfa.sourceSet.nodes.item(nIndex).close();
```

FormCalc
```
xfa.sourceSet.nodes.item(nIndex).close()
```

createNode

Creates a new node based on a valid class name.

Note: You cannot use the `createNode` method to create any of the following XML Form Object Model objects:

- area
- draw
- exclGroup
- pageArea
- pageSet
- subform
- subformSet

Syntax
```
Reference_Syntax.createNode( STRING param1 [, STRING param2 [, STRING param3 ]
] )
```

Parameters

param1	A valid string representing the class name of the object to create.
param2 (Optional)	A valid string representing the name to assign to the node. If empty, the value of this parameter defaults to an empty string.
param3 (Optional)	A valid string representing the XML namespace that the created node will exist in. If empty, the value of this parameter defaults to an empty string.

Returns

An object representing a valid node.

Applies to

model class.

Version

XFA 2.1

Examples

JavaScript

```
// Create a node of type dataGroup.
var oGroupNode = xfa.datasets.createNode("dataGroup", "NewGroupNode");
```

FormCalc

```
// Create a node of type dataGroup.
var oGroupNode = xfa.datasets.createNode("dataGroup", "NewGroupNode")
```

See also

"Creating a node in the data model" on page 422

delete

Deletes the current record from the record set.

Syntax

Reference_Syntax.delete()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.delete();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.delete()
```

deleteItem

Deletes the specified item.

Syntax

Reference_Syntax.deleteItem(INTEGER *param*)

Parameters

param	A valid integer representing the zero-based index into the item.

Returns

True if the item was deleted and false if it was not deleted.

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

Example

JavaScript

```
ListBox1.deleteItem(ListBox1.selectedIndex);
```

FormCalc

```
ListBox1.deleteItem(ListBox1.selectedIndex)
```

documentCountInBatch

Determines the number of documents in the current batch.

Syntax

Reference_Syntax.documentCountInBatch()

Parameters

None

Returns

An integer representing the total number of documents in the batch. Hosts that do not support batching return `1`.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.5

documentInBatch

Determines the ordinal number of the current document within the batch.

Syntax
Reference_Syntax.documentInBatch()

Parameters

None

Returns

An integer representing a physical document number (zero based). Hosts that do not support batching return 0.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.5

emit

Notifies the form event manager that an event has occurred.

Syntax
Reference_Syntax.emit()

Parameters

None

Returns
Empty

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.event.emit();
```

FormCalc
```
xfa.event.emit()
```

enumerate

Enumerates all the XML signatures found in the document.

Syntax
Reference_Syntax.enumerate()

Parameters

None

Returns

An object representing an XFA node list of all the XML signature nodes.

Applies to

Model	Object
Signature Model	signaturePseudoModel

Version

XFA 2.1

evaluate

Gets the list of objects referred to in the manifest.

Syntax
Reference_Syntax.evaluate()

Parameters

None

Returns

An object representing the list of objects.

Applies to

Model	Object
Form Model	manifest

Version

XFA 2.1

execCalculate

Executes the calculate script of the field.

Syntax
Reference_Syntax.execCalculate()

Parameters

None

Returns

Empty

Applies to

Model	Object
Form Model	exclGroup field form subform

Version

XFA 2.1

Examples

JavaScript

```
TextField1.execCalculate();
```

FormCalc

```
TextField1.execCalculate()
```

execEvent

Executes the event script of the object.

Syntax

Reference_Syntax.execEvent(STRING *param*)

Parameters

param	A valid string representing the name of the event to execute.

Returns

Empty

Applies to

Model	Object
Form Model	exclGroup field subform

Version

XFA 2.1

Examples

JavaScript
```
Button1.execEvent("click");
```

FormCalc
```
Button1.execEvent("click")
```

execInitialize

Executes the initialize script of the field.

Syntax
Reference_Syntax.execInitialize()

Parameters

None

Returns

Empty

Applies to

Model	Object
Form Model	exclGroup field form subform

Version

XFA 2.1

Examples

JavaScript
```
Subform1.execInitialize();
```

FormCalc
```
Subform1.execInitialize()
```

execute

Executes a connection.

Syntax
Reference_Syntax.execute(BOOLEAN *param*)

Parameters

param	• `True` (Default) Forces the remerging of the form design and the imported WSDL data. • `False` Imports the WSDL data into current Form without merging it with the form design.

Returns

`True` if the connection was executed successfully, and `false` if it is unsuccessful.

Applies to

Model	Object
connectionSet Model	wsdlConnection

Version

XFA 2.1

execValidate

Executes the validate script of the field.

Syntax

Reference_Syntax`.execValidate()`

Parameters

None

Returns

Empty

Applies to

Model	Object
Form Model	field form subform

Version

XFA 2.1

Examples

JavaScript

```
NumericField1.execValidate();
```

FormCalc

```
NumericField1.execValidate()
```

exportData

Exports the data from the current form in either XDP or XML format to a file.

For security reasons, if you provide the first parameter, the `exportData` method executes only when performed on certified documents. If you do not provide the first parameter, the document does not need to be certified and the user is prompted to provide a location and file name.

Syntax

Reference_Syntax`.exportData( [ STRING` *param1* `[, BOOLEAN` *param2* `] ])`

Parameters

param1 (Optional)	Specifies the location and file name of the file where the data will export. If you omit this parameter, a dialog box opens to let the user select the file manually. **Note:** This parameter is only valid on certified documents where the user has sufficient permissions.
param2 (Optional)	Specifies the export format for the data. • `True` (Default) Export to XDP format. • `False` Export plain XML data. **Note:** To change the export type without specifying a file name, you must provide an empty string as the first parameter. For example: `xfa.host.exportData("",0)`

Returns

Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.exportData("filename.xdp");
```

FormCalc

```
xfa.host.exportData("filename.xdp")
```

See also

"Saving a form" on page 431

first

Moves to the first record in the record set, and populates the data model with the record data.

Note: The data connection method `xfa.sourceSet.DataConnection.first` looks up a table and updates the table if the data has changed. It uses the hasDataChanged method to determine whether the data has changed.

Syntax
Reference_Syntax`.first()`

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript
```
xfa.sourceSet.dataConnectionName.first();
```

FormCalc
```
xfa.sourceSet.dataConnectionName.first()
```

formNodes

Returns a list of all form model objects that are bound to a specified data object.

Syntax
Reference_Syntax`.formNodes( OBJECT` *param* `)`

Parameters

param	A valid reference syntax expression representing a data model object.

Returns

An object representing the list of all form model objects that have a relationship with the specified data object.

Applies to

Model	Object
Form Model	form

Version

XFA 2.1

getAttribute

Gets a specified property value.

Syntax

Reference_Syntax.getAttribute(STRING *param*)

Parameters

param	A valid string representing the name of the property to retrieve.

Returns

A valid string representing the value of the property.

Applies to

Model	Object
XFA Model	packet

Also applies to the node class.

Version

XFA 2.1

Examples

JavaScript

```
var sBOFBackup =
oDB.nodes.item(nIndex).query.recordSet.getAttribute("bofAction");
```

FormCalc

```
var sBOFBackup =
oDB.nodes.item(nIndex).query.recordSet.getAttribute("bofAction")
```

getDelta

Gets a delta script object for a specific property.

Syntax

Reference_Syntax.getDelta(STRING param)

Parameters

param	A string representing the reference syntax to a property.

Returns

A valid object representing a delta script object.

Applies to

container class

Version

XFA 2.5

getDeltas

Recursively gets all the deltas script objects for this container object and all its descendants.

Note: Depending on the number of deltas script objects, this method can negatively affect the run time performance of your form.

Syntax

```
Reference_Syntax.getDeltas( )
```

Parameters

None

Returns

A valid object representing a deltas script object.

Applies to

container class

Version

XFA 2.5

getDisplayItem

Retrieves the item display text for the specified item index.

Syntax

```
Reference_Syntax.getDisplayItem( INTEGER param )
```

Parameters

param1	An integer representing the zero-based index into the item.

Returns

A valid string representing the text of the item or null if no display item exists.

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

getElement

Returns a specified object property.

Syntax

Reference_Syntax.getElement(STRING *param1* [, INTEGER *param2*])

Parameters

param1	A valid string representing the name of the object to retrieve.
param2 (Optional)	An integer value representing the instance of the object to retrieve.

Returns

The specified object.

Applies to

node class

Version

XFA 2.1

getFocus

Finds and returns the form object that currently has the input focus.

Syntax

Reference_Syntax.getFocus()

Parameters

None

Returns

The form object that currently has the input focus, or null if no form object has the input focus.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.6

getItemState

Returns the selection state of the specified item.

Syntax

Reference_Syntax.getItemState(INTEGER param)

Parameters

param	A valid integer representing the zero-based index into the item.

Returns

True if the item was selected and false if it was not selected.

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

getSaveItem

Retrieves the data value for the specified item index.

Syntax

Reference_Syntax.getSaveItem(INTEGER *param*)

Parameters

param	A valid integer representing the zero-based index into the item.

Returns

A valid string representing the text of the data item or null if no data item exists.

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

gotoRecord

Moves the current record of the data window to a particular record within the range of records in the data.

Syntax

Reference_Syntax.gotoRecord(INTEGER *param*)

Parameters

param	A valid integer value representing the specified record in the range of records.

Returns

Empty

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript

```
xfa.dataWindow.gotoRecord(2);
```

FormCalc

```
xfa.dataWindow.gotoRecord(2)
```

gotoURL

Retrieves the specified URL. It is available only for client applications.

Syntax

Reference_Syntax.gotoURL(STRING *param1* [, BOOLEAN *param2*])

Parameters

param1	A valid string representing a fully qualified or a relative URL. It is possible to include a query string at the end of the URL. If the form is being viewed inside a browser or Acrobat Capture® is not available, the Weblink plug-in retrieves the requested URL. If the form is running inside Acrobat, the URL of the current document is obtained either from the document's base URL, from the URL of page 0 (if the document was Web Captured), or from the file system.
param2 (Optional)	• `True` (default) Appends the resulting pages to the current document. • `False` This flag is false if the document is running inside the web browser, the Acrobat Capture plug-in is not available, or if the URL is of type 'file:///'.

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.host.gotoURL("http://www.adobe.com");
```

FormCalc
```
xfa.host.gotoURL("http://www.adobe.com")
```

h

Determines the height of a given form design object.

Syntax
Reference_Syntax.h(OBJECT *param1* [, STRING *param2* [, INTEGER *param3*]])

Parameters

param1	The fully qualified reference syntax expression of one of the following container XML Form Object Model objects: area, contentArea, draw, field, pageArea, subform.
param2 (Optional)	A string representing the unit type of the return value. If left blank, the default unit type is points.
param3 (Optional)	An integer representing the amount to offset the height value of a form design object, beginning with the first page the object occurs on. If left blank, the default value is 0.

Returns
The height of the form design object.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.layout.h(TextField1,"in");
```

FormCalc
`xfa.layout.h(TextField1,"in")`

hasDataChanged

Determines whether the current record data has changed.

This method is a pre-commit test of the active record. It compares the current record data with the record data from the current data source. If the data has changed, then this method returns `true`.

Note: The data connection methods xfa.sourceSet.DataConnection.first, xfa.sourceSet.DataConnection.next, xfa.sourceSet.DataConnection.previous, and xfa.sourceSet.DataConnection.last perform an implict update if the data has changed.

Syntax
`Reference_Syntax``.hasDataChanged()`

Parameters

None

Returns
`True` if the data has changed, and `false` if the data has not changed.

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

importData

Imports data to the current form from a specified file.

For security reasons, if you provide the parameter, the `importData` method executes only when performed on certified documents. If you do not provide the parameter, the document does not need to be certified and the user is prompted to provide a location and file name.

Syntax
`Reference_Syntax``.importData( [ STRING` *`param`* `] )`

Parameters

param (Optional)	A valid string representing the location and name of the file from which the data will be imported. If you omit this parameter, a dialog box opens to let the user select the file manually. **Note:** This parameter is valid only on certified documents where the user has sufficient permissions.

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version
XFA 2.1

Examples

JavaScript
```
xfa.host.importData("filename.xdp");
```

FormCalc
```
xfa.host.importData("filename.xdp")
```

insert

Inserts a node before a specific node in the node list.

Syntax
Reference_Syntax.insert(OBJECT *param1*, OBJECT *param2*)

Parameters

param1	A valid reference syntax expression representing the node to be inserted.
param2	A valid reference syntax expression representing the node to insert before.

Returns
Empty

Applies to
list class

Version
XFA 2.1

Examples

JavaScript
```
xfa.datasets.connectionData.DataConnection.nodes.insert(oHeader,oFirst);
```

FormCalc
```
xfa.datasets.connectionData.DataConnection.nodes.insert(oHeader,oFirst)
```

insertInstance

Inserts a new instance of a subform or subform set into a form.

Syntax

Reference_Syntax.insertInstance(INTEGER *param1* [, BOOLEAN *param2*])

Parameters

param1	An integer specifying the zero-indexed position to insert the instance within a set of instances.
param2 (optional)	A Boolean value indicating if data should be merged with the new subform instance. • True Merges the new subform instance with the available data. • False The new subform instance is not merged with data.

Returns

An object representing the new instance of the subform or subform set.

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
Subform1.instanceManager.insertInstance(3, false);
```

FormCalc

```
Subform1.instanceManager.insertInstance(3, false)
```

isBOF

Determines if the current location is at the beginning of the record set. The bofAction property must be set to `stayBOF`.

Syntax

Reference_Syntax.isBOF()

Parameters

None

Returns

`True` if the current location is at the beginning of the record set, and `False` if the current location is not at the beginning of the record set.

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.nodes.item(nIndex).isBOF();
```

FormCalc

```
xfa.sourceSet.nodes.item(nIndex).isBOF()
```

isCompatibleNS

Determines if a specified namespace is functionally equivalent, that is compatible, with the namespace of this model. It determines if the two namespaces are equivalent, even though the strings that represent them may not be identical.

Syntax

Reference_Syntax.isCompatibleNS(STRING *param*)

Parameters

param	A valid string representing the namespace to compare.

Returns

`True` if the namespaces are equivalent and `False` if they are not compatible.

Applies to

model class

Version

XFA 2.1

isEOF

Determines if the current location is at the end of the record set. The eofAction property must be set to `stayEOF`.

Syntax

Reference_Syntax.isEOF()

Parameters

None

Returns

`True` if the current location is at the end of the record set, and `false` if the current location is not at the end of the record set.

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.nodes.item(nIndex).isEOF();
```

FormCalc

```
xfa.sourceSet.nodes.item(nIndex).isEOF()
```

isPropertySpecified

Checks if a specific property has been defined for this node.

Syntax

Reference_Syntax`.isPropertySpecified( STRING` *param1* `[, BOOLEAN` *param2* `[, INTEGER` *param3* `] ] )`

Parameters

param1	A valid string representing the name of the object property to search on.
param2 (Optional)	A Boolean value that indicates if inheritance from parent classes should be taken into consideration. • `True` (default) Determines if this property is inherited from a parent class. • `False` Determines if this property is defined for the current object, regardless of inheritance.
param3 (Optional)	An integer value specifying which occurrence of the property to examine. This parameter is only valid for those properties that can have multiple instances.

Returns

`True` if the property is specified and `false` if it is not specified.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
TextField1.isPropertySpecified("ui");
```

FormCalc

```
TextField1.isPropertySpecified("ui")
```

isRecordGroup

Indicates if a particular dataGroup object is also a single record.

Syntax

Reference_Syntax.isRecordGroup(OBJECT *param*)

Parameters

param	A valid dataGroup object from the current data source.

Returns

`True` if the specified data group is also a single record, and `false` if it is not.

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript

```
xfa.dataWindow.isRecordGroup(xfa.datasets.data.dataNodeName);
```

FormCalc

```
xfa.dataWindow.isRecordGroup(xfa.datasets.data.dataNodeName)
```

item

Describes a zero-based index into the collection.

Syntax

Reference_Syntax.item(INTEGER *param*)

Parameters

param	A zero-based index into the collection.

Returns

An object representing an XFA tree.

Applies to

list class

Version

XFA 2.1

See also

- "Referencing objects" on page 420
- "Changing the background color" on page 428
- "Populating a drop-down list" on page 430
- "Disabling all form fields" on page 434

last

Moves to the last record in the record set, and populates the data model with the record data.

Note: The data connection method `xfa.sourceSet.DataConnection.last` looks up a table and updates the table if the data has changed. It uses the hasDataChanged method to determine whether the data has changed.

Syntax

Reference_Syntax`.last()`

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.last();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.last()
```

loadXML

Loads and appends a specified XML document to the current object.

Syntax

Reference_Syntax.EMPTY loadXML(STRING *param1* [, BOOLEAN *param2* [, BOOLEAN *param3*]])

Parameters

param1	A valid string representing the name of the XML document.
param2 (Optional)	A Boolean value indicating if the root node within the XML document should be ignored. • `True` (default) Ignores the root node of the XML document, and appends the remaining XML nodes directly to the current XML Form Object Model object. • `False` Appends the root node of the XML document directly to the current XML Form Object Model object.
param3 (Optional)	A Boolean value indicating if the data from the XML document should overwrite the information for the current XML Form Object Model object. • `True` Replaces the content of the current XML Form Object Model object with the XML document data. • `False` (default) Appends the XML document data to the current XML Form Object Model object.

Returns

Empty

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
xfa.datasets.data.loadXML(xmlData,0,1);
```

FormCalc

```
xfa.datasets.data.loadXML(xmlData,0,1)
```

messageBox

Displays a dialog box on the screen. It is available only for client applications.

Syntax

Reference_Syntax.messageBox(STRING *param1* [, STRING *param2* [, INTEGER *param3* [, INTEGER *param4*]]])

Parameters

param1	A valid string representing the message to display.
param2 (Optional)	A valid string representing the title to appear in the title bar of the dialog window. To help protect against internet spoofing, the dialog window title begins with the text "Warning: JavaScript Window -". The window title that you specify in this parameter displays after the warning text.
param3 (Optional)	An integer representing the icon to display in the dialog box. • 0 (Error) - This is the default. • 1 (Warning) • 2 (Question) • 3 (Status)
param4 (Optional)	An integer representing the buttons to display. • 0 (OK) - This is the default. • 1 (OK, Cancel) • 2 (Yes, No) • 3 (Yes, No, Cancel)

While *param2*, *param3*, and *param4* are optional, if you want to include a particular parameter, you must also include all of the preceding parameters. For example, the following JavaScript is incorrect:

```
xfa.host.messageBox("Hello World!",3,1);
```

In this case you must also specify a value for *param2* for the JavaScript to execute correctly.

Returns

A valid integer representing the value of the button pressed by the user:

- 1 (OK)
- 2 (Cancel)
- 3 (No)
- 4 (Yes)

Caution: In a rendered form guide, the return value of the messageBox method is always 0, regardless of what button the user selects.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.messageBox("This is a message", "This is a title", 3, 1);
```

FormCalc

```
xfa.host.messageBox("This is a message", "This is a title", 3, 1)
```

See also

- "Creating a node in the data model" on page 422
- "Populating a drop-down list" on page 430
- "Making an object visible or invisible" on page 432
- "Using radio buttons and check boxes" on page 433
- "Determining that a form has changed" on page 433

metadata

Collects a comprehensive Extensible Metadata Platform (XMP) metadata packet for the document.

Any third-party metadata is collected and converted to XMP as follows:

- All elements are given the namespace http://ns.adobe.com/xfa/promoted-desc/, with the suggested prefix `desc:`.
- The value of the `name` object becomes the object name.
- A `desc:ref` property qualifier is added, whose value is an XPath expression pointing back to the parent of the original `desc`. The order of desc objects within a single parent is not preserved. Multiple desc objects of the same name are not collected. Only the first desc object appears in the output.
- Content under the desc object is converted as follows:

Object	XMP type
boolean	Boolean
date	Date
dateTime	Date
decimal	Real
exData	external: URI embedded: Thumbnail
float	Real
image	external: URI embedded: Thumbnail
integer	Integer
text	Text
time	Date

When the XDP file is rendered as a PDF file, the collected metadata is written to the PDF file's XMP packet. Copies of the same metadata continue to exist in the XFA stream inside the PDF file.

Syntax

Reference_Syntax.metadata (INTEGER param)

Parameters

param	(Optional) An integer representing the serialization format. • 0 (RDF) (default) • 1 (PlainXMP)

Returns

A valid string representing the XML serialization of the XMP metadata.

Applies to

Model	Object
Form Model	desc

Version

XFA 2.5

moveCurrentRecord

Repositions the current record to another location within the range of records.

Syntax

Reference_Syntax.moveCurrentRecord(INTEGER *param*)

Parameters

param	A valid integer representing the number of records separating the current record and the desired destination record. A positive integer indicates a record between the current record and the end of the range of records, a negative value indicates a record between the current record and the beginning of the range.

Returns

Empty

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript

```
xfa.dataWindow.moveCurrentRecord(1);
```

FormCalc

```
xfa.dataWindow.moveCurrentRecord(1)
```

moveInstance

Moves a `subform` object within a set of subform instances.

The corresponding data model information for the subform is also relocated within the data model.

Syntax

Reference_Syntax`.moveInstance( INTEGER` *param1*`, INTEGER` *param2* `)`

Parameters

param1	A valid integer representing the 0 based index position of the form model object to move.
param2	A valid integer representing the 0 based position of the child object within the set of instances.

Returns

Empty

Applies to

Model	Object
Form Model	instanceManager

Version

XFA 2.1

Examples

JavaScript

```
Subform1.instanceManager.moveInstance(0,6);
```

FormCalc

```
Subform1.instanceManager.moveInstance(0,6)
```

See also

"Manipulating instances of a subform" on page 424

namedItem

Gets the first child of this node with the given name.

Syntax

Reference_Syntax`.namedItem( STRING` *param* `)`

Parameters

param	A valid string representing the name of this node.

Returns

An object representing the first child of this node with the given name.

Applies to

treeList class

Version

XFA 2.1

next

Moves to the next record in the record set, and populates the data model with the record data.

Note: The data connection method `xfa.sourceSet.DataConnection.next` looks up a table and updates the table if the data has changed. It uses the hasDataChanged method to determine whether the data has changed.

Syntax

Reference_Syntax`.next()`

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.next();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.next()
```

open

Connects to the data source and populates the data model with the results of the current record.

Syntax
Reference_Syntax.open()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript
```
xfa.sourceSet.nodes.item(nIndex).open();
```

FormCalc
```
xfa.sourceSet.nodes.item(nIndex).open()
```

openList

Opens the drop-down list specified by the reference syntax expression.

It is available only for client applications.

Syntax
Reference_Syntax.openList(OBJECT *param*)
Reference_Syntax.openList(STRING *param*) (deprecated)

Parameters

param	A fully qualified reference syntax expression that specifies a drop-down list.

Returns

Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.6

XFA 2.1 (deprecated)

page

Determines the page number that contains a given form design object. If the object spans multiple pages, this method returns the first page the object occurs on.

Syntax

Reference_Syntax.page(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following a container form design objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the logical page number (based on the initial page number) that contains the specified form object. This method returns 0 if the object specified in *param* cannot be found on the form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.page(this);
```

FormCalc

```
xfa.layout.page($)
```

See also

"Working with page numbers and page counts" on page 426

pageContent

Retrieves types of form design objects from a specified page of a form.

Syntax

Reference_Syntax.pageContent(INTEGER *param1* [, STRING *param2* [, BOOLEAN *param3*]])

Parameters

param1	An integer representing the desired page number. This value is 0-based.
param2 (Optional)	Return the following types of containers: • `field` Returns all of the following form design objects: Button, Check Box, Date/Time Field, Drop-down List, Document Signature Field, Image Field, List Box, Numeric Field, Password Field, Radio Button, and Text Field. • `draw` Returns all of the following form design objects: Circle, Line, Rectangle, Static Image, and Static Text. • `subform` Returns all subform form design objects. • `area` Returns all area form design objects. • `pageArea` Returns all pageArea form design objects. • `contentArea` Returns all contentArea form design objects. • `empty` (default) Returns all containers.
param3 (Optional)	• `True` (default) Returns only pageArea content nodes. • `False` Returns all non-pageArea content nodes.

Returns

A collection of form design objects from the specified page number.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
// Get the fields in a document
```

```
var oFields = xfa.layout.pageContent(i, "field");
```

FormCalc

```
// Get the fields in a document
var oFields = xfa.layout.pageContent(i, "field")
```

See also

- "Referencing objects" on page 420
- "Disabling all form fields" on page 434

pageCount

Determines the number of pages of the current form.

Syntax

Reference_Syntax.pageCount()

Parameters

None

Returns

An integer representing the total number of pages of the form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.pageCount();
```

FormCalc

```
xfa.layout.pageCount()
```

See also

- "Referencing objects" on page 420
- "Working with page numbers and page counts" on page 426
- "Disabling all form fields" on page 434

pageDown

Moves to the next page of a form. Use the pageDown method at run time.

Syntax
Reference_Syntax.pageDown()

Parameters

None

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.host.pageDown();
```

FormCalc
```
xfa.host.pageDown()
```

See also

"Working with page numbers and page counts" on page 426

pageSpan

Determines the number of logical pages a given form design object spans.

Syntax
Reference_Syntax.pageSpan(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following a container form design objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the total number of pages of the form.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.layout.pageSpan(this);
```

FormCalc
```
xfa.layout.pageSpan($)
```

pageUp

Moves to the previous page of a form. Use the pageUp method at run time.

Syntax
Reference_Syntax.pageUp()

Parameters

None

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.host.pageUp();
```

FormCalc
```
xfa.host.pageUp()
```

See also

"Working with page numbers and page counts" on page 426

previous

Moves to the previous record in the record set, and populates the data model with the record data.

Note: The data connection method `xfa.sourceSet.DataConnection.previous` looks up a table and updates the table if the data has changed. It uses the hasDataChanged method to determine whether the data has changed.

Syntax
Reference_Syntax.previous()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.previous();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.previous()
```

print

Prints a specific number of pages from a document. It is available only for client applications.

Syntax

```
Reference_Syntax.print( BOOLEAN param1, INTEGER param2, INTEGER param3,
BOOLEAN param4, BOOLEAN param5, BOOLEAN param6, BOOLEAN param7, BOOLEAN param8
)
```

Parameters

param1	• `True` (default) Displays a print dialog box and prompts the user for printing setup information and confirmation of the action. • `False` Does not display a print dialog box. Printing proceeds without prompting the user for information or confirmation.
param2	A valid string representing the page number of the beginning of the range to print. Page values are 0-based, so you represent page 1 with a value of `0`. The start page is included in the printing.
param3	A valid string representing the page number of the end of the range to print. Page values are 0-based, so you represent page 1 with a value of `0`. The end page is included in the printing.

param4	• `True` (default) Does not display a cancel dialog box during the printing process. • `False` Displays a cancel dialog box to stop the printing process.
param5	• `True` (default) Shrinks the page (if necessary) to fit within the imageable area of the printed page. • `False` Does not shrink the page to fit within the imageable area of the printed page.
param6	• `True` (default) Prints each page as an image. • `False` Prints each page as a page of text.
param7	• `True` (default) Prints the pages in reverse order. • `False` Prints the pages in order.
param8	• `True` (default) Prints all annotations. • `False` Does not print annotations.

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version
XFA 2.1

Examples

JavaScript
```
xfa.host.print(1, "0", "0", 0, 1, 0, 0, 0);
```

FormCalc
```
xfa.host.print(1, "0", "0", 0, 1, 0, 0, 0)
```

recalculate

Forces a specific set of scripts located on calculate events to execute. The specific events can be either pending calculate events or all calculate events.

Syntax

Reference_Syntax.recalculate(BOOLEAN *param*)

Parameters

param	A Boolean value indicating which calculation scripts should execute. • `True` (default) All calculation scripts are re-executed. • `False` Only pending calculation scripts should execute.

Returns

Empty

Applies to

Model	Object
Form Model	form template

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.recalculate(1);
```

FormCalc

```
xfa.form.recalculate(1)
```

record

Returns a record in a position relative to the current record.

Syntax

Reference_Syntax.record(INTEGER *param*)

Parameters

param	A valid integer representing the number of records separating the current record and the desired destination record. A positive integer indicates a record between the current record and the end of the range of records, a negative value indicates a record between the current record and the beginning of the range.

Returns
Object

Applies to

Model	Object
Data Model	dataWindow

Version

XFA 2.1

Examples

JavaScript
```
xfa.dataWindow.record(0).dataNodeName.value;
```

FormCalc
```
xfa.dataWindow.record(0).dataNodeName.value
```

See also

- "Creating a node in the data model" on page 422
- "Concatenating data values" on page 427
- "Populating a drop-down list" on page 430

relayout

Reapplies the layout options to the current form.

Syntax
Reference_Syntax.relayout()

Parameters

None

Returns
Empty

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.layout.relayout();
```

FormCalc
```
xfa.layout.relayout()
```

relayoutPageArea

Replaces the layout of the pageArea object content with a new layout.

Syntax
Reference_Syntax.relayoutPageArea([INTEGER *param*])

Parameters

param (Optional)	The page number of the page to substitute. Page number values are 0 based.

Returns
Empty

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript
```
xfa.layout.relayoutPageArea(0);
```

FormCalc
```
xfa.layout.relayoutPageArea(0)
```

remerge

Forces the remerging of the data model and template model to re-create the form model. After the remerge is complete, any layout model processing must be redone if necessary for the completed form.

Syntax
Reference_Syntax.remerge()

Parameters

None

Returns

Empty

Applies to

Model	Object
Form Model	form

Version

XFA 2.1

Examples

JavaScript

```
xfa.form.remerge();
```

FormCalc

```
xfa.form.remerge()
```

remove

Removes a node from the node list.

Syntax

Reference_Syntax.remove(OBJECT *param*)

Parameters

param	A valid reference syntax expression representing the node to be removed.

Returns

Empty

Applies to

list class

Version

XFA 2.1

Examples

JavaScript

```
xfa.record.nodes.remove(oNode);
```

FormCalc

```
xfa.record.nodes.remove(oNode)
```

See also

"Creating a node in the data model" on page 422

removeAttribute

Removes the specified property.

Syntax

Reference_Syntax.removeAttribute(STRING *param*)

Parameters

param	A valid string representing the name of the property to remove.

Returns

Empty

Applies to

Model	Object
XFA Model	packet

Version

XFA 2.1

removeInstance

Removes a specified subform or subform set from the form model.

When removing a subform instance, avoid subform occurrence violations. You cannot remove a subform instance if it has reached the minimum number of instances. When a subform reaches the minimum number of instances, the JavaScript debugger displays an error message:

```
The element [min] has violated its allowable number of occurrences.
```

If the end user is allowed to remove every instance of a subform, reset the minimum number of instances to `0` before attempting to remove an instance. Otherwise, the script should prevent any attempt to remove subform instances beyond the minimum number.

Syntax

Reference_Syntax.removeInstance(INTEGER *param*)

Parameters

param	A valid integer representing the 0 based index position within the form model of the subform or subform set to remove.

Returns

Empty

Applies to

Model	Object
Form Model	instanceManager

Version

XFA 2.1

Examples

JavaScript

```
Subform2.instanceManager.removeInstance(3);
```

FormCalc

```
Subform2.instanceManager.removeInstance(3)
```

See also

"Manipulating instances of a subform" on page 424

requery

Updates the current data binding by re-executing the query on which the object data is based. Calling this method is equivalent to calling the close and open methods in succession.

Syntax

Reference_Syntax.requery()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

reset

Resets all of the properties within the XML form event model.

Syntax

Reference_Syntax.reset()

Parameters

None

Returns

Empty

Applies to

Model	Object
Event Model	eventPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.event.reset();
```

FormCalc

```
xfa.event.reset()
```

resetData

Resets the fields to their default values within a document.

Syntax

Reference_Syntax.resetData([STRING *param*])

Parameters

param (Optional)	A valid string listing either the names or the equivalent reference syntax expressions of the fields to reset. The list entries are delimited by the "," (comma) character. If the string is not present or empty, all the fields in the form are reset to their default value.

Returns

Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.resetData("xfa.form.form1.TextField1,xfa.form.form1.TextField2");
```

FormCalc

```
xfa.host.resetData("xfa.form.form1.TextField1,xfa.form.form1.TextField2")
```

resolveNode

Evaluates the specified reference syntax expression, beginning with the current XML form object model object, and returns the value of the object specified in the reference syntax expression.

The search for an object starts at a different point in the form hierarchy, depending on how the resolveNode property was accessed:

- `this.resolveNode()` The search starts from the current object and moves up the form hierarchy.
- `xfa.resolveNode()` The search starts at the top of the form hierarchy and moves down.

Note: The search could return unexpected results if the form contains several objects that use the same name. It returns the value of the first object that it finds.

Syntax

Reference_Syntax`.resolveNode( STRING` *param* `)`

Parameters

param	A valid string representing a reference syntax expression that evaluates to a specific XML form object model object.

Returns

A single object corresponding to the reference syntax expression, if it exists. If no such object exists, this method returns `null`.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
xfa.resolveNode("#subform").x = "2in";

TextField1.resolveNode("ui.#textEdit").border.edge.stroke = "lowered";
```

See also

- "Referencing objects" on page 420
- "Creating a node in the data model" on page 422
- "Manipulating instances of a subform" on page 424
- "Populating a drop-down list" on page 430

resolveNodes

Evaluates the specified reference syntax expression, beginning with the current XML form object model object, and returns the value of the object or objects specified in the reference syntax expression.

The search for an object starts at a different point in the form hierarchy, depending on how the resolveNode property was accessed:

- `this.resolveNodes()` The search starts from the current object and moves up the form hierarchy.
- `xfa.resolveNodes()` The search starts at the top of the form hierarchy and moves down.

Note: The search could return unexpected results if the form contains several objects that use the same name. It returns the value of the first object that it finds.

Syntax

Reference_Syntax`.resolveNodes( STRING` *param* `)`

Parameters

param	A valid string representing a reference syntax expression that evaluates to one or many XML form object model objects.

Returns

An object or multiple objects corresponding to the reference syntax expression, if such objects exist. If no such objects exist, this method returns `null`.

Applies to

tree class

Version

XFA 2.1

Examples

JavaScript

```
xfa.resolveNodes("Subform1[*]");
```

FormCalc

```
xfa.resolveNodes("Subform1[*]")
```

See also

- "Referencing objects" on page 420
- "Concatenating data values" on page 427
- "Using radio buttons and check boxes" on page 433

response

Displays a dialog box containing a question and an entry field for the user to reply to the question. The return value is a string containing the user's response. If the user presses the cancel button on the dialog box, the response is null.

Syntax

Reference_Syntax`.response(STRING` *param1* `[, STRING` *param2* `[, STRING` *param3* `[, BOOLEAN` *param4* `] ] ])`

Parameters

param1	A valid string representing a question for the user.
param2 (Optional)	A valid string representing the title that appears in the title bar of the dialog box.
param3 (Optional)	A valid string representing the default value for the answer to the question.
param4 (Optional)	• `True` (default) Masks the user's answer with * (asterisks). • `False` Does not mask the user's answer.

Returns

A string representing the user's answer. If the user presses the cancel button on the dialog box, the answer is the null object.

Applies to

Model	Object
Host Model	hostPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.host.response("Question", "Title", "Default Value");
```

FormCalc

```
xfa.host.response("Question", "Title", "Default Value")
```

restore

Updates the property's current value with the saved value.

The script should perform any required validations prior to calling the `restore` property.

Syntax

Reference_Syntax`.restore()`

Parameters

None

Returns

Null

Applies to

Model	Object
Form Model	delta

Version

XFA 2.5

resync

Refreshes the current record set or data connection.

Syntax

Reference_Syntax.resync()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

saveFilteredXML

Saves the current node to a string, but includes only a subset of the child nodes.

Syntax

Reference_Syntax.saveFilteredXML(OBJECT *param1* [, STRING *param2*])

Parameters

param1	A manifest object that describes the subset of nodes to include in the string.
param2 (Optional)	A valid string representing how to print the resulting XML string. For example, use the string `pretty` to pretty-print the resulting XML string.

Returns

A valid string representing the XML fragment that includes only the specified subsetof the current node.

Applies to

node class

Version

XFA 2.4

saveXML

Saves the current node to a string.

Syntax

Reference_Syntax.saveXML()

Parameters

None

Returns

A valid string representing the XML fragment of the current object.

Applies to

node class

Version

XFA 2.1

Examples

JavaScript

```
xfa.data.saveXML();
```

FormCalc

```
xfa.data.saveXML()
```

See also

"Determining that a form has changed" on page 433

selectedMember

Returns the selected member of an exclusion group.

Syntax

Reference_Syntax.selectedMember([STRING *param*])

Parameters

param (Optional)	A valid string representing the name of the exclusion group member, provided the exclusion group member is within the same scope as the referencing object. Otherwise, a valid string representing the reference syntax expression of the exclusion group member to select.

Returns

The object representing the selected member of the exclusion group. In LiveCycle Designer ES, for example, this method would return the selected radio button.

Applies to

Model	Object
Form Model	exclGroup

Version

XFA 2.1

setAttribute

Sets the value of a specified property.

Syntax

```
Reference_Syntax.setAttribute( STRING param1, STRING param2 )
```

Parameters

param1	A valid string representing the new value of the property.
param2	A valid string representing the name of the property.

Returns

Empty

Applies to

Model	Object
XFA Model	packet

Also applies to the node class.

Version

XFA 2.1

setElement

Sets a specified object to be the current object.

Syntax

```
Reference_Syntax.setElement( OBJECT param1 [, STRING param2 ] )
```

Parameters

param1	An object representing the new object.
param2 (Optional)	A valid string representing the name of the object to replace.

Returns
Empty

Applies to
node class

Version
XFA 2.1

setFocus

Sets the keyboard focus to the form object specified by the reference syntax expression.

It is available only for client applications.

When the *param1* argument is omitted or null, `setFocus` performs a clear focus operation. If any form object has the input focus, the focus is removed from that object and any pending edits in that object are committed, dirtying the document if appropriate. If committing the changes causes a validation error, that error is displayed. If no form object has the input focus, the zero-argument `setFocus` does nothing.

You cannot use `setFocus` with the `form:ready`, `layout:ready`, or `initialize` events.

Syntax
```
Reference_Syntax.setFocus( OBJECT param )
Reference_Syntax.setFocus( STRING param ) (deprecated)
```

Parameters

param	(Optional) A valid string representing a fully qualified reference syntax expression for the form object.

Returns
Empty

Applies to

Model	Object
Host Model	hostPseudoModel

Version
XFA 2.6

XFA 2. 1 (deprecated)

Examples

JavaScript

```
xfa.host.setFocus(xfa.form.form1.TextField1);
```

FormCalc

```
xfa.host.setFocus(xfa.form.form1.TextField1)
```

setInstances

Adds or removes specified subforms or subform sets from the form model.

Syntax

Reference_Syntax.setInstance(INTEGER *param*)

Parameters

param	A valid integer representing the desired number of instances of a particular subform or subform set in the form model.

Returns

Empty

Applies to

Model	Object
Form Model	instanceManager

Version

XFA 2.1

Examples

JavaScript

```
Subform1.instanceManager.setInstances(5);
```

FormCalc

```
Subform1.instanceManager.setInstances(5)
```

See also

"Manipulating instances of a subform" on page 424

setItemState

Sets the selection state of the specified item.

Syntax

Reference_Syntax.setItemState(INTEGER *param1*, BOOL *param2*)

Parameters

param1	A valid integer representing the zero-based index into the item.
param2	• True Adds this item to the current selection. • False Removes this item from the current selection.

Returns

None

Applies to

Model	Object
Form Model	field

Version

XFA 2.5

sheet

Determines the sheet number that contains the form object.

Some duplex documents use sheet numbers to number only the front surfaces. For example, you can use sheet numbers when the front surfaces contain variable data and the back surfaces contain boilerplate text, such as instructions, disclaimers, or legends.

Syntax

Reference_Syntax.sheet(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following form objects: field, draw, subform, area, pageArea, contentArea.

Returns

A zero-based integer representing the sheet number.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.5

sheetCount

Determines the number of sheets in the current form.

Syntax

Reference_Syntax.sheetCount( )

Parameters

None

Returns

An integer representing the total number of sheets.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.5

sheetCountInBatch

Determines the sheet count of the current batch.

Syntax

Reference_Syntax.sheetCountInBatch( )

Parameters

None

Returns

An integer representing the sheet count of the current batch.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.5

sheetInBatch

Determines which sheet within the batch contains the form object.

Syntax

Reference_Syntax.sheetInBatch(OBJECT *param*)

Parameters

param	The fully qualified reference syntax expression of one of the following form objects: field, draw, subform, area, pageArea, contentArea.

Returns

An integer representing the sheet number that contains the form object.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.5

sign

Signs a given node list and places the signature in the target location.

Syntax

```
Reference_Syntax.sign( OBJECT param1, STRING param2, STRING param3 [, STRING
param4 [, BOOLEAN param5 [, OBJECT param6 [, OBJECT param7 ] ] ] ] )
```

Parameters

param1	input	A valid XFA node list of all the nodes to be signed.
param2	input	A valid string representing a reference syntax expression to the parent of the signature node.
param3	input	A valid string representing an XML identification value for the signature.
param4 (Optional)	input (Optional)	A valid string indicating the behavior of the signed data nodes. The values are: • `open` (default) The data nodes are open for edit and can be manipulated at runtime. • `locked` The data node is locked and cannot be manipulated at runtime.
param5 (Optional)	input (Optional)	• `True` (default) Indicates that a dialog is used for this operation. • `False` Indicates that a dialog is not used for this operation.

param6 (Optional)	input (Optional)	Represents the SecurityHandler object that is used to sign. Security objects normally require initialization before they can be used for signing.
param7 (Optional)	output (Optional)	Represents an output SignatureInfo object containing the writable properties of the signature.

Returns

`True` if the signature was applied successfully and `False` if the signing option was canceled. An exception is returned if the signing operation fails.

Applies to

Model	Object
Signature Model	signaturePseudoModel

Version

XFA 2.1

update

Updates the current record in the record set.

Syntax

`Reference_Syntax``.update()`

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

Examples

JavaScript

```
xfa.sourceSet.dataConnectionName.update();
```

FormCalc

```
xfa.sourceSet.dataConnectionName.update()
```

updateBatch

Writes all pending batch updates to the data source.

Syntax

Reference_Syntax.updateBatch()

Parameters

None

Returns

Empty

Applies to

Model	Object
sourceSet Model	source

Version

XFA 2.1

verify

Checks the validity of a signature.

Syntax

Reference_Syntax.verify(OBJECT *param1* [, BOOLEAN *param2* [, OBJECT *param3* [, OBJECT *param4*]]])

Parameters

param1	input	A valid XML signature node.
param2 (Optional)	input (Optional)	• True (default) Indicates that a dialog box is used for this operation. • False Indicates that a dialog box is not used for this operation.
param3 (Optional)	input (Optional)	The SecurityHandler object that is used to sign. Security objects normally require initialization before they can be used for signing.
param4 (Optional)	output (Optional)	An output SignatureInfo object containing the writable properties of the signature.

Returns

An integer representing the validity of the signature. The following table describes the validity values:

Value	Description
-1	Not a signature node.
0	Signature is blank.
1	Unknown status.
2	Signature is invalid.
3	Signature is valid, but the identity of the signer could not be verified.
4	Signature is valid and the identity of the signer is valid.

Applies to

Model	Object
Signature Model	signaturePseudoModel

Version

XFA 2.1

w

Determines the width of a given form design object.

Syntax

```
Reference_Syntax.w( OBJECT param1 [, STRING param2 [, INTEGER param3 ] ] )
```

Parameters

param1	The fully qualified reference syntax expression of one of the following container XML Form Object Model objects: area, contentArea, draw, field, pageArea, subform.
param2 (Optional)	A string representing the unit type of the return value. If left blank, the default unit type is points.
param3 (Optional)	An integer representing the number on which to adjust the width of the object, beginning with the first page the object occurs on. If left blank, the default value is `0`.

Returns

The width of the form design object as a double.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.w(TextField1,"in");
```

FormCalc

```
xfa.layout.w(TextField1,"in")
```

x

Determines the x coordinate of a given form design object.

Syntax

Reference_Syntax.x(OBJECT *param1* [, STRING *param2* [, INTEGER *param3*]])

Parameters

param1	The fully qualified reference syntax expression of one of the following container XML Form Object Model objects: area, contentArea, draw, field, pageArea, subform.
param2 (Optional)	A string representing the unit type of the return value. If left blank, the default unit type is points.
param3 (Optional)	An integer representing the number of pages to offset the x coordinate of the object, beginning with the first page the object occurs on. If left blank, the default value is 0.

Returns

The x coordinate of the form design object as a double.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.x(TextField1,"in");
```

FormCalc

```
xfa.layout.x(TextField1,"in")
```

y

Determines the y coordinate of a given form design object.

Syntax

Reference_Syntax.y(OBJECT *param1* [, STRING *param2* [, INTEGER *param3*]])

Parameters

param1	The fully qualified reference syntax expression of one of the following container XML Form Object Model objects: area, contentArea, draw, field, pageArea, subform.
param2 (Optional)	A string representing the unit type of the return value. If left blank, the default unit type is points.
param3 (Optional)	An integer representing the number of pages to offset the y coordinate of the object, beginning with the first page the object occurs on. If left blank, the default value is `0`.

Returns

The y coordinate of the form design object as a double.

Applies to

Model	Object
Layout Model	layoutPseudoModel

Version

XFA 2.1

Examples

JavaScript

```
xfa.layout.y(TextField1,"in");
```

FormCalc

```
xfa.layout.y(TextField1,"in")
```

6 Understanding the XML Form Object Model

A DOM is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. DOMs are commonly used with data expressed in XML.

All of the DOMs used in the XML Form Object Model share the following characteristics:

- They are strictly tree-structured.
- A node may have mandatory children. In such cases, the mandatory child nodes are created at the same time as their parent.
- The non-mandatory children of each node in the tree are ordered by age. That is, the DOM is aware of the order in which the non-mandatory child nodes were added.

For each step in the form processing, there is a DOM that holds the data structures for that stage. Scripts can examine and modify each DOM. DOMs are responsible for maintaining internal consistency but not external consistency. For instance, when a script turns on a radio button by assigning the corresponding field, all the other buttons coupled to that one are automatically turned off. This is a matter of internal consistency so it is managed by the Form DOM itself.

By contrast, the Data DOM does nothing to prevent a script from violating the rules of XML, for instance, by giving an object two properties with the same name. This is a matter of external consistency so it is the responsibility of the script author, not the DOM.

Each time a form design is combined with data, the XML Form Object Model is used to facilitate the process of combining template and data to create the resulting form. This process begins by using the existing XML DOMs' representations of the form design and the XML data to create separate models. These separate models store a structured representation of the original form design and original XML data. The Template DOM corresponds to the form design, and the Data DOM corresponds to the user-supplied XML data.

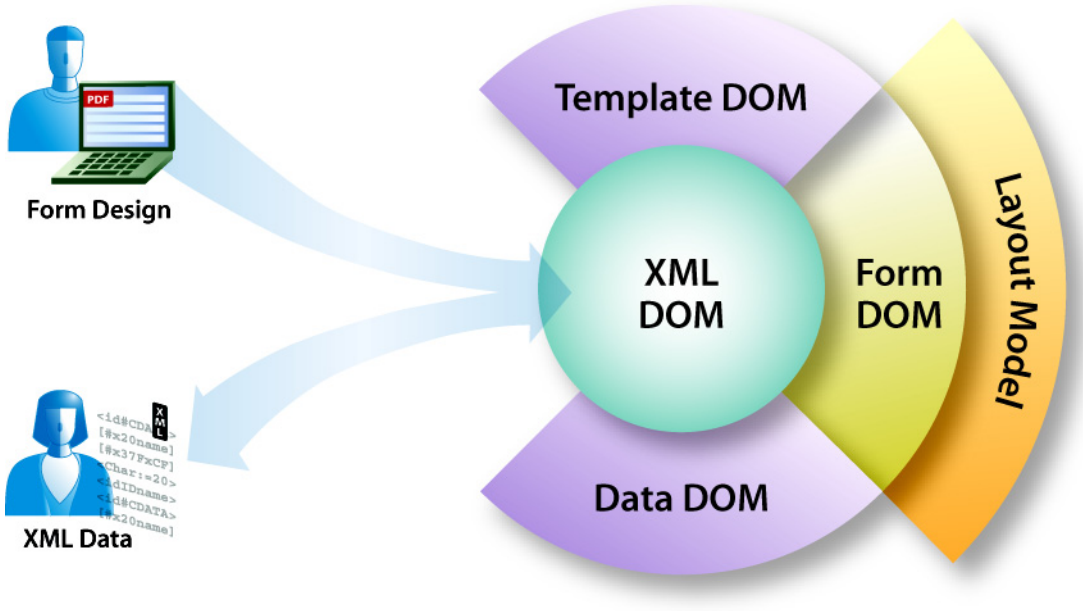
After the template and data models are created, a third model, the Form DOM, is created that represents the merged information. The Form DOM acts as a medium for combining the specific values from the XML data with the presentation rules defined by the form design.

If you are creating an interactive form, after the form DOM is created, the form is complete and ready for deployment to users. Interactive form designs may have associated data that they are merged with, but most interactive forms are designed to support user-entered data.

The process up to and including the creation of the form DOM is identical for all forms. However, non-interactive forms have a set of data to merge with their form design. In the case of forms that have a fixed layout, data merging does not determine the presentation rules for the form; that is, data is merged into the appropriate fields without changing the field properties. In contrast, when data is merged with forms that have a flowable layout, the fields grow or shrink to accommodate the amount of data merged into them.

The Form DOM for forms with both fixed and flowable layouts looks very similar; it is one long form with no pagination. When the data and presentation rules are applied to these types of forms, they must be formatted according to the layout information. A Layout DOM is created from the Form DOM that

structures the form into pages and applies any other page-based rules, such as page numbering, headers, and trailers. The following diagram illustrates this process.



After the layout rules are applied to forms that have a fixed or flowable layout, both types of forms are complete.

XML Form Object Model DOMs

connectionSet Model

The connectionSet model controls a data schema as well as a data source used by a particular form. This model describes connections to XML schema, sample XML data, or web services. Using the connectionSet model, it is possible to extract the details, such as a URL, for a referenced schema or WSDL for reporting purposes.

The connectionSet model consists of the following objects:

- "connectionSet" on page 44
- "operation" on page 96
- "rootElement" on page 109
- "soapAction" on page 115
- "soapAddress" on page 116
- "uri" on page 131
- "wsdlAddress" on page 135
- "wsdlConnection" on page 135
- "xmlConnection" on page 136
- "xsdConnection" on page 137

Data Model

The Data model is the in-memory representation of user data. When a form design and data are merged using the data-binding process, the data model supplies the content for fields on the final form.

Using this model, you can access and manipulate data from one of the following data sources:

- XML document
- OLEDB database
- XML schema file
- WSDL file

The Data model consists of the following objects:

- "dataGroup" on page 47
- "dataModel" on page 47
- "dataValue" on page 48
- "dataWindow" on page 48

Event Model

The Event model controls the changes in a form that occur before, during, and after actions take place. These actions include dynamic form events, such as the point when the data and form design are merged but before any pagination is applied, as well as interactive form events such as when a user updates the value of a field.

The Event model consists of the following object:

- "eventPseudoModel" on page 62

Form Model

The Form model is the in-memory representation of the merged Template model and Data model. Using this model, you can affect the look of the form, adjust field values, or perform other changes prior to either displaying the completed form to a user or processing the form through the Layout model.

Scripts run against the Form model by default; therefore, you do not need to specify the Form model in your reference syntax.

The Form model consists of the following objects:

"arc" on page 25
"area" on page 26
"assist" on page 26
"barcode" on page 27
"bind" on page 29
"bindItems" on page 30
"bookend" on page 30
"boolean" on page 31
"border" on page 32
"break" on page 33
"breakAfter" on page 34
"breakBefore" on page 35
"button" on page 36
"calculate" on page 37
"caption" on page 37
"certificate" on page 38
"certificates" on page 39
"checkButton" on page 40
"choiceList" on page 40
"color" on page 41
"comb" on page 42
"connect" on page 43
"contentArea" on page 45
"corner" on page 46
"date" on page 49
"dateTime" on page 50
"dateTime" on page 50
"dateTimeEdit" on page 51
"decimal" on page 51
"defaultUi" on page 52
"delta" on page 53
"deltas" on page 54
"desc" on page 54
"digestMethod" on page 55
"digestMethods" on page 56
"draw" on page 56
"dSigData" on page 58
"edge" on page 59
"encoding" on page 59
"encodings" on page 60
"encrypt" on page 60
"event" on page 61
"exclGroup" on page 63
"exData" on page 66
"execute" on page 67
"exObject" on page 68
"extras" on page 68
"field" on page 69
"fill" on page 72
"filter" on page 73
"float" on page 74
"font" on page 75
"form" on page 76
"format" on page 77
"handler" on page 77
"image" on page 80
"imageEdit" on page 81
"instanceManager" on page 82
"integer" on page 83
"issuers" on page 83
"items" on page 84
"keep" on page 85
"keyUsage" on page 85
"line" on page 88
"linear" on page 88
"manifest" on page 89
"margin" on page 90
"mdp" on page 91
"medium" on page 92
"message" on page 93
"numericEdit" on page 93
"occur" on page 94
"oid" on page 95
"oids" on page 96
"overflow" on page 97
"pageArea" on page 98
"pageSet" on page 99
"para" on page 100
"passwordEdit" on page 102
"pattern" on page 102
"picture" on page 103
"proto" on page 104
"radial" on page 105
"reason" on page 106
"reasons" on page 106
"rectangle" on page 108
"ref" on page 108
"script" on page 110
"setProperty" on page 111
"signature" on page 112
"signatureProperties (deprecated)" on page 113
"signData" on page 114
"signing" on page 114
"solid" on page 116
"speak" on page 118
"stipple" on page 119
"subform" on page 120
"subformSet" on page 122
"subjectDN" on page 123
"subjectDNs" on page 123
"submit" on page 124
"template" on page 125
"text" on page 125
"textEdit" on page 126
"time" on page 127
"timeStamp" on page 128
"toolTip" on page 128
"traversal" on page 129
"traverse" on page 130
"ui" on page 130
"validate" on page 132
"value" on page 133
"variables" on page 134

Host Model

The Host model provides a set of properties and methods for working at the application level. These properties and methods are available for scripting regardless of the hosting application.

The Host model consists of the following object:

- "hostPseudoModel" on page 78

Layout Model

The Layout model is the in-memory representation of a form after it is merged with data. This representation is the final layout of a form.

The Layout model consists of the following object:

- "layoutPseudoModel" on page 87

Signature Model

The Signature model provides a set of methods for working with XML digital signatures that conform to the W3C XML-Signature standard (http://www.w3.org/TR/xmldsig-core/). It lets you specify script commands to sign, clear, enumerate, and verify signatures.

The Signature model consists of the following object:

- "signaturePseudoModel" on page 113

sourceSet Model

The sourceSet model provides a connection between an external OLEDB database and the Data model. Using this model, you can control connections to the data source, as well as manage records within the data source.

The sourceSet model consists of the following objects:

- "bind" on page 29
- "boolean" on page 31
- "command" on page 43
- "connect" on page 43
- "connectString" on page 44
- "delete" on page 53
- "extras" on page 68
- "insert" on page 81
- "integer" on page 83
- "map" on page 90
- "password" on page 101
- "query" on page 105
- "recordSet" on page 107
- "select" on page 111
- "source" on page 117
- "sourceSet" on page 118
- "text" on page 125
- "update" on page 131
- "user" on page 132

XFA Model

The XFA model defines the application model that LiveCycle Designer ES uses to implement the XML Form Object Model. The application model is the base model from which all other models are derived.

The XFA model consists of the following objects:

- "packet" on page 98
- "xfa" on page 136

A JavaScript Examples

This section provides illustrative examples of properties and methods that are supported in this scripting environment.

Referencing objects

These examples illustrate several ways to reference an object.

When accessing a specific instance of an object, be aware of the occurrence number of the object where the script resides. The script will return the object with the same occurrence number as the object where the script resides. For example, there are three buttons with the same name (Button1[0], Button1[1] and Button1[2]) and three text fields with the same name (TF1[0], TF1[1] and TF1[2]). If the script on Button1[2] is `xfa.host.messageBox(TF1.rawValue)`, the result will be `TF1[2].rawValue`, and not `TF1[0].rawValue`.

See also

- For an example that illustrates how to access a data model value, see "Setting a data object's value" on page 426.
- For an example that illustrates how to access a field in a repeating subform by looping through the node list, see "Calculating totals" on page 428.

Uses

Properties		Methods
access index layout length name newText numPages	oneOfChild parent prevText rawValue target this	item resolveNode resolveNodes pageContent pageCount

Scripts

Accessing the first instance of a text field

```
// Access a sibling field using the field name.
// Access the first instance of TextField1.
TextField1.rawValue = "Hello";
```

Accessing the first instance of a text field

```
// Access the first instance of TextField1. When scripting with JavaScript, use
// xfa.resolveNode to start the search at the top and move down the form
// hierarchy.
xfa.resolveNode("TextField1").rawValue = "Hello";
xfa.resolveNode("TextField1[0]").rawValue = "Hello";
```

Accessing a field with accessors

```
// When scripting with JavaScript, use the resolveNode() method to access a
```

```
// field with a SOM expression that contains a # or [] operator. When searching
// with this.resolveNode, the search starts at the current object and moves up
// the form hierarchy.
this.resolveNode("Subform2[1].NumericField4").rawValue = 25;
```

Accessing a subform with an index number

```
// Access a subform with an index number. When using xfa.resolveNode,the search
// starts at the top of the form hierarchy and moves down.
var nIndex = 2;
var sSOM = "Subform2[" + nIndex + "]";
var oSubform = xfa.resolveNode(sSOM);
oSubform.NumericField4.rawValue = "25";
```

Accessing a field property

```
// Access a field property using a property name and value.
// Change the field properties of a specific subform.
// Use the [] operator to access an object's property.
var sProperty = "access";
var sValue = "readOnly";

// First, get the subform nodes.
   var oNodes = Subform2.nodes;
   var nNodesLength = oNodes.length;

   // Loop through the subform's nodes and look for fields.
   for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount ++) {
      // Set the field property.
      if (oNodes.item(nNodeCount).className == "field") {
         oNodes.item(nNodeCount)[sProperty] = sValue;
      }
   }
```

Counting the text fields in a document

```
// Count the number of text fields in a document.
// Get the field containers from each page.
for (var nPageCount = 0; nPageCount < xfa.host.numPages; nPageCount++) {

   var oFields = xfa.layout.pageContent(nPageCount, "field");
   var nNodesLength = oFields.length;
   var nCount = 0;

   for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {

      if (oFields.item(nNodeCount).ui.oneOfChild.className == "textEdit") {
         nCount++;
      }
   }
   TextField1.rawValue = nCount;
}
```

Accessing fields using partial object names

```
// Access fields using partial object names.
// Get the field containers from each page.
for (var nPageCount = 0; nPageCount < xfa.host.numPages; nPageCount++) {

   var oFields = xfa.layout.pageContent(nPageCount, "field");
```

```
   var nNodesLength = oFields.length;

   for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {
      if (oFields.item(nNodeCount).name.substr(0,2) == "Te") {
         xfa.host.messageBox(oFields.item(nNodeCount).name);
      }
   }
}
```

Accessing a choice list value

```
// Use the newText or prevText property to access the choice list value before
// or after the value changed.
// Trigger the script on a change event.
TextField1.rawValue = xfa.event.prevText;
TextField2.rawValue = xfa.event.newText;
```

Accessing a field in a subform

```
// Access a field nested inside a sibling subform by prefixing the field name
// with its parent name.
Subform2.TextField3.rawValue = "Hello";
```

Accessing fields in a subform

```
// Access the first-level fields nested inside a subform.
Subform1.resoveNodes("#field[*]")
```

Getting the fields from each page

```
// Get the field containers from each page.
for (var i = 0; i < xfa.host.numPages; i++) {

var oFields = xfa.layout.pageContent(i, "field");
var nodesLength = oFields.length;

// Set the access type.
for (var j = 0; j < nodesLength; j++) {

     var oItem = oFields.item(j);

     if (oItem != this) {

     oItem.access = "readOnly";
     }
   }
}
```

Creating a node in the data model

This example illustrates how to create or clone a new data model node.

Uses

Properties	Methods	
length nodes rawValue value	append clone createNode messageBox	record remove resolveNode

Script

Creating a data node

```
// Display the number of child nodes under the rootNode (xfa.record).
// rootNode is the data file's root node.
xfa.host.messageBox("Initial number of nodes under rootNode: " +
xfa.record.nodes.length);
// Create a node of type dataGroup.
var oGroupNode = xfa.datasets.createNode("dataGroup", "NewGroupNode");

// Append the data group node to an existing data model node.
xfa.record.nodes.append(oGroupNode);

// Display the number of child nodes under rootNode.
xfa.host.messageBox("Number of nodes under rootNode after first append: " +
xfa.record.nodes.length);

// Create a node of type dataValue.
var oValueNode = xfa.datasets.createNode("dataValue", "NewValueNode");

// Set the value of the new data value node.
oValueNode.value = "The item value";

// Append the data value node to the data group created above.
xfa.record.NewGroupNode.nodes.append(oValueNode);

// Get the value from the data model.
TextField1.rawValue = xfa.record.NewGroupNode.NewValueNode.value;

// Append a cloned data group node.
xfa.record.nodes.append(xfa.record.NewGroupNode.clone(1));

// Display the number of child nodes under rootNode.
xfa.host.messageBox("Number of nodes under rootNode after appending clone: " +
xfa.record.nodes.length);

// Set the value of the new data value node.
xfa.resolveNode("xfa.record.NewGroupNode[1].NewValueNode").value = "The clone
value";

// Get the value of the cloned data value node.
TextField2.rawValue =
xfa.resolveNode("xfa.record.NewGroupNode[1].NewValueNode").value;

// Remove the cloned data group from the node list.
var oRemoveNode = xfa.resolveNode("xfa.record.NewGroupNode[1]");
```

```
xfa.record.nodes.remove(oRemoveNode);

// Display the number of child nodes under rootNode.
xfa.host.messageBox("Number of nodes under rootNode once clone node removed: "
+ xfa.record.nodes.length);
```

Manipulating instances of a subform

These examples illustrate several ways to add or remove instances of a subform at run time.

Use the instance manager to manipulate the pages of a form that has a fixed layout. Each page is a subform; therefore, adding or removing a subform will look like adding or removing a page. However, at run time, you cannot change the layout of a form that has a fixed layout. You can add and delete instances at the `form:ready` event; however, if the script is on a run-time event, such as `click`, nothing will happen.

Uses

Properties	Methods	
min index parent value	addInstance moveInstance removeInstance	resolveNode setInstances

Scripts

Adding an instance by invoking the instance manager

```
// Add an instance of a subform by using the underscore syntax to invoke the
// instance manager directly.
// Forms rendered in a web browser do not support the underscore syntax.
// However, the underscore syntax is supported if the script runs at the
// server.
_Subform2.addInstance(1);
```

Adding an instance by invoking the instanceManager property

```
// Add an instance of a subform by invoking the instanceManager property. Be
// careful to ensure that adding a subform will not violate the max occur
// value.
Subform2.instanceManager.addInstance(1);
```

Removing an instance

```
// Remove an instance of a subform. Set the min occur value only if removing an
// instance will violate it. For example, set the min occur to 0 if you want to
// remove the last, or the only, instance of a subform.
// Forms rendered in a web browser do not support the underscore syntax.
// However, the underscore syntax is supported if the script runs at the
// server.
Subform2.occur.min = "0";
_Subform2.removeInstance(0);
```

Removing the parent subform

```
// Remove the parent subform.
parent.occur.min = "0";
parent.instanceManager.removeInstance(parent.index);
```

Setting the number of instances

```
// Set the number of instances of a subform.
var oSubform = xfa.resolveNode("Subform2");
oSubform.instanceManager.setInstances(5);
```

Inserting a new subform instance

```
// Insert a new subform instance. This script will not work with a static form.
// The script is invoked by a button, named Insert Subform, that is nested
// inside a repeating subform. The new subform is inserted below the current
// subform.
var oSubform = this.resolveNode("Subform2");
var oNewInstance = oSubform.instanceManager.addInstance(1);
var nIndexFrom = oNewInstance.index;
var nIndexTo = this.parent.index + 1;
// Invoke the instanceManager to insert the subform below the current one.
oSubform.instanceManager.moveInstance(nIndexFrom, nIndexTo);
```

Adding and removing a subform

```
// Invoke the instance manager to add and remove the comments subform.
if (fComments.value == "0") {
// In this example, fComments is a document variable used as a flag.
// The fComments variable equals 1 when the comments subform is displayed.
_comments.setInstance(1);
// Add the comments subform. Change the button's caption.
this.resolveNode("caption.value.#text").value = "Clear Comments";
// Set the flag value.
fComments.value = "1";
}
else {
// Remove the comments subform.
_comments.setInstance(0);
// Change the button's caption.
this.resolveNode("caption.value.#text").value = "Add Comments";
// Reset the flag value.
fComments.value = "0";
}
```

Getting or setting object values

These examples illustrate several ways to get or set a value for an object.

Uses

Properties
formattedValue rawValue value

Scripts

Using rawValue

```
// Use the rawValue property to set and get a field's raw value.
TextField1.rawValue = "K1V1W3"; // Set the field's raw value.
TextField2.rawValue = TextField1.rawValue // Get the field's raw value.
```

Using value

```
// Use the value property to set and get the field's raw value.
TextField1.rawValue = "k1V1W3";
TextField2.rawValue = TextField1.value.oneOfChild.value
```

Using formattedValue

```
// Use the formattedValue property to set and get the field's formatted value.
// Use the value property to set and get an object's value (picture).
TextField1.rawValue = "K1V1W3"; // Set the field's raw value.
TextField1.format.picture.value = "A9A 9A9"; // Set the field's display
picture format.
TextField2.rawValue = TextField1.formattedValue; // Get the field's formatted
value.
```

Setting a data object's value

```
// Use the value property to set and get a data object's value.
// In this script, groupNode is a data group and addressL1 is a data value in
// the data file.
TextField1.rawValue = xfa.record.groupNode.address.line1.value;
```

Setting the document variable's value

```
// Use the value property to set and get the document variable's value.
TextField1.rawValue = docVar.value;
```

Working with page numbers and page counts

These examples illustrate several ways to use the host and layout models to work with page numbers and page counts.

The host and layout models have several different properties and methods for working with page numbers and page counts. The properties and methods that you should use depend on what the script does and when it executes.

Many of the host properties and methods are unavailable on the server. Use the host properties and methods to set or get page numbers at run time.

None of the layout methods set the page number. Use the layout methods to get the current page at `layout:ready` or to display the page numbers at the bottom of the page and see the page number when you open a form on a client.

Uses

Properties		Methods	
currentPage layout numPages	rawValue this	absPage absPageCount page	pageCount pageDown pageUp

Scripts

Getting the page number

```
// Use the page layout methods to get the current page number.
TextField1.rawValue = xfa.layout.page(this); // 1-based.
TextField2.rawValue = xfa.layout.absPage(this); // 0-based.
```

Getting the page count using the pageCount method

```
// Use the layout pageCount methods to get the number of pages in a document.
TextField1.rawValue = xfa.layout.pageCount(); // Get the logical number of
pages.
TextField2.rawValue = xfa.layout.absPageCount(); // Get the physical number of
pages.
```

Formatting the pagination

```
// Use the layout page and pageCount methods to format the pagination.
TextField1.rawValue = "Page " + xfa.layout.page(this) + " of " +
xfa.layout.pageCount();
```

Getting and setting the current page number

```
// Use the host currentPage property to get and set the current page number at
// run time.
// This script cannot be used during a layout:ready, form:ready, or initialize
// event. However, it will work if the script is on a button.
xfa.host.currentPage = 1; // Go to page 2 (0-based).
```

Getting the page count using the numPages property

```
// Use the host numPages property to get the number of pages in a document.
TextField1.rawValue = xfa.host.numPages; // Get the number of pages.
```

Navigating down a document

```
// Use the host pageDown() method to navigate through a document.
xfa.host.pageDown(); // Go to the next page.
```

Navigating up a document

```
// Use the host pageUp() method to navigate through a document.
xfa.host.pageUp(); // Go to the previous page.
```

Concatenating data values

This example illustrates how to concatenate data values into an address block and ensure that there are no blank lines.

Uses

Properties		Methods
multiLine oneOfChild	rawValue value	record

Script

Concatenating data values

```
// Get the values from the data model.
var sName = xfa.record.groupNode.address.line1.value;
var sPostbox = xfa.record.groupNode.address.line2.value;
var sStreet = xfa.record.groupNode.address.line3.value;
var sCity = xfa.record.groupNode.address.line4.value;
var sRegion = xfa.record.groupNode.address.line5.value;
var sCountry = xfa.record.groupNode.address.line6.value;
var sPostcode = xfa.record.groupNode.address.line7.value;
```

```
var addressArray = new
Array(sName,sPostbox,sStreet,sCity,sRegion,sCountry,sPostcode);

var sAddressBlock = "";

// Don't display the postbox if the value is not provided.
if (addressArray[1] == null) {
     sAddressBlock = addressArray[0] + "\n" + addressArray[2] + "\n" +
addressArray[3] + "\n";
} else {
     sAddressBlock = addressArray[0] + "\n" + addressArray[1] + "\n" +
addressArray[3] + "\n";
}

// Do not display the region if the value is not provided.
if (addressArray[4] == null) {
sAddressBlock = sAddressBlock + addressArray[5] + " " + addressArray[6];
} else {
sAddressBlock = sAddressBlock + addressArray[4] + ", " + addressArray[5] + " "
+ addressArray[6];
}
TextField2.rawValue = sAddressBlock;
// Make sure the field is set to display a multiple line value. To set the
// multiLine property programmatically, add the following line:
     TextField2.ui.oneOfChild.multiLine = "1";
```

Calculating totals

This example illustrates how to calculate totals.

Uses

Properties	Methods
length rawValue	resolveNodes

Script

Calculating totals

```
// Access a field in a repeating subform by looping through the node list.
var oFields = xfa.resolveNodes("Subform2[*].NumericField4");
var nNodesLength = oFields.length;
var nSum = 0;
for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {
   nSum += oFields.item(nNodeCount).rawValue;
}
TextField1.rawValue = nSum;
```

Changing the background color

These examples illustrate how to change the background color of a subform or fields.

In a form that has a flowable layout, you can change the background color of the entire field, including the caption and the field area, at run time. However, in a form that has a fixed layout, you can only change the background color of the field area at run time.

Uses

Properties		Methods
fillColor	nodes	item
index	parent	resetData
length	value	resolveNode
name	this	resolveNodes

Scripts

Changing the background color of a subform

```
// Alternate the background color of a repeating subform.
var oNodes = xfa.resolveNodes("Subform2[*]");
var nNodesLength = oNodes.length;

for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {
   if (oNodes.item(nNodeCount).index%2 != 0) {
      oNodes.item(nNodeCount).border.fill.color.value = "200,200,250";
   } else {
      oNodes.item(nNodeCount).border.fill.color.value = "200,150,250";
   }
}
```

Changing the background color of a field

```
// Alternate the background color of the NumericField4 field.
// Before running this script, set a background color or set the
// border.fill.presence property to visible.
var oNodes = xfa.resolveNodes("Subform2[*]");
var nNodesLength = oNodes.length;
var sFillColor;

for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {

   if (oNodes.item(nNodeCount).index%2 != 0) {
      sFillColor = "200,200,250";

   } else {
      sFillColor = "200,150,250";

   }
   oNodes.item(nNodeCount).NumericField4.fillColor = sFillColor;
}
```

Changing the background color of rows in a subform

```
// Reset the fields of the current subform.
var dString = "xfa.form.form1.dtls[" + this.parent.index + "]";
var oDetails = xfa.form.resolveNode(dString);
var sDtlFields;

// Build the string of field names to reset.
for (var i = 0; i < oDetails.nodes.length; i++) {
```

```
   sDtlFields = sDtlFields + "," + dString + "." + oDetails.nodes.item(i).name;
}
// Pass the string variable as a parameter.
xfa.host.resetData(sDtlFields); OR
// Alternate the background color of the repeating rows.
if (this.index%2 != 0)  this.border.fill.color.value = "255,255,255"; else
this.border.fill.color.value = "201,201,146";
```

Populating a drop-down list

These examples illustrate several ways to add or remove list items in a drop-down list.

Save the item list before you populate a drop-down list at run time; otherwise, the items will be lost. Only the value is saved in the data.

Uses

Properties		Methods	
length	prevText	addItem	messageBox
newText	rawValue	clearItems	record
nodes	value	item	resolveNode

Scripts

Populating a drop-down list from a web service

```
// Populate the drop-down list with values from a web service.
// The web service used in this example is fictional.
SOAP.wireDump = false;
var oListURL = "http://www.webservice.net/wsdl/query.wsdl";
var e;
try
{
   xfa.host.messageBox("Starting list retrieval.");

   var service = SOAP.connect(oListURL);

   if(typeof service != "object") {
      xfa.host.messageBox("Couldn't get List object.");
   }
   if(service.getAllServiceNames == "undefined") {
      xfa.host.messageBox("Couldn't get getAllServiceNames Call.");
      }

   // Start the query
   var oItems = service.getAllServiceNames();
   if(oItems == null) {
      xfa.host.messageBox("List empty.");
      }
var nCount = 0;
   var nLimit = 10;

   for(var nItemCount in oItems)
   {
```

```
      for(var nItemNode in oItems[nItemCount])
      {
        if (nItemNode == "name")
          DropDownList1.addItem(oItems[nItemCount][nItemNode]);
      }
      if (++nCount >= nLimit)
        break;
    }
}
catch(e)
{
   xfa.host.messageBox("Problem with list Call: " + e);
}
```

Clearing a drop-down list

```
// Clear the items in a drop-down list.
DropDownList1.clearItems();
```

Populating a drop-down list from a data file

```
// Populate the drop-down list with values from a data file.
var oItems = xfa.resolveNode("xfa.record.groupNode.list");
var nItemsLength = oItems.nodes.length;

for (var nItemCount = 0; nItemCount < nItemsLength; nItemCount++) {
   DropDownList1.addItem(oItems.nodes.item(nItemCount).value);
}
DropDownList1.rawValue = "Second item in list";
```

Saving the values from a drop-down list in another field

```
// Access the items in a drop-down list box and save their values in a separate
// field.
var oItems = xfa.resolveNode("DropDownList1.#items");
var nItemsLength = oItems.nodes.length;

for (nItemCount = 0; nItemCount < nItemsLength; nItemCount++){

   if (TextField2.rawValue == null) {
     TextField2.rawValue = oItems.nodes.item(nItemCount).value;
   } else {
     TextField2.rawValue = TextField2.rawValue + "\n" +
oItems.nodes.item(nItemCount).value;
   }
}
```

Accessing a drop-down list value using newText or prevText properties

```
// Use the newText or prevText properties to access a drop-down list value
// before or after the value changes.
// Execute the script on a change event.
TextField1.rawValue = xfa.event.prevText;
TextField2.rawValue = xfa.event.newText;
```

Saving a form

These examples illustrate how to export data from a form and save a form.

Uses

Properties	Methods
target	exportData

Scripts

Exporting form data without specifying a file name

```
// Export a form's data without specifying a file name. The end user is
// prompted to provide the file name.
xfa.host.exportData();      // Will generate data in XDP format.
xfa.host.exportData("", 0); // Will generate data in XML format.
```

Exporting form data using a filename

```
// If you specify a file name, the script must run on a certified form.
xfa.host.exportData("filename.xdp");    // Will generate data in XDP format.
xfa.host.exportData("filename.xml", 0); // Will generate data in XML format.
```

Saving a form

```
// Saving the form is done at the application level, so you need to invoke the
// Acrobat app model.
App.executeMenuItem("SaveAs"); // The end user will be prompted to specify a
// file name.
// However, you must save the form silently if the form needs to be certified
// and the certificate must be trusted for privileged JavaScript.
var mydoc = event.target;
mydoc.saveAs();
```

Making an object visible or invisible

This example illustrates how to make an object visible or invisible. If a print button is invisible, it will prevent the user from printing a form.

The prePrint event triggers immediately before the form is rendered for printing. Similarly, the postPrint event triggers immediately after the form has been printed.

Uses

Properties
presence relevant

Scripts

Setting a field to be visible or invisible

```
// If a field is visible, make it invisible and vice versa.
if(Field1.presence == "visible")
{
   Field1.presence = "invisible";
}
else
{
   Field1.presence = "visible";
}
```

Setting a button to be visible but non-printing

```
// Set a button to be visible but non-printing at design time.
Button1.relevant="-print"
```

Using radio buttons and check boxes

These examples illustrate how to select and clear radio buttons and check boxes.

Uses

Properties	Methods
rawValue	messageBox resolveNodes

Scripts

Selecting a radio button

```
// Select the first radio button.
RadioButtonList.rawValue = '1';
xfa.host.messageBox('Value of RadioButtonList: ' + RadioButtonList.rawValue);

// Select the second radio button.
RadioButtonList.rawValue = '2';
xfa.host.messageBox('Value of RadioButtonList: ' + RadioButtonList.rawValue);
```

Accessing radio buttons

```
// Access the radio buttons.
RadioButtonList.resolveNodes("#field[*]")
```

Clearing a radio button

```
// Clear a RadioButtonList value. Any invalid value will clear the list.
RadioButtonList.rawValue = '3';
xfa.host.messageBox('Value of RadioButtonList: ' + RadioButtonList.rawValue);
```

Selecting a check box

```
// Select a check box.
CheckBox1.rawValue = 1;
xfa.host.messageBox('Value of checkbox: ' + CheckBox1.rawValue);
```

Deselecting a check box

```
// Deselect a check box.
CheckBox1.rawValue = 0;
xfa.host.messageBox('Value of checkbox: ' + CheckBox1.rawValue);
```

Determining that a form has changed

This example illustrates how to determine that a form has changed.

Uses

Properties	Methods
rawValue	messageBox saveXML

Script

Determining that a form has changed

```
// Save a copy of the original XML file.
var sOriginalXML = xfa.data.saveXML();

// Change the form data.
TextField1.rawValue = "changed";

// Determine whether the form data has changed.
if(sOriginalXML == xfa.data.saveXML())
{
   xfa.host.messageBox("Form has not changed.");
}
else
{
     xfa.host.messageBox("Form has changed.");
}
```

Disabling all form fields

This example illustrates how to disable all the fields on a form.

Uses

Properties	Methods
access layout length numPages	item pageContent pageCount

Script

Disabling all form fields

```
// Get the field containers from each page.
for (var nPageCount = 0; nPageCount < xfa.host.numPages; nPageCount++) {
var oFields = xfa.layout.pageContent(nPageCount, "field");
var nNodesLength = oFields.length;

// Set the field property.
for (var nNodeCount = 0; nNodeCount < nNodesLength; nNodeCount++) {
oFields.item(nNodeCount).access = "readOnly";
}
}
```

Index

#

#text scripting property 138

^

{default} scripting property 138

A

absPage scripting method 321
absPageCount scripting method 321
absPageCountInBatch scripting method 322
absPageInBatch scripting method 322
absPageSpan scripting method 323
access scripting property 139
accessKey scripting property 140
action scripting property 141
activity scripting property 142
addInstance scripting method 323
addItem scripting method 324
addNew scripting method 325
addRevocationInfo scripting property 144
Adobe LiveCycle Designer 15
after scripting property 145
afterTarget scripting property 147
aliasNode scripting property 147
all scripting property 148
allowMacro scripting property 148
allowNeutral scripting property 149
allowRichText scripting property 149
anchorType scripting property 150
append scripting method 325
applyXSL scripting method 326
appType scripting property 151
archive scripting property 152
aspect scripting property 153
assignNode scripting method 326

B

baselineShift scripting property 153
beep scripting method 327
before scripting property 154
beforeTarget scripting property 156
bind scripting property 156
binding scripting property 156
blank scripting property 157
bofAction scripting property 159
bookendLeader scripting property 159
bookendTrailer scripting property 160
borderColor scripting property 160
borderWidth scripting property 161
bottomInset scripting property 161
boundItem scripting method 328
break scripting property 162

C

calculationsEnabled scripting property 162
cancel scripting method 328
cancelBatch scripting method 329
cap scripting property 163
change scripting property 164
charEncoding scripting property 164
checksum scripting property 166
circular scripting property 167
classAll scripting property 167
classes
    about 18
    container 22
    content 23
    list 18
    model 23
    node 20
    object 18
    textNode 24
    tree 19
    treeList 19
classId scripting property 168
classIndex scripting property 168
className scripting property 169
clear scripting method 329
clearErrorList scripting method 330
clearItems scripting method 330
clone scripting method 331
close scripting method 331
codeBase scripting property 169
codeType scripting property 170
colSpan scripting property 170
columnWidths scripting property 171
commandType scripting property 171
commitKey scripting property 172
commitOn scripting property 173
connection scripting property 174
ConnectionSet model 385
container class 22
containers 17
contains scripting property 175
content class 23
content scripting property 175
contentType scripting property 176
context scripting property 177
createNode scripting method 332
credentialServerPolicy scripting property 178
crlSign scripting property 178
cSpace scripting property 179
currentPage scripting property 179
currentRecordNumber scripting property 180
currentValue scripting property 181
cursorLocation scripting property 181

cursorType scripting property 181

D

Data model 386
data scripting property 182
dataColumnCount scripting property 183
dataDescription scripting property 184
dataEncipherment scripting property 184
dataLength scripting property 185
dataPrep scripting property 185
dataRowCount scripting property 186
db scripting property 187
decipherOnly scripting property 187
delayedOpen scripting property 188
delete scripting method 333
deleteItem scripting method 333
delimiter scripting property 188
digitalSignature scripting property 189
disable scripting property 189
Document Object Model (DOM) 15
    about 17
    form processing 384
documentCountInBatch scripting method 334
documentInBatch scripting method 334
draws 17

E

editValue scripting property 190
embedPDF scripting property 190
emit scripting method 334
encipherOnly scripting property 191
endChar scripting property 191
enumerate scripting method 335
eofAction scripting property 192
errorCorrectionLevel scripting property 192
evaluate scripting method 335
Event model 386
examples
    calculating totals 398
    changing the background color 398
    concatenating data values 397
    creating a node 392
    determining that a form has changed 403
    disabling all form fields 404
    getting or setting object values 395
    making an object visible or invisible 402
    manipulating subform instances 394
    populating a drop-down list 400
    referencing objects 390
    saving a form 401
    using radio buttons and check boxes 403
    working with page numbers and page counts 396
execCalculate scripting method 336
execEvent scripting method 336
execInitialize scripting method 337
execute scripting method 337
executeType scripting property 193
execValidate scripting method 338
exportData scripting method 338

F

fields 17
fillable regions 17
fillColor scripting property 193
first scripting method 339
fontColor scripting property 194
form designs
    subforms and containers 17
Form model 387
Form Object Model. *See* XML Form Object Model
form processing 384
format scripting property 195
formatMessage scripting property 195
formattedValue scripting property 196
formatTest scripting property 196
formNodes scripting method 340
fracDigits scripting property 197
from scripting property 198
fullText scripting property 198

G

getAttribute scripting method 340
getDelta scripting method 341
getDeltas scripting method 341
getDisplayItem scripting method 342
getElement scripting method 342
getItemState scripting method 343
getSaveItem scripting method 343
gotoRecord scripting method 343
gotoURL scripting method 344

H

h scripting method 345
h scripting property 199
hAlign scripting property 200
hand scripting property 200
hasDataChanged scripting method 345
Host model 387
href scripting property 202

I

id scripting property 203
imagingBBox scripting property 204
importData scripting method 346
index scripting property 204
initial scripting property 205
initialNumber scripting property 205
input scripting property 206
insert scripting method 346
insertInstance scripting method 347
instanceIndex scripting property 206
intact scripting property 207
inverted scripting property 208
isBOF scripting method 348
isCompatibleNS scripting method 348
isContainer scripting property 208
isDefined scripting property 209
isEOF scripting method 349

isNull scripting property 209
isPropertySpecified scripting method 349
isRecordGroup scripting method 350
item scripting method 350

J

JavaScript
    See also examples
join scripting property 210

K

keyAgreement scripting property 211
keyCertSign scripting property 211
keyDown scripting property 212
keyEncipherment scripting property 212

L

labelRef scripting property 213
language scripting property 213
last scripting method 351
Layout model 388
layout scripting property 214
leadDigits scripting property 215
leader scripting property 215
leftInset scripting property 216
length scripting property 217
lineHeight scripting property 217
lineThrough scripting property 218
lineThroughPeriod scripting property 219
list class 18
loadXML scripting method 351
locale scripting property 219
lockType scripting property 220
long scripting property 221

M

mandatory scripting property 221
mandatoryMessage scripting property 222
marginLeft scripting property 222
marginRight scripting property 223
match scripting property 224
max scripting property 225
maxChars scripting property 226
maxH scripting property 226
maxLength scripting property 227
maxW scripting property 228
messageBox scripting method 352
min scripting property 228
minH scripting property 229
minW scripting property 230
model
    about DOMs 384
    connectionSet 385
    Data 386
    Event 386
    Form 387
    Host 387
    Layout 388
    Signature 388
    sourceSet 388
    XFA 389
model class 23
model scripting property 230
modifier scripting property 231
moduleHeight scripting property 231
moduleWidth scripting property 232
moveCurrentRecord scripting method 353
moveInstance scripting method 354
multiLine scripting property 233

N

name scripting property 234
namedItem scripting method 355
newContentType scripting property 234
newText scripting property 235
next scripting method 355
next scripting property 235
node class 20
nodes scripting property 236
nonRepudiation scripting property 237
ns scripting property 237
nullTest scripting property 238
numbered scripting property 239
numPages scripting property 240

O

object class 18
oneOfChild scripting property 241
open scripting method 356
open scripting property 242
openList scripting method 356
operation scripting property 243
orientation scripting property 245
output scripting property 245
overflowLeader scripting property 246
overflowTarget scripting property 246
overflowTrailer scripting property 247
overline scripting property 247
overlinePeriod scripting property 248
override scripting property 249

P

page scripting method 356
pageContent scripting method 357
pageCount scripting method 359
pageDown scripting method 359
pageSpan scripting method 360
pageUp scripting method 360
parent scripting property 251
parentSubform scripting property 251
passwordChar scripting property 252
permissions scripting property 252
placement scripting property 253
platform scripting property 254
posture scripting property 254
presence scripting property 255

preserve scripting property 256
prevContentType scripting property 256
previous scripting method 361
previous scripting property 257
prevText scripting property 258
print scripting method 362
printCheckDigit scripting property 258
priority scripting property 259

R

radius scripting property 260
radixOffset scripting property 260
rate scripting property 261
rawValue scripting property 262
ready scripting property 263
recalculate scripting method 363
record scripting method 364
recordsAfter scripting property 263
recordsBefore scripting property 264
reenter scripting property 264
ref scripting property 265
relation scripting property 266
relayout scripting method 365
relayoutPageArea scripting method 365
relevant scripting property 266
remerge scripting method 366
remove scripting method 366
removeAttribute scripting method 367
requery scripting method 368
reserve scripting property 267
reset scripting method 368
resetData scripting method 369
resolveNode scripting method 369
resolveNodes scripting method 370
response scripting method 371
restore scripting method 372
resync scripting method 372
rightInset scripting property 269
role scripting property 270
rotate scripting property 270
rowColumnRatio scripting property 271
runAt scripting property 272

S

save scripting property 272
savedValue scripting property 273
saveFilteredXML scripting method 372
saveXML scripting method 373
scope scripting property 273
scripting examples. *See* examples
scripting methods
    absPage 321
    absPageCount 321
    absPageCountInBatch 322
    absPageInBatch 322
    absPageSpan 323
    addInstance 323
    addItem 324
    addNew 325
    append 325
    applyXSL 326
    assignNode 326
    beep 327
    boundItem 328
    cancel 328
    cancelBatch 329
    clear 329
    clearErrorList 330
    clearItems 330
    clone 331
    close 331
    createNode 332
    delete 333
    deleteItem 333
    documentCountInBatch 334
    documentInBatch 334
    emit 334
    enumerate 335
    evaluate 335
    execCalculate 336
    execEvent 336
    execInitialize 337
    execute 337
    execValidate 338
    exportData 338
    first 339
    formNodes 340
    getAttribute 340
    getDelta 341
    getDeltas 341
    getDisplayItem 342
    getElement 342
    getItemState 343
    getSaveItem 343
    gotoRecord 343
    gotoURL 344
    h 345
    hasDataChanged 345
    importData 346
    insert 346
    insertInstance 347
    isBOF 348
    isCompatibleNS 348
    isEOF 349
    isPropertySpecified 349
    isRecordGroup 350
    item 350
    last 351
    loadXML 351
    messageBox 352
    moveCurrentRecord 353
    moveInstance 354
    namedItem 355
    next 355
    open 356
    openList 356
    page 356
    pageContent 357
    pageCount 359

pageDown 359
pageSpan 360
pageUp 360
previous 361
print 362
recalculate 363
record 364
relayout 365
relayoutPageArea 365
remerge 366
remove 366
removeAttribute 367
removeInstance 367
requery 368
reset 368
resetData 369
resolveNode 369
resolveNodes 370
response 371
restore 372
resync 372
saveFilteredXML 372
saveXML 373
selectedMember 373
setAttribute 374
setElement 374
setFocus 375
setInstance 375
setItemState 376
sheet 377
sheetCount 377
sheetCountInBatch 377
sheetInBatch 378
sign 378
update 379
updateBatch 380
verify 380
w 381
x 382
y 382
scripting properties
#text 138
{default} 138
access 139
accessKey 140
action 141
activity 142
addRevocationInfo 144
after 145
afterTarget 147
aliasNode 147
all 148
allowMacro 148
allowNeutral 149
allowRichText 149
anchorType 150
appType 151
archive 152
aspect 153
baselineShift 153
before 154
beforeTarget 156
bend 156
binding 156
blank 157
bofAction 159
bookendLeader 159
bookendTrailer 160
borderColor 160
borderWidth 161
bottomInset 161
break 162
calculationsEnabled 162
cap 163
change 164
charEncoding 164
checksum 166
circular 167
classAll 167
classId 168
classIndex 168
className 169
codeBase 169
codeType 170
colSpan 170
columnWidths 171
commandType 171
commitKey 172
commitOn 173
connection 174
contains 175
content 175
contentType 176
context 177
credentialServerPolicy 178
crlSign 178
cSpace 179
currentPage 179
currentRecordNumber 180
currentValue 181
cursorLocation 181
cursorType 181
data 182
dataColumnCount 183
dataDescription 184
dataEncipherment 184
dataLength 185
dataPrep 185
dataRowCount 186
db 187
decipherOnly 187
delayedOpen 188
delimiter 188
digitalSignature 189
disable 189
editValue 190
embedPDF 190
encipherOnly 191
endChar 191
eofAction 192

errorCorrectionLevel 192
executeType 193
fillColor 193
fontColor 194
format 195
formatMessage 195
formattedValue 196
formatTest 196
fracDigits 197
from 198
fullText 198
h 199
hAlign 200
hand 200
href 202
id 203
imagingBBox 204
index 204
initial 205
initialNumber 205
input 206
instanceIndex 206
intact 207
inverted 208
isContainer 208
isDefined 209
isNull 209
join 210
keyAgreerment 211
keyCertSign 211
keyDown 212
keyEncipherment 212
labelRef 213
language 213
layout 214
leadDigits 215
leader 215
leftInset 216
length 217
lineHeight 217
lineThrough 218
lineThroughPeriod 219
locale 219
lockType 220
long 221
mandatory 221
mandatoryMessage 222
marginLeft 222
marginRight 223
match 224
max 225
maxChars 226
maxH 226
maxLength 227
maxW 228
min 228
minH 229
minW 230
model 230
modifier 231
moduleHeight 231
moduleWidth 232
multiLine 233
name 234
newContentType 234
newText 235
next 235
nodes 236
nonRepudiation 237
ns 237
nullTest 238
numbered 239
numPages 240
oneOfChild 241
open 242
operation 243
orientation 245
output 245
overflowLeader 246
overflowTarget 246
overflowTrailer 247
overline 247
overlinePeriod 248
override 249
parent 251
parentSubform 251
passwordChar 252
permissions 252
placement 253
platform 254
posture 254
presence 255
preserve 256
prevContentType 256
previous 257
prevText 258
printCheckDigit 258
priority 259
radius 260
radixOffset 260
rate 261
rawValue 262
ready 263
recordsAfter 263
recordsBefore 264
reenter 264
ref 265
relation 266
relevant 266
reserve 267
rightInset 269
role 270
rotate 270
rowColumnRatio 271
runAt 272
save 272
savedValue 273
scope 273
scriptTest 274
selectedIndex 275

selEnd 276
selStart 276
server 277
shape 277
shift 278
short 279
signatureType 279
size 280
slope 280
soapFaultCode 281
soapFaultString 281
somExpression 282
spaceAbove 282
spaceBelow 283
startAngle 283
startChar 284
startNew 284
stateless 285
stock 285
stroke 286
sweepAngle 287
tabDefault 288
tabStops 289
target 289
targetType 290
textEncoding 291
textEntry 293
textIndent 293
textLocation 294
thickness 295
this 295
timeout 296
timeStamp 296
title 297
topInset 297
trailer 298
transferEncoding 298
transient 299
truncate 299
type 300
typeface 304
underline 305
underlinePeriod 306
url 306
urlPolicy 306
usage 307
use 308
uuid 310
validationMessage 311
validationsEnabled 311
vAlign 312
value 312
valueRef 314
variation 314
version 315
w 316
weight 317
wideNarrowRatio 317
x 318
xdpContent 319
y 320
scriptTest scripting property 274
selectedIndex scripting property 275
selectedMember scripting method 373
selEnd scripting property 276
selStart scripting property 276
server scripting property 277
setAttribute scripting method 374
setElement scripting method 374
setFocus scripting method 375
setInstance scripting method 375
setItemState scripting method 376
shape scripting property 277
sheet scripting method 377
sheetCount scripting method 377
sheetCountInBatch scripting method 377
sheetInBatch scripting method 378
shift scripting property 278
short scripting property 279
sign scripting method 378
Signature model 388
signatureType scripting property 279
size scripting property 280
slope scripting property 280
soapFaultCode scripting property 281
soapFaultString scripting property 281
somExpression scripting property 282
sourceSet model 388
spaceAbove scripting property 282
spaceBelow scripting property 283
startAngle scripting property 283
startChar scripting property 284
startNew scripting property 284
stateless scripting property 285
stock scripting property 285
stroke scripting property 286
subforms 17
sweepAngle scripting property 287

T

tabDefault scripting property 288
tabStops scripting property 289
target scripting property 289
targetType scripting property 290
textEncoding scripting property 291
textEntry scripting property 293
textIndent scripting property 293
textLocation scripting property 294
textNode class 24
thickness scripting property 295
this scripting property 295
timeout scripting property 296
timeStamp scripting property 296
title scripting property 297
topInset scripting property 297
trailer scripting property 298
transferEncoding scripting property 298
transient scripting property 299
tree class 19

treeList class 19
truncate scripting property 299
type scripting property 300
typeface scripting property 304

U

underline scripting property 305
underlinePeriod scripting property 306
update scripting method 379
updateBatch scripting method 380
url scripting property 306
urlPolicy scripting property 306
usage scripting property 307
use scripting property 308
uuid scripting property 310

V

validationMessage scripting property 311
validationsEnabled scripting property 311
vAlign scripting property 312
value scripting property 312
valueRef scripting property 314
variation scripting property 314
verify scripting method 380
version scripting property 315

W

w scripting method 381
w scripting property 316
weight scripting property 317
wideNarrowRatio scripting property 317

X

x scripting method 382
x scripting property 318
xdpContent scripting property 319
XFA model 389
XML Form Object Model
    about 15, 17
    class hierarchy 18
    DOM form processing 384

Y

y scripting method 382
y scripting property 320