



JavaScript™ for Acrobat® API Reference

Adobe® Acrobat® SDK
バージョン 8.0

© 2006 Adobe Systems Incorporated. All rights reserved.

Adobe® Acrobat® SDK 8.0 JavaScript for Acrobat API Reference (Microsoft® Windows® / Macintosh® 版)

Edition 1.0、2006 年 10 月

本マニュアルがエンドユーザ使用許諾契約を含むソフトウェアと共に提供される場合、本マニュアルおよびその中に記載されているソフトウェアは、エンドユーザ使用許諾契約にもとづいて提供されるものであり、当該エンドユーザ使用許諾契約の契約条件に従つてのみ使用または複製することが可能となるものです。当該エンドユーザ使用許諾契約により許可されている場合を除き、本マニュアルのいかなる部分といえども、Adobe Systems Incorporated (アドビシステムズ社) の書面による事前の許可なしに、電子的、機械的、録音、その他いかなる形式・手段であれ、複製、検索システムへの保存、または伝送を行うことはできません。本マニュアルの内容は、エンドユーザ使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

本マニュアルに記載される内容は、あくまでも参考用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従つて、当該情報が、アドビシステムズ社による確約として解釈されではありません。アドビシステムズ社は、本マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。

新しいアートワークを創作するためにテンプレートとして取り込もうとする既存のアートワークまたは画像は、著作権法により保護されている可能性のあるものであることをご留意ください。保護されているアートワークまたは画像を新しいアートワークに許可なく取り込んだ場合、著作権者の権利を侵害することがあります。従つて、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名および会社ロゴは、実在の会社・組織を示すものではありません。

Adobe、Adobe ロゴ、Acrobat、Distiller、FrameMaker、LiveCycle、PostScript および Reader は、アドビシステムズ社の米国ならびに他の国における商標または登録商標です。

Apple および Mac OS は、米国およびその他の国で登録された Apple Inc. の商標です。

JavaScript は、米国およびその他の国における Sun Microsystems, Inc. の商標または登録商標です。

Microsoft および Windows は、米国およびその他の国における Microsoft Corporation の登録商標または商標です。

その他すべての商標は、それぞれの権利帰属者の所有物です。

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

Notice to U.S. Government End Users.The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. § 2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. § 12.212 or 48 C.F.R. § 227.7202, as applicable.Consistent with 48 C.F.R. § 12.212 or 48 C.F.R. §§ 227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein.Unpublished-rights reserved under the copyright laws of the United States.Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA.For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

はじめに	28
このマニュアルの内容	28
対象読者	28
関連ドキュメント	28
1 概要	30
構文	30
パス	31
セーフパス	31
セキュリティによる制限がないコンテキスト	31
セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト	32
ユーザ環境設定	32
クイックバー	32
コードサンプルで使用しているドメイン名	34
2 JavaScript API	35
ADBC	36
ADBC のプロパティ	36
SQL 型	36
JavaScript 型	37
ADBC のメソッド	38
getDataSourceList	38
newConnection	38
Alerter	40
Alerter のメソッド	40
dispatch	40
AlternatePresentation	43
AlternatePresentation のプロパティ	43
active	43
type	43
AlternatePresentation のメソッド	44
start	44
stop	44
Annotation	45
注釈タイプ	45
Annotation のプロパティ	47
alignment	47
AP	48
arrowBegin	49
arrowEnd	50
attachIcon	50
author	51
borderEffectIntensity	51
borderEffectStyle	52
callout	52
caretSymbol	52

2 JavaScript API (続き)

Annotation (続き)

Annotation のプロパティ (続き)	
contents	53
creationDate	53
dash	54
delay	54
doc	55
doCaption	56
fillColor	56
gestures	57
hidden	57
inReplyTo	58
intent	58
leaderExtend	59
leaderLength	59
lineEnding	60
lock	60
modDate	61
name	61
notelcon	62
noView	63
opacity	63
page	64
point	64
points	65
popupOpen	66
popupRect	66
print	67
quads	67
rect	68
readOnly	68
refType	68
richContents	69
richDefaults	70
rotate	70
seqNum	71
soundIcon	71
state	72
stateModel	72
strokeColor	72
style	73
subject	73
textFont	74
textSize	75
toggleNoView	75
type	76
vertices	76
width	77

2 JavaScript API (続き)

Annotation (続き)

Annotation のメソッド	77
destroy	77
getProps	78
getStateInModel	79
setProps	79
transitionToState	80
Annot3D	82
Annot3D のプロパティ	82
activated	82
context3D	82
innerRect	83
name	83
page	83
rect	84
app	85
app のプロパティ	85
activeDocs	85
calculate	86
constants	86
focusRect	87
formsVersion	87
fromPDFConverters	88
fs	88
fullscreen	89
language	89
media	90
monitors	90
numPlugIns	91
openInPlace	91
platform	92
plugins	92
printColorProfiles	93
printerNames	93
runtimeHighlight	94
runtimeHighlightColor	94
thermometer	94
toolbar	95
toolbarHorizontal	95
toolbarVertical	96
viewerType	96
viewerVariation	97
viewerVersion	97
app のメソッド	97
addMenuItem	97
addSubMenu	99
addToolBar	100
alert	101
beep	103
beginPriv	104

2 JavaScript API (続き)

app (続き)	
app のメソッド (続き)	
browseForDoc	104
clearInterval	106
clearTimeOut	106
endPriv	106
execDialog	107
execMenuItem	120
getNthPlugInName	122
getPath	122
goBack	123
goForward	123
hideMenuItem	124
hideToolbarButton	124
launchURL	124
listMenuItems	125
listToolbarButtons	126
mailGetAddrs	127
mailMsg	128
newDoc	129
newFDF	130
openDoc	131
openFDF	133
popUpMenu	134
popUpMenuEx	135
removeToolBar	136
response	137
setInterval	138
setTimeOut	139
trustedFunction	140
trustPropagatorFunction	143
app.media	148
app.media のプロパティ	148
align	148
canResize	149
closeReason	149
defaultVisible	150
ifOffScreen	150
layout	151
monitorType	151
openCode	152
over	153
pageEventNames	153
raiseCode	154
raiseSystem	154
renditionType	155
status	155
trace	156
version	156
windowType	157

2 JavaScript API (続き)

app.media (続き)	
app.media のメソッド	157
addStockEvents	157
alertFileNotFound	158
alertSelectFailed	158
argsDWIM	159
canPlayOrAlert	159
computeFloatWinRect	160
constrainRectToScreen	161
createPlayer	161
getAltTextData	163
getAltTextSettings	164
getAnnotStockEvents	165
getAnnotTraceEvents	165
getPlayers	165
getPlayerStockEvents	166
getPlayerTraceEvents	167
getRenditionSettings	167
getURLData	167
getURLSettings	168
getWindowBorderSize	170
openPlayer	170
removeStockEvents	172
startPlayer	172
Bookmark	173
Bookmark のプロパティ	173
children	173
color	173
doc	174
name	174
open	175
parent	175
style	175
Bookmark のメソッド	176
createChild	176
execute	176
insertChild	177
remove	178
setAction	178
catalog	179
catalog のプロパティ	179
isIdle	179
jobs	179
catalog のメソッド	180
getIndex	180
remove	180
CatalogJob	181
CatalogJob のプロパティ	181
path	181
type	181
status	181

2 JavaScript API (続き)

Certificate	182
Certificate のプロパティ	182
binary	182
issuerDN	182
keyUsage	183
MD5Hash	183
privateKeyValidityEnd	183
privateKeyValidityStart	184
SHA1Hash	184
serialNumber	184
subjectCN	185
subjectDN	185
ubRights	185
usage	186
validityEnd	188
validityStart	188
Collab	189
Collab のメソッド	189
addStateModel	189
documentToStream	190
removeStateModel	190
color	191
カラー配列	191
color のプロパティ	191
color のメソッド	192
convert	192
equal	193
colorConvertAction	194
colorConvertAction のプロパティ	194
action	194
alias	194
colorantName	195
convertIntent	195
convertProfile	196
embed	196
isProcessColor	196
matchAttributesAll	197
matchAttributesAny	198
matchIntent	198
matchSpaceTypeAll	199
matchSpaceTypeAny	200
preserveBlack	200
useBlackPointCompensation	201
Column	202
Column のプロパティ	202
columnNum	202
name	202
type	202
typeName	203
value	203

2 JavaScript API (続き)

ColumnInfo	204
ColumnInfo のプロパティ	204
name	204
description	204
type	204
typeName	205
Connection	206
Connection のメソッド	206
close	206
getColumnList	206
getTableList	207
newStatement	207
console	208
console のメソッド	208
clear	208
hide	208
println	208
show	209
Data	210
Data のプロパティ	210
creationDate	210
description	210
MIMEType	211
modDate	211
name	211
path	212
size	212
DataSourceInfo	213
DataSourceInfo のプロパティ	213
name	213
description	213
dbg	214
dbg のプロパティ	214
bps	214
dbg のメソッド	215
c	215
cb	215
q	215
sb	216
si	217
sn	217
so	217
sv	217
Dialog	218
Dialog のメソッド	218
enable	218
end	218
load	219
store	219

2 JavaScript API (続き)

DirConnection	220
DirConnection のプロパティ	220
canList	220
canDoCustomSearch	220
canDoCustomUISearch	221
canDoStandardSearch	221
groups	222
name	222
uiName	222
DirConnection のメソッド	223
search	223
setOutputFields	224
Directory	226
Directory のプロパティ	226
info	226
Directory のメソッド	228
connect	228
Doc	230
Doc のプロパティ	231
alternatePresentations	231
author	232
baseURL	232
bookmarkRoot	232
calculate	233
creationDate	233
creator	234
dataObjects	234
delay	235
dirty	235
disclosed	236
docID	237
documentFileName	237
dynamicXFAForm	238
external	238
filesize	239
hidden	239
hostContainer	240
icons	240
info	241
innerAppWindowRect	242
innerDocWindowRect	243
isModal	243
keywords	243
layout	244
media	244
metadata	245
modDate	246
mouseX	247
mouseY	247
noautocomplete	248

2 JavaScript API (続き)

Doc (続き)

Doc のプロパティ (続き)	
nocache	248
numFields	249
numPages	249
numTemplates	250
path	250
outerAppWindowRect	251
outerDocWindowRect	251
pageNum	251
pageWindowRect	252
permStatusReady	252
producer	253
requiresFullSave	253
securityHandler	253
selectedAnnots	254
sounds	254
spellDictionaryOrder	255
spellLanguageOrder	255
subject	256
templates	256
title	257
URL	257
viewState	257
xfa	259
XFAForeground	260
zoom	261
zoomType	261
Doc のメソッド	262
addAnnot	262
addField	264
addIcon	265
addLink	265
addRecipientListCryptFilter	267
addRequirement	268
addScript	269
addThumbnails	270
addWatermarkFromFile	270
addWatermarkFromText	272
addWeblinks	274
bringToFront	274
calculateNow	275
closeDoc	275
colorConvertPage	276
createDataObject	277
createTemplate	278
deletePages	279
deleteSound	279
embedDocAsDataObject	280
embedOutputIntent	280

2 JavaScript API (続き)

Doc (続き)

Doc のメソッド (続き)	
encryptForRecipients	281
encryptUsingPolicy	283
exportAsFDF	285
exportAsFDFStr	286
exportAsText	287
exportAsXFDF	288
exportAsXFDFStr	289
exportDataObject	290
exportXFAData	291
extractPages	293
flattenPages	294
getAnnot	294
getAnnot3D	295
getAnnots	295
getAnnots3D	296
getColorConvertAction	297
getDataObject	297
getDataObjectContents	298
getField	299
getIcon	300
getLegalWarnings	301
getLinks	304
getNthFieldName	305
getNthTemplate	306
getOCGs	306
getOCGOrder	307
getPageBox	307
getPageLabel	308
getPageNthWord	308
getPageNthWordQuads	309
getPageNumWords	309
getPageRotation	310
getPageTransition	310
getPrintParams	311
getSound	311
getTemplate	311
getURL	312
gotoNamedDest	313
importAnFDF	313
importAnXFDF	314
importDataObject	314
importIcon	315
importSound	316
importTextData	317
importXFAData	318
insertPages	318
mailDoc	319
mailForm	320

2 JavaScript API (続き)

Doc (続き)

Doc のメソッド (続き)	
movePage	321
newPage	322
openDataObject	322
print	323
removeDataObject	325
removeField	325
removeIcon	326
removeLinks	326
removeRequirement	327
removeScript	327
removeTemplate	327
removeThumbnails	328
removeWeblinks	328
replacePages	329
resetForm	329
saveAs	330
scroll	332
selectPageNthWord	333
setAction	333
setDataObjectContents	334
setOCGOrder	336
setPageAction	336
setPageBoxes	337
setPageLabels	337
setPageRotations	338
setPageTabOrder	339
setPageTransitions	340
spawnPageFromTemplate	340
submitForm	342
syncAnnotScan	346
Doc.media	348
Doc.media のプロパティ	348
canPlay	348
Doc.media のメソッド	349
deleteRendition	349
getAnnot	349
getAnnots	350
getOpenPlayers	351
getRendition	352
newPlayer	352
Embedded PDF	354
Embedded PDF のプロパティ	354
messageHandler	354
Embedded PDF のメソッド	355
postMessage	355

2 JavaScript API (続き)

Error	356
Error のプロパティ	357
fileName	357
lineNumber	357
extMessage	357
message	358
name	358
Error のメソッド	358
toString	358
event	359
イベントのタイプと名前の組み合わせ	359
文書イベントの処理	368
フォームイベントの処理	369
マルチメディアイベントの処理	369
event のプロパティ	370
change	370
changeEx	370
commitKey	372
fieldFull	372
keyDown	373
modifier	374
name	374
rc	375
richChange	375
richChangeEx	376
richValue	377
selEnd	378
selStart	378
shift	379
source	379
target	380
targetName	380
type	381
value	381
willCommit	382
EventListener	383
EventListener のメソッド	384
afterBlur	384
afterClose	384
afterDestroy	385
afterDone	385
afterError	386
afterEscape	386
afterEveryEvent	386
afterFocus	387
afterPause	387
afterPlay	388
afterReady	388

2 JavaScript API (続き)

EventListener (続き)

EventListener のメソッド (続き)	
afterScript	389
afterSeek	390
afterStatus	391
afterStop	391
onBlur	392
onClose	392
onDestroy	393
onDone	393
onError	393
onEscape	394
onEveryEvent	394
onFocus	395
onGetRect	395
onPause	396
onPlay	396
onReady	396
onScript	397
onSeek	397
onStatus	397
onStop	398
Events	399
Events のメソッド	399
add	399
dispatch	400
remove	401
FDF	402
FDF のプロパティ	402
deleteOption	402
isSigned	402
numEmbeddedFiles	403
FDF のメソッド	403
addContact	403
addEmbeddedFile	404
addRequest	405
close	405
mail	406
save	407
signatureClear	407
signatureSign	408
signatureValidate	409
Field	410
フィールドの属性とウィジェットの属性	411
Field のプロパティ	412
alignment	412
borderStyle	412
buttonAlignX	413
buttonAlignY	414

2 JavaScript API (続き)

Field (続き)

Field のプロパティ (続き)	
buttonFitBounds	415
buttonPosition	415
buttonScaleHow	416
buttonScaleWhen	416
calcOrderIndex	417
charLimit	418
comb	418
commitOnSelChange	419
currentValueIndices	419
defaultStyle	421
defaultValue	422
doNotScroll	422
doNotSpellCheck	423
delay	423
display	424
doc	425
editable	425
exportValues	426
fileSelect	426
fillColor	427
hidden	428
highlight	428
lineWidth	429
multiline	430
multipleSelection	430
name	431
numItems	431
page	432
password	433
print	433
radiosInUnison	433
readonly	434
rect	434
required	435
richText	436
richValue	437
rotation	438
strokeColor	439
style	439
submitName	440
textColor	441
textFont	441
textSize	443
type	443
userName	444
value	445
valueAsString	445

2 JavaScript API (続き)

Field (続き)

Field のメソッド	446
browseForFileToSubmit	446
buttonGetCaption	446
buttonGetIcon	447
buttonImportIcon	448
buttonSetCaption	449
buttonSetIcon	449
checkThisBox	450
clearItems	451
defaultIsChecked	451
deleteItemAt	452
getArray	453
getItemAt	453
getLock	454
insertItemAt	454
isBoxChecked	455
isDefaultChecked	455
setAction	456
setFocus	457
setItems	457
setLock	458
signatureGetModifications	459
signatureGetSeedValue	461
signatureInfo	461
signatureSetSeedValue	462
signatureSign	470
signatureValidate	472
FullScreen	474
FullScreen のプロパティ	474
backgroundColor	474
clickAdvances	474
cursor	475
defaultTransition	475
escapeExits	476
isFullScreen	476
loop	476
timeDelay	477
transitions	477
usePageTiming	478
useTimer	478
global	479
グローバルプロパティの作成	479
グローバルプロパティの削除	480
グローバルオブジェクトセキュリティポリシー	480
global のメソッド	481
setPersistent	481
subscribe	481

2 JavaScript API (続き)

HostContainer	483
HostContainer のプロパティ	483
messageHandler	483
HostContainer のメソッド	485
postMessage	485
Icon	486
Icon Stream	487
identity	488
identity のプロパティ	488
corporation	488
email	488
loginName	488
name	489
Index	490
Index のプロパティ	490
available	490
name	490
path	491
selected	491
Index のメソッド	491
build	491
Link	493
Link のプロパティ	493
borderColor	493
borderWidth	493
highlightMode	493
rect	494
Link のメソッド	494
setAction	494
Marker	495
Marker のプロパティ	495
frame	495
index	495
name	495
time	496
Markers	497
Markers のプロパティ	497
player	497
Markers のメソッド	497
get	497
MediaOffset	499
MediaOffset のプロパティ	499
frame	499
marker	499
time	500
MediaPlayer	501
MediaPlayer のプロパティ	501
annot	501
defaultSize	501
doc	502

2 JavaScript API (続き)

MediaPlayer (続き)	
MediaPlayer のプロパティ (続き)	
events	502
hasFocus	502
id	503
innerRect	503
isOpen	504
isPlaying	504
markers	504
outerRect	505
page	505
settings	506
uiSize	506
visible	507
MediaPlayer のメソッド	508
close	508
open	508
pause	509
play	509
seek	510
setFocus	511
stop	512
triggerGetRect	512
where	513
MediaReject	514
MediaReject のプロパティ	514
rendition	514
MediaSelection	515
MediaSelection のプロパティ	515
selectContext	515
players	516
rejects	516
rendition	517
MediaSettings	518
MediaSettings のプロパティ	518
autoPlay	518
baseURL	518
bgColor	519
bgOpacity	519
data	520
duration	520
endAt	521
floating	522
layout	523
monitor	523
monitorType	524
page	525
palindrome	525
players	526

2 JavaScript API (続き)

MediaSettings (続き)	
MediaSettings のプロパティ (続き)	
rate	527
repeat	527
showUI	528
startAt	528
visible	529
volume	530
windowType	530
Monitor	532
Monitor のプロパティ	532
colorDepth	532
isPrimary	532
rect	533
workRect	533
Monitors	534
Monitors のメソッド	534
bestColor	534
bestFit	535
desktop	535
document	536
filter	536
largest	537
leastOverlap	537
mostOverlap	538
nonDocument	538
primary	539
secondary	539
select	539
tallest	540
widest	540
Net	542
Net のプロパティ	542
SOAP	542
Discovery	543
HTTP	543
Net のメソッド	544
Net.HTTP	545
Net.HTTP のメソッド	545
request	545
OCG	548
OCG のプロパティ	548
constants	548
initState	549
locked	549
name	549
state	550

2 JavaScript API (続き)

OCG (続き)	
OCG のメソッド	551
getIntent	551
setAction	551
setIntent	552
PlayerInfo	553
PlayerInfo のプロパティ	553
id	553
mimeTypes	553
name	554
version	554
PlayerInfo のメソッド	555
canPlay	555
canUseData	555
honors	556
PlayerInfoList	560
PlayerInfoList のメソッド	560
select	560
PlugIn	561
PlugIn のプロパティ	561
certified	561
loaded	561
name	561
path	562
version	562
PrintParams	563
PrintParams のプロパティ	563
binaryOK	563
bitmapDPI	563
booklet	564
colorOverride	566
colorProfile	567
constants	568
downloadFarEastFonts	568
fileName	569
firstPage	570
flags	570
fontPolicy	572
gradientDPI	573
interactive	573
lastPage	574
nUpAutoRotate	574
nUpNumPagesH	575
nUpNumPagesV	575
nUpPageBorder	576
nUpPageOrder	576
pageHandling	577
pageSubset	578
printAsImage	579
printContent	579

2 JavaScript API (続き)

PrintParams (続き)

PrintParams のプロパティ (続き)	
printerName	580
psLevel	581
rasterFlags	581
reversePages	583
tileLabel	583
tileMark	583
tileOverlap	584
tileScale	584
transparencyLevel	585
usePrinterCRD	585
useT1Conversion	586
RDN	587
ReadStream	589
Rendition	590
Rendition のプロパティ	590
altText	590
doc	590
fileName	591
type	591
uiName	592
Rendition のメソッド	592
getPlaySettings	592
select	593
testCriteria	594
Report	595
Report のプロパティ	595
absIndent	595
color	595
size	596
style	596
Report のメソッド	597
breakPage	597
divide	597
indent	597
mail	598
open	598
outdent	599
Report	599
save	599
writeText	600
Row	602
ScreenAnnot	603
ScreenAnnot のプロパティ	603
altText	603
alwaysShowFocus	603
display	604
doc	604

2 JavaScript API (続き)

ScreenAnnot (続き)	
ScreenAnnot のプロパティ (続き)	
events	604
extFocusRect	605
innerDeviceRect	605
noTrigger	606
outerDeviceRect	606
page	607
player	607
rect	607
ScreenAnnot のメソッド	608
hasFocus	608
setFocus	608
search	609
search のプロパティ	609
attachments	609
available	609
bookmarks	610
docInfo	610
docText	610
docXMP	611
ignoreAccents	611
ignoreAsianCharacterWidth	611
indexes	612
jpegExif	612
legacySearch	612
markup	613
matchCase	613
matchWholeWord	613
maxDocs	614
objectMetadata	614
proximity	614
proximityRange	615
refine	615
soundex	615
stem	616
thesaurus	616
wordMatching	617
search のメソッド	617
addIndex	617
getIndexForPath	618
query	618
removeIndex	619
security	620
security の定数	620
security のプロパティ	621
handlers	621
validateSignaturesOnOpen	622

2 JavaScript API (続き)

security (続き)	
security のメソッド	623
chooseRecipientsDialog	623
chooseSecurityPolicy	625
exportToFile	626
getHandler	626
getSecurityPolicies	627
importFromFile	629
SecurityHandler	630
SecurityHandler のプロパティ	630
appearances	630
digitalIDs	631
directories	632
directoryHandlers	632
docDecrypt	633
docEncrypt	633
isLoggedIn	633
loginName	634
loginPath	634
name	634
signAuthor	635
signFDF	635
signInvisible	635
signValidate	636
signVisible	636
uiName	636
validateFDF	637
SecurityHandler のメソッド	637
login	637
logout	640
newDirectory	640
newUser	641
setPasswordTimeout	642
SecurityPolicy	644
SecurityPolicy のプロパティ	644
SignatureInfo	645
SignatureInfo のプロパティ	645
SOAP	654
SOAP のプロパティ	654
wireDump	654
SOAP のメソッド	654
connect	654
queryServices	657
resolveService	659
request	661
response	669
streamDecode	670
streamDigest	671
streamEncode	671

2 JavaScript API (続き)

SOAP (続き)	
SOAP のメソッド (続き)	
streamFromString	672
stringFromStream	672
Sound	673
Sound のプロパティ	673
name	673
Sound のメソッド	673
pause	673
play	673
stop	673
Span	674
Span のプロパティ	674
alignment	674
fontFamily	674
fontStretch	675
fontStyle	675
fontWeight	676
strikethrough	676
subscript	676
superscript	676
text	677
textColor	678
textSize	678
underline	678
spell	680
spell のプロパティ	680
available	680
dictionaryNames	680
dictionaryOrder	681
domainNames	681
languages	682
languageOrder	683
spell のメソッド	684
addDictionary	684
addWord	684
check	685
checkText	686
checkWord	686
customDictionaryClose	687
customDictionaryCreate	688
customDictionaryDelete	689
customDictionaryExport	689
customDictionaryOpen	690
ignoreAll	691
removeDictionary	691
removeWord	692
userWords	692

2 JavaScript API (続き)

Statement	694
Statement のプロパティ	694
columnCount	694
rowCount	694
Statement のメソッド	695
execute	695
getColumn	695
getColumnArray	696
getRow	696
nextRow	697
TableInfo	699
Template	700
Template のプロパティ	700
hidden	700
name	700
Template のメソッド	701
spawn	701
Thermometer	703
Thermometer のプロパティ	703
cancelled	703
duration	704
text	704
value	704
Thermometer のメソッド	705
begin	705
end	705
this	706
TTS	708
TTS のプロパティ	708
available	708
numSpeakers	708
pitch	709
soundCues	709
speaker	709
speechCues	710
speechRate	710
volume	710
TTS のメソッド	710
getNthSpeakerName	710
pause	711
qSilence	711
qSound	711
qText	712
reset	712
resume	712
stop	712
talk	712

2 JavaScript API (続き)	
util	713
util のメソッド	713
crackURL	713
iconStreamFromIcon	714
printd	715
printf	717
printx	719
scand	720
spansToXML	721
streamFromString	721
stringFromStream	722
xmlToSpans	723
XFA	724
XMLData	725
XMLData のメソッド	725
applyXPath	725
parse	729
3 新しい機能および変更	733
Acrobat 8.0 での変更	733
Acrobat 7.0.5 での変更	736
Acrobat 7.0 での変更	737
Acrobat 7.0 での新しい機能	738
Acrobat 7.0 での変更	740
Acrobat 6.0 での変更	742
Acrobat 6.0 での新しい機能	742
Acrobat 6.0 での変更	749
Acrobat 6.0 での非推奨	750
Acrobat 6.0.2 での新しい機能	751
Acrobat 5.0 での変更	757
Acrobat 5.0 での新しい機能	757
Acrobat 5.0 での変更	764
Acrobat 5.0 での非推奨	764
Acrobat 5.05 での変更	765
Adobe Reader 5.1 での変更	765

はじめに

このマニュアルでは、Adobe® Acrobat® Professional、Acrobat Standard、Adobe Reader® の JavaScript™ 拡張機能として使用できるオブジェクト、プロパティ、メソッドについて説明します。

このマニュアルの内容

このマニュアルでは、Acrobat の JavaScript API について説明します。

- [35 ページの「JavaScript API」](#) では、JavaScript API について詳しく説明します。すべてのオブジェクト、プロパティ、メソッドについて説明するとともに、多数のコード例を紹介します。
- [733 ページの「新しい機能および変更」](#) では、最近のバージョンの Acrobat で導入された新機能や変更点について説明します。

注意：JavaScript インタプリタでは、ここに記載されていないプロパティやメソッドがいくつか使用できますが、それらの記載されていないプロパティやメソッドは使用しないでください。それらのサポートは行われておらず、いつでも予告なく変更される可能性があります。

対象読者

このマニュアルでは、読者がコア JavaScript 1.6 に習熟していることを前提としています。対象読者としては、インタラクティブな PDF 文書の作成者、インテリジェントな文書を設計するフォームデザイナー、Acrobat プラグインの開発者などが含まれます。

Acrobat ユーザインターフェイスに関する知識は必須です。PDF ファイル形式に関する知識があれば役に立ちます。

Acrobat には、ADBC、マルチメディア、SOAP、XML、様々なセキュリティプロトコルなどの追加機能がありますが、JavaScript を使用してこれらの機能を制御するには、その技術に関する知識が必要です。

関連ドキュメント

このマニュアルでは、JavaScript や関連テクノロジーに関して、次のリソースを参照しています。Acrobat のマニュアルは、Acrobat Developer Center (http://www.adobe.com/go/acrobat_developer) から入手できます。

知りたい情報	参照するマニュアル
Acrobat SDK のマニュアルに関する説明	『Acrobat SDK Documentation Roadmap』
既知の問題点や実装の詳細	『Readme』
Acrobat SDK でよくある質問への回答	『Developer FAQ』
このリリースの Acrobat SDK の新機能	『What's New』
Acrobat SDK の概要	『Overview』
Acrobat SDK で Adobe Reader 向けの開発を行う方法	『Developing for Adobe Reader』

知りたい情報	参照するマニュアル
Acrobat SDK に付属するサンプルコードの説明	『Guide to SDK Samples』
DDE、OLE、Apple イベント、AppleScript を使用して Acrobat や Adobe Reader を制御したり、PDF 文書をレンダリングしたりする方法	『Developing Applications Using Interapplication Communication』
Acrobat や Adobe Reader を制御したり、PDF 文書をレンダリングしたりするために使用する DDE、OLE、Apple イベント、AppleScript API の詳細	『Interapplication Communication API Reference』
PDF 文書の 3D 注釈にインタラクティブな機能を追加する JavaScript API の詳細	『JavaScript for Acrobat 3D Annotations API Reference』
JavaScript を使用して Acrobat や Adobe Reader に機能を追加する方法	『Developing Acrobat Applications Using JavaScript』
常時接続でない環境で RSS を使用してリモートリソースを追跡する方法	『Acrobat Tracker』
pdfmark と呼ばれる PostScript® 言語の拡張機能を利用して、PostScript ファイル内に PDF 固有の機能を記述する技術の詳細	『pdfmark Reference』
PDF ファイル形式の詳細	『PDF Reference』
JavaScript を使用して複数のファイルに反復処理を行う方法	『Batch Sequences』
URL に埋め込まれたコマンドを使用して、PDF ファイルを開いたり PDF ファイルでアクションを実行するときに使用するパラメータの詳細	『Parameters for Opening PDF Files』
XFA の仕様	『XFA Specification』
データのフォーマットや解析に使用するパターンを記述するための固有の言語に関する説明	『XFA-Picture Clause 2.0 Specification』
XFDF の仕様	『XML Form Data Format Specification』
XML フォームオブジェクトモデルを使用してスクリプトを作成するときに利用できる様々なモデル（各モデルに用意されている様々なオブジェクトや、そのプロパティやメソッドの詳細）	『Adobe XML Form Object Model Reference』
Acrobat フォームを Adobe LiveCycle® Designer フォームに変換する方法や、Acrobat と LiveCycle Designer でのフォームオブジェクトモデルの違い	『Converting Acrobat JavaScript for Use in LiveCycle Designer Forms』
証明済み文書、署名可能なフォーム、独自のワークフローや外観を作成するために使用する、Acrobat の電子署名機能に関する説明	『Acrobat 8.0 Security Feature User Reference』

Adobe Acrobat 製品ファミリー (Acrobat Professional、Acrobat Standard、Adobe Reader) では、クロスプラットフォーム対応のスクリプト言語として JavaScript がサポートされています。文書作成者、フォームデザイナー、プラグイン開発者は、JavaScript の拡張機能を使用して、ビューアーアプリケーションやプラグインの様々な機能にアクセスできます。

この機能を使用すれば、次のことが行えます。

- 文書内でのフォーム処理
- 複数の PDF 文書のバッチ処理
- オンライン共同作業スキームの開発や保守
- ローカルデータベースとのやり取り
- マルチメディアイベントの制御

Acrobat の JavaScript 拡張機能として用意されているオブジェクト、プロパティ、メソッドは、Acrobat や Adobe Reader で使用できるのはもちろん、PDF 文書の処理を自動化する Microsoft Visual Basic プログラムからもアクセスできます。詳しくは、『Interapplication Communication API Reference』を参照してください。

構文

JavaScript には、指定のスペルを入力してそのまま使用できる静的なオブジェクトがいくつか用意されています。例えば、app オブジェクトは JavaScript アプリケーションを表します。このオブジェクトは 1 つのみ存在しており、app というスペルで表記する必要があります（大文字と小文字は区別されます）。

それ以外のオブジェクトは、変数に割り当てることができる動的なオブジェクトです。例えば、Doc オブジェクトを取得して変数に割り当てることができます。

```
var myDoc = app.newDoc();
```

この myDoc を使用すれば、Doc オブジェクトのすべてのメソッドやプロパティにアクセスできます。例えば、次のようにになります。

```
myDoc.closeDoc();
```

メソッドの引数

Acrobat に用意されている多くの JavaScript メソッドでは、通常の JavaScript のように引数のリストを指定することもできますし、単一のオブジェクト引数を使用してそのプロパティで引数を指定することもできます。例えば、次の 2 つの呼び出しが同じ意味になります。

```
app.alert( "Acrobat Multimedia", 3);  
app.alert({ cMsg: "Acrobat Multimedia", nIcon: 3});
```

注意：マルチメディアをサポートする JavaScript メソッドでは、1 つの形式でしか引数を指定できません。メソッドの説明で示されている引数の形式を使用してください。

パラメータヘルプ

Acrobat Professional を使用している場合は、JavaScript Debugger コンソール（または任意の内部 JavaScript エディタ）で引数に acrohelp を指定して Acrobat メソッドを実行すると、そのメソッドの引数が返されます。

例えば、コンソールウィンドウで次のコードを入力します。

```
app.response(acrohelp)
```

入力した行にカーソルがある状態で、Ctrl+Enter キーか、数字キーパッドの Enter キーを押すと、コンソールに次のメッセージが表示されます。

```
HelpError: Help.  
app.response:1:Console undefined:Exec  
===== [cQuestion: string]  
===== [cTitle: string]  
===== [cDefault: string]  
===== [bPassword: boolean]  
===== [cLabel: string]
```

角っこで囲まれているパラメータは、オプションのパラメータです。

注意：パラメータヘルプは、すべての JavaScript メソッドに用意されているわけではありません。例えば、App という JavaScript フォルダで定義されているメソッドには用意されていません。

パス

いくつかのメソッドでは、引数としてデバイスに依存しないパスを指定することができます。デバイスに依存しないパスの形式について詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

セーフパス

Acrobat 6.0 では、パラメータで渡されたパスに基づいてローカルハードドライブにデータを書き込む JavaScript メソッドに対して、セーフパスという概念が導入されました。

システムの重要なフォルダであるルートディレクトリ、Windows ディレクトリ、システムディレクトリなどを参照するパスは使用できません。また、その他の様々な条件にも従う必要があります。

例えば、多くのメソッドでは、保存するデータのタイプに適した拡張子をファイル名に付ける必要があります。また、ファイルの上書きが不可能なメソッドもあります。これらの制限について詳しくは、このマニュアル内を参照してください。

通常、パスが安全でないと見なされると、NotAllowedError 例外が発生し ([Error オブジェクトを参照](#))、メソッドは失敗します。

セキュリティによる制限がないコンテキスト

通常は制限されている操作に対して、実行権限が付与されるコンテキストのことです。この権限は、特定の方法で（コンソールやバッチ処理で）メソッドを実行したり、特定の PDF プロパティが設定されている場合に付与されます。また、信頼できる人によって文書が署名されている場合にも付与されます。例えば、JavaScript の実行を許可する証明書によって文書が証明されており、その証明書をユーザが信頼している場合、その文書は JavaScript の実行が制限されないコンテキストと見なされ、通常は行えない JavaScript の実行が行えます。

セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト

このマニュアルのクイックバーの3列目には **S** と表示されている JavaScript メソッドは、セキュリティによって制限されています。このようなメソッドは、コンソールイベント、バッチイベント、アプリケーション初期化イベントなどの、セキュリティによる制限がないコンテキストでのみ実行できます。他のすべてのイベント（ページを開くイベントや、マウスボタンを放すイベントなど）は、セキュリティによる制限があるコンテキストと見なされます。

セキュリティによる制限があるメソッドについては、そのメソッドを実行できるイベントがこのリファレンスマニュアルで説明されています。

Acrobat 6.0 以降では、埋め込まれている特権の高い JavaScript の実行を許可する証明書によって文書が証明されており、その証明書をユーザが信頼している場合は、セキュリティによる制限があるメソッドを実行できます。

7.0 より前のバージョンの Acrobat では、メニューイベントはセキュリティによる制限がないコンテキストと見なされていました。Acrobat 7.0 以降では、メニューイベントでの JavaScript の実行にも、セキュリティによる制限が加えられました。セキュリティによる制限があるメソッドをメニューイベントで実行するには、次のいずれかの方法を使用します。

- Acrobat 環境設定の「JavaScript」分類を開き、「メニュー項目の JavaScript 実行権限を有効にする」にチェックを付けます。
- Acrobat 7.0 で導入された信頼済み関数を使用して、メソッドを実行します。信頼済み関数を使用すれば、セキュリティによる制限があるコード（セキュリティによる制限がないコンテキストでしか通常は実行できないコード）からその制限が解除され、セキュリティによる制限があるコンテキストでも実行できるようになります。詳しい説明と例については、[app.trustedFunction](#) を参照してください。

ユーザ環境設定

このマニュアルでは、多くの箇所で、Acrobat のユーザ環境設定について言及しています。環境設定ダイアログボックスにアクセスするには、プラットフォームに応じて次のメニュー命令を使用します。

Microsoft® Windows® : 編集／環境設定

Macintosh : Acrobat／環境設定

環境設定ダイアログボックスは、関連するコマンドが集められた様々な分類（「フォーム」、「一般」、「JavaScript」など）で構成されています。

クイックバー

ほとんどのプロパティやメソッドの説明では、動作環境や推奨される使用法を示す小さな表（クイックバー）が冒頭に示されています。

クイックバーの例を次に示します。各列の見出しあは説明のために追加したもので、本文のクイックバーにはありません。

バージョンまたは非推奨	保存と環境設定	セキュリティ	可用性
6.0	D	S	C

次の表に、各列に表示されるマークとその意味を示します。

列 1：バージョンまたは非推奨

#.# この列の番号は、プロパティやメソッドを使用できるソフトウェアのバージョンを表します。番号が明記されている場合、プロパティやメソッドはそのバージョン以降の Acrobat でのみ使用できます。

Acrobat 8.0 については、旧バージョンとの互換性を考慮する必要があります。下位互換性があるかどうかを確認するには、プロパティやメソッドにアクセスする前に、明記されているバージョンとフォームのバージョンが等しいか、またはフォームのバージョンのほうが大きいことをスクリプトで確認する必要があります。例えば、次のようにになります。

```
if (typeof app.formsVersion != "undefined" && app.formsVersion >= 8.0)
{
    // バージョン特有の操作を行う
}
```

最初の列が空白の場合は、互換性を確認する必要はありません。

X プロパティやメソッドは非推奨です。

列 2：保存と環境設定

D このプロパティやメソッドに書き込むと、PDF 文書が変更されます。その後で文書を保存すると、メソッドで行った変更も保存されます（Adobe Reader で文書を保存するには、特別な権限が必要です）。

P このプロパティに書き込んでも文書は変更されませんが、アプリケーションの環境設定が永続的に変更される可能性があります。

例 3：セキュリティ

S セキュリティ上の理由から、このプロパティやメソッドを使用できるイベントが制限されています。使用できるイベントとしては、バッチ処理、アプリケーションの起動、コンソールでの実行があります（Acrobat のイベントについて詳しくは、`event` オブジェクトを参照）。

Acrobat 7.0 以降では、セキュリティによる制限があるメソッドをメニューイベントから実行するには、次のいずれかを行なう必要があります。

- 「JavaScript」ユーザ環境設定で、「メニュー項目の JavaScript 実行権限を有効にする」という項目にチェックを付けます。
- 信頼済み関数を使用してメソッドを実行します。詳しい説明と例については、`app.trustedFunction` メソッドを参照してください。

詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

注意： (Acrobat 6.0 以降) 埋め込まれている特權の高い JavaScript の実行を許可する証明書によって文書が証明されており、その証明書をユーザが信頼している場合は、**S** のマークが付いているメソッドであっても、制限なく実行できます（クイックバーのその他の条件が満たされている場合）。

例 4：可用性

この列が空白の場合、プロパティやメソッドは Adobe Reader、Acrobat Professional、Acrobat Standard で使用できます。

X プロパティやメソッドは、Adobe Reader では使用できませんが、Acrobat Professional と Acrobat Standard では使用できます。

F プロパティやメソッドは、Acrobat Professional と Acrobat Standard で使用できます。文書に適用されている追加の使用権限によっては、Adobe Reader バージョン 5.1 以降でアクセスできる場合があります。

- S**
- F — フォーム権限を必要とします。

D

 - C — 注釈の操作権限を必要とします。

G

 - S — 文書の保存権限を必要とします。
 - D — ファイルの添付権限を必要とします。
 - G — 電子署名の権限を必要とします。

P プロパティやメソッドは、Acrobat Professional でのみ使用できます。

コードサンプルで使用しているドメイン名

このマニュアルには、URL を使用したコードサンプルが多数掲載されています。このような例では、説明用のドメイン名として特別に予約されている `example.com`、`example.net`、`example.org` を使用しています。また、IP アドレスを使用した例では、プライベートネットワーク用のアドレスとして予約されている `172.16.0.0 ~ 172.31.255.255` の範囲を使用しています。

この章では、Acrobat で使用できる JavaScript 拡張機能と、そのオブジェクト、メソッド、プロパティについて詳しく説明します。この章は、オブジェクト名のアルファベット順に構成されています。

コア JavaScript の Acrobat 拡張機能は、Adobe Exchange 3.01 で初めて提供されました。このバージョンに「Acrobat Forms Author Plug-in 3.5 Update」をインストールすることで、JavaScript の機能を追加することができました。次の表に示すように、当初は JavaScript バージョン 1.2 が使用されていました。Acrobat 5.0 でコア JavaScript（当時のバージョンは 1.5）が大幅に拡張され、アプリケーションやプラグインの機能が幅広く利用できるようになりました。最新の Acrobat では、JavaScript 1.6 が使用されています。

Acrobat のバージョン	3.01	4.0	5.0	6.0	7.0	8.0
JavaScript のバージョン	1.2	1.2	1.5	1.5	1.5	1.6

JavaScript ソリューションを開発するときは、どのバージョンの Acrobat（または Adobe Reader）まで下位互換性を確保するかに注意してください。この表でターゲットアプリケーションを選択すれば、使用すべき JavaScript のバージョンがわかります。

JavaScript API の大半の機能は、すべてのバージョンの Acrobat や Adobe Reader で使用できますが、新しいバージョンでしか使用できない機能もいくつかあるので注意してください。また、API によって、Acrobat Professional でのみ使用できるもの、Adobe Reader では使用できないもの、Reader 用の適切な追加権限が文書に設定されている場合にのみ Adobe Reader で使用できるものなどがあるので、この点にも注意が必要です。JavaScript ソリューションを開発するときは、こうした要因をすべて考慮する必要があります。

プロパティやメソッドの説明の冒頭に表示されている記号については、[32 ページの「クイックバー」](#) を参照してください。このクイックバーには、メソッドが最初に定義されたバージョン番号、セキュリティ制限、Adobe Reader に関する制約、必要な Adobe Reader の使用権限などが表示されています。

コア JavaScript について詳しくは、Mozilla Developer Center (<http://developer.mozilla.org/en/docs/JavaScript>) を参照してください。

ADBC

5.0			X
-----	--	--	---

ADBC プラグインで提供されている一貫したオブジェクトモデルを使用すれば、PDF 文書内の JavaScript からデータベースにアクセスすることができます。ADBC は Windows 専用の機能で、クライアントマシンに ODBC がインストールされている必要があります。

このオブジェクトモデルは、ODBC API や JDBC API のオブジェクトモデルで採用されている一般的なルールに基づいて構築されており、ODBC や JDBC と同様に、SQL を介してデータベースとのやり取りが行えます。

注意： ADBC には、データベースアクセスのセキュリティ機能は用意されていません。データのセキュリティについては、データベース管理者が責任を持つ必要があります。

ADBC オブジェクトはグローバルオブジェクトであり、スクリプトでこのオブジェクトのメソッドを使用して、データベースへの接続を確立することができます。次に、データベースアクセスに使用する個々の関連オブジェクトについて説明します。

関連するオブジェクト	簡単な説明	ページ
Connection	このオブジェクトでは、接続済みのデータベースに含まれるテーブルのリストを取得できます。	206 ページ
Statement	このオブジェクトでは、SQL 文を実行し、クエリに基づいてレコードを抽出できます。	694 ページ

ADBC のプロパティ

SQL 型

5.0			X
-----	--	--	---

ADBC オブジェクトには、様々な SQL 型を表す次の定数プロパティがあります。

定数プロパティ名	値	バージョン
SQLT_BIGINT	0	
SQLT_BINARY	1	
SQLT_BIT	2	
SQLT_CHAR	3	
SQLT_DATE	4	
SQLT_DECIMAL	5	
SQLT_DOUBLE	6	
SQLT_FLOAT	7	
SQLT_INTEGER	8	

定数プロパティ名	値	バージョン
SQLT_LONGVARBINARY	9	
SQLT_LONGVARCHAR	10	
SQLT_NUMERIC	11	
SQLT_REAL	12	
SQLT_SMALLINT	13	
SQLT_TIME	14	
SQLT_TIMESTAMP	15	
SQLT_TINYINT	16	
SQLT_VARBINARY	17	
SQLT_VARCHAR	18	
SQLT_NCHAR	19	6.0
SQLT_NVARCHAR	20	6.0
SQLT_NTEXT	21	6.0

これらのプロパティは、`column` オブジェクトや `ColumnInfo` オブジェクトの `type` プロパティで使用します。

JavaScript 型

5.0			X
-----	--	--	---

ADBC オブジェクトには、JavaScript の様々なデータ型を表す次の定数プロパティがあります。

定数プロパティ名	値
Numeric	0
String	1
Binary	2
Boolean	3
Time	4
Date	5
TimeStamp	6

これらの型は、`Statement` オブジェクトの `getColumn` メソッドや `getColumnArray` メソッドで使用します。

ADBC のメソッド

getDataSourceList

5.0			X
-----	--	--	---

指定のシステムでアクセス可能なデータベースに関する情報を取得します。

戻り値

システムでアクセス可能な各データベースの `DataSourceInfo` オブジェクトを含む配列。このメソッドが失敗することはありませんが、配列長が 0 になる場合があります。

例

ADBC.[newConnection](#) を参照してください。

newConnection

5.0		S	X
-----	--	---	---

指定のデータベースに対する `Connection` オブジェクトを作成します。オプションで、ユーザ ID とパスワードを指定することもできます。

注意：(Acrobat 6.0 以降) データソース名 (DSN) のない接続文字列を使用してデータベースに接続することは可能ですが、この処理が行えるのは Acrobat 6.0 以降のコンソールイベントまたはバッチイベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。

パラメータ

cDSN データベースのデータソース名 (DSN)

cUID (オプション) ユーザ ID

cPWD (オプション) パスワード

戻り値

`Connection` オブジェクト。失敗した場合は `null`。

例

システムで利用可能な `DataSourceInfo` オブジェクトの配列を取得して、q32000data という名前のデータソースを探しながら、それらをコンソールに表示します。

```
/* まずシステムで利用可能な DataSourceInfo オブジェクトの配列を取得 */
var aList = ADBC.getDataSourceList();
console.show(); console.clear();
```

```
try {
    /* 「q32000data」という名前のオブジェクトを探しながら
     * それを表示。*/
    var DB = "", msg = "";
    if (aList != null) {
        for (var i=0; i < aList.length; i++) {
            console.println("Name: "+aList[i].name);
            console.println("Description: "+aList[i].description);
            // 目的のオブジェクトを選択。
            if (aList[i].name=="q32000data")
                DB = aList[i].name;
        }
    }

    // データベースを見つけたか？
    if (DB != "") {
        // 見つけたので接続を確立。
        console.println("The requested database has been found!");
        var Connection = ADBC.newConnection(DB);
        if (Connection == null) throw "Not Connected!";
        } else
            // 見つけていないのでコンソールにメッセージを表示。
            throw "Could not find the requested database.";
    } catch (e) {
        console.println(e);
    }

    // または、単純に直接接続してもよい。
    var Connection = ADBC.newConnection("q32000data");
}
```

Alerter

7.0			
-----	--	--	--

Acrobat のマルチメディアプラグインは、存在しないメディアファイルがある場合など、様々な状況でエラー警告を表示します。JavaScript コードを使用すると、文書全体や個別のメディアプレーヤーの警告をカスタマイズできます。

警告が表示されるときには、`app.media.alert` という内部関数が呼び出され、警告に関する情報がパラメータとして渡されます。この `app.media.alert` メソッドは、次の順序で `alerter` オブジェクトを検索し、その `dispatch` メソッドを呼び出して警告を処理します。

```
args.alerter
doc.media.alerter
doc.media.stockAlerter
```

特定のプレーヤーの警告をカスタマイズするには、`app.media.createPlayer` や `app.media.openPlayer` を呼び出すときに、`args.alerter` に `alerter` オブジェクトを割り当てます。

文書全体の警告をカスタマイズするには、`doc.media.alerter` に `alerter` オブジェクトを割り当てます。

プレーヤーまたは文書の警告をすべて抑制するには、`args.alerter` または `doc.media.alerter` を `null` に設定します。

カスタムの `alerter` が指定されていない場合は、`doc.media.stockAlerter` のデフォルトの警告が使用されます。このプロパティは、`app.media.alert` によって自動的に初期化されます。通常、開発者が `doc.media.stockAlerter` にアクセスする必要はありません。

Alerter のメソッド

dispatch

7.0			
-----	--	--	--

警告状況を処理するために `app.media.alert` によって呼び出されます。

パラメータ

`alert` [Alert オブジェクト](#) (以下を参照)。

戻り値

ブーリアン値。警告処理を停止する場合は `true`、処理を継続する場合は `false`。

Alert オブジェクト

プロパティ	型	説明
type	文字列	すべての警告タイプ
doc	Doc オブジェクト	すべての警告タイプ
fromUser	ブーリアン	すべての警告タイプ
error	オブジェクト	Exception タイプの警告に使用可能。この error オブジェクトには message というプロパティがあります。 error: { message: String }
errorText	文字列	PlayerError タイプの警告に使用可能。
fileName	文字列	FileNotFoundException タイプの警告に使用可能。
selection	MediaSelection オブジェクト	SelectFailed タイプの警告に使用可能。

例 1

メディアプレーヤーを開いて、このプレーヤーのすべての警告を抑制します。

```
app.media.openPlayer({ alerter: null });

// 同じことを行うための、さらに複雑な方法
app.media.openPlayer(
{
  alerter:
  {
    dispatch() { return true; }
  }
});
```

例 2

この文書内のすべてのプレーヤーについて、すべての警告をテキストフィールドに記録し、通常の警告ボックスを表示します。

```
function logAlerts( doc )
{
  count = 0;
  doc.alerter =
  {
    dispatch( alert )
    {
      doc.getField("AlertLog").value += "Alert #"
      + ++count + ": " + alert.type + "\n";
    }
  }
  logAlerts( this );
}
```

```
// カウンタを維持する別の方法
function logAlerts( doc )
{
    doc.alerter =
    {
        count = 0,
        dispatch( alert )
        {
            doc.getField("AlertLog").value += "Alert #"
                + ++this.count + ": " + alert.type + "\n";
        }
    }
    logAlerts( this );
}
```

例 3

ここで PlayerError 警告を処理し、別の警告のデフォルトを使用します。

```
this.media.alerter =
{
    dispatch( alert )
    {
        switch( alert.type )
        {
            case "PlayerError":
                app.alert( "Player error: " + alert.errorText );
                return true;
        }
    }
}
```

AlternatePresentation

このオブジェクトは、文書の特定の代替プレゼンテーションに対するインターフェイスを提供します。AlternatePresentation オブジェクトを取得するには、Doc オブジェクトの [alternatePresentations](#) プロパティを使用します。

代替プレゼンテーションについて詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

AlternatePresentation のプロパティ

active

6.0			
-----	--	--	--

プレゼンテーションが現在アクティブである場合は `true`、アクティブでない場合は `false` を返します。どのプレゼンテーションがアクティブであるかによって、プレゼンテーションを所有している文書の表示が制御されます。

型

ブーリアン

アクセス

R

例

例については、[start](#) メソッドを参照してください。

type

6.0			
-----	--	--	--

代替プレゼンテーションのタイプ。現在サポートされているタイプは「SlideShow」のみです。

型

文字列

アクセス

R

AlternatePresentation のメソッド

start

6.0			
-----	--	--	--

文書のビューを代替プレゼンテーションモードに切り替え、active プロパティを true に設定します。いずれかの代替プレゼンテーション（この代替プレゼンテーションを含む）が既にアクティブになっているときにこのメソッドを呼び出すと、例外が発生します。

パラメータ

cOnStop	(オプション) 何らかの理由（stop の呼び出し、明示的なユーザーアクション、プレゼンテーションロジックなど）によってプレゼンテーションが完了したときに、Acrobat で評価する式。
cCommand	(オプション) 代替プレゼンテーションに渡すコマンドまたはスクリプト。 注意： これはプレゼンテーション固有のコマンドです（JavaScript の式ではありません）。

例

「MySlideShow」という名前のプレゼンテーションが文書に存在しているとします。

```
// oMySlideShow は AlternatePresentation オブジェクト
oMySlideShow = this.alternatePresentations.MySlideShow;
if (!oMySlideShow.active) oMySlideShow.start();
```

指定のプレゼンテーションにプロパティ名でアクセスするために this.alternatePresentations を使用している点に注目してください。

stop

6.0			
-----	--	--	--

プレゼンテーションを停止して、文書を標準（PDF）プレゼンテーションに切り替えます。このプレゼンテーションがアクティブでない場合は、例外が発生します。

例

この例では、oMySlideShow という AlternatePresentations オブジェクトを使用しています。関連する例については、[start](#) を参照してください。

```
// 既にアクティブならスライドショーを停止
if (oMySlideShow.active) oMySlideShow.stop();
```

Annotation

このオブジェクトは、Acrobat の注釈を表します。注釈を作成するには、Acrobat の注釈ツールを使用するか、Doc オブジェクトの `addAnnot` メソッドを使用します。

注釈にアクセスするには、Doc オブジェクトの `getAnnot` などのメソッドを使用して、JavaScript 変数に注釈を割り当てる必要があります。

```
var a = this.getAnnot(0, "Important");
```

これによって、最初のページ（ページは 0 から数えます）にある「Important」という注釈を、変数 `a` を通じて操作できるようになります。例えば、次のコードでは、この注釈のタイプを変数 `thetype` に格納し、作成者を「John Q. Public」に変更しています。

```
var thetype = a.type;           // プロパティを読み取る
a.author = "John Q. Public"; // プロパティを書き込む
```

Annotation オブジェクトにアクセスする別の方法として、Doc オブジェクトの `getAnnots` メソッドを使用する方法があります。

注意：Adobe Reader 5.1 以降では、`contents` 以外のすべての注釈プロパティの値を取得できます。それらのプロパティを設定するためには、文書に注釈権限が付与されている必要があります（クリックバーに  のアイコンが示されています）。

英語版の Acrobat ユーザインターフェイスでは、様々なタイプの注釈（annotations）を総称して Comments と呼んでいます。

注釈タイプ

注釈には様々なタイプがあり、`type` プロパティに反映されています。次の表に、それぞれのタイプを示します。また、このマニュアルで説明されている、`getProps` メソッドで返されるプロパティをすべて列挙します。

注釈タイプ	プロパティ
Caret	<code>author</code> 、 <code>borderEffectIntensity</code> 、 <code>borderEffectStyle</code> 、 <code>caretSymbol</code> 、 <code>contents</code> 、 <code>creationDate</code> 、 <code>delay</code> 、 <code>hidden</code> 、 <code>inReplyTo</code> 、 <code>intent</code> 、 <code>lock</code> 、 <code>modDate</code> 、 <code>name</code> 、 <code>noView</code> 、 <code>opacity</code> 、 <code>page</code> 、 <code>popupOpen</code> 、 <code>popupRect</code> 、 <code>print</code> 、 <code>readOnly</code> 、 <code>rect</code> 、 <code>refType</code> 、 <code>richContents</code> 、 <code>rotate</code> 、 <code>seqNum</code> 、 <code>strokeColor</code> 、 <code>style</code> 、 <code>subject</code> 、 <code>toggleNoView</code> 、 <code>type</code> 、 <code>width</code>
Circle	<code>author</code> 、 <code>borderEffectIntensity</code> 、 <code>borderEffectStyle</code> 、 <code>contents</code> 、 <code>creationDate</code> 、 <code>dash</code> 、 <code>delay</code> 、 <code>fillColor</code> 、 <code>hidden</code> 、 <code>inReplyTo</code> 、 <code>intent</code> 、 <code>lock</code> 、 <code>modDate</code> 、 <code>name</code> 、 <code>noView</code> 、 <code>opacity</code> 、 <code>page</code> 、 <code>popupOpen</code> 、 <code>popupRect</code> 、 <code>print</code> 、 <code>readOnly</code> 、 <code>rect</code> 、 <code>refType</code> 、 <code>richContents</code> 、 <code>rotate</code> 、 <code>seqNum</code> 、 <code>strokeColor</code> 、 <code>style</code> 、 <code>subject</code> 、 <code>toggleNoView</code> 、 <code>type</code> 、 <code>width</code>
FileAttachment	<code>attachIcon</code> 、 <code>author</code> 、 <code>borderEffectIntensity</code> 、 <code>borderEffectStyle</code> 、 <code>contents</code> 、 <code>creationDate</code> 、 <code>delay</code> 、 <code>hidden</code> 、 <code>inReplyTo</code> 、 <code>intent</code> 、 <code>lock</code> 、 <code>modDate</code> 、 <code>name</code> 、 <code>noView</code> 、 <code>opacity</code> 、 <code>page</code> 、 <code>point</code> 、 <code>print</code> 、 <code>readOnly</code> 、 <code>rect</code> 、 <code>refType</code> 、 <code>richContents</code> 、 <code>rotate</code> 、 <code>seqNum</code> 、 <code>strokeColor</code> 、 <code>style</code> 、 <code>subject</code> 、 <code>toggleNoView</code> 、 <code>type</code> 、 <code>width</code>

注釈タイプ	プロパティ
FreeText	alignment 、 author 、 borderEffectIntensity 、 borderEffectStyle 、 callout 、 contents 、 creationDate 、 dash 、 delay 、 fillColor 、 hidden 、 inReplyTo 、 intent 、 lineEnding 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 richDefaults 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 textFont 、 textSize 、 toggleNoView 、 type 、 width
Highlight	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 quads 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Ink	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 dash 、 delay 、 gestures 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Line	arrowBegin 、 arrowEnd 、 author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 dash 、 delay 、 doCaption 、 fillColor 、 hidden 、 inReplyTo 、 intent 、 leaderExtend 、 leaderLength 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 points 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Polygon	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 dash 、 delay 、 fillColor 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 vertices 、 width
PolyLine	arrowBegin 、 arrowEnd 、 author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 dash 、 delay 、 fillColor 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 vertices 、 width
Sound	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 point 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 soundIcon 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Square	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 dash 、 delay 、 fillColor 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width

注釈タイプ	プロパティ
Squiggly	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 quads 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Stamp	AP 、 author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type
StrikeOut	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 quads 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Text	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 noteIcon 、 opacity 、 page 、 point 、 popupOpen 、 popupRect 、 print 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 state 、 stateModel 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width
Underline	author 、 borderEffectIntensity 、 borderEffectStyle 、 contents 、 creationDate 、 delay 、 hidden 、 inReplyTo 、 intent 、 lock 、 modDate 、 name 、 noView 、 opacity 、 page 、 popupOpen 、 popupRect 、 print 、 quads 、 readOnly 、 rect 、 refType 、 richContents 、 rotate 、 seqNum 、 strokeColor 、 style 、 subject 、 toggleNoView 、 type 、 width

Annotation のプロパティ

『PDF Reference』バージョン 1.7 には、注釈のプロパティとその保存形式がすべて記載されています。

PDF 文書に保存されるプロパティ値には、名前として保存されるものと、文字列として保存されるものがあります（詳しくは、『PDF Reference』バージョン 1.7 を参照）。名前として保存されるプロパティは、文字の長さが 127 文字までに制限されています。

127 文字の制限があるプロパティとしては、AP、beginArrow、endArrow、attachIcon、noteIcon、soundIcon などがあります。

alignment

5.0	D		C
-----	---	--	---

FreeText 注釈のテキストの整列を制御します。

整列	値
左揃え	0
中央揃え	1
右揃え	2

型

数値

アクセス

R / W

注釈

FreeText

AP

5.0	D		C
-----	---	--	---

スタンプ注釈の表示に使用するスタンプの外観の名前。標準のスタンプ注釈の名前は、次のとおりです。

Approved
AsIs
Confidential
Departmental
Draft
Experimental
Expired
Final
ForComment
ForPublicRelease
NotApproved
NotForPublicRelease
Sold
TopSecret

型

文字列

アクセス

R / W

注釈

Stamp

例

プログラムでスタンプ注釈を追加します。

```
var annot = this.addAnnot ({
    page: 0,
    type: "Stamp",
    author: "A. C. Robat",
    name: "myStamp",
    rect: [400, 400, 550, 500],
    contents: "Try it again, this time with order and method!",
    AP: "NotApproved" });
```

注意：特定のスタンプの名前を確認したい場合は、`Stamps` フォルダにある、目的のスタンプが含まれている PDF ファイルを開いてください。文書で現在使用されているスタンプ名のリストについては、Doc オブジェクトの [icons](#) プロパティを参照してください。

arrowBegin

5.0			
-----	--	--	--

線注釈の始点に使用する形状を設定します。有効な値は、次のとおりです。

```
None (デフォルト)
OpenArrow
ClosedArrow
ROpenArrow      // Acrobat 6.0
RClosedArrow    // Acrobat 6.0
Butt           // Acrobat 6.0
Diamond
Circle
Square
Slash          // Acrobat 7.0
```

型

文字列

アクセス

R / W

注釈

Line、PolyLine

例

[setProps](#) メソッドを参照してください。

arrowEnd

5.0	D		C
-----	---	--	---

線注釈の終点に使用する形状を設定します。有効な値は、次のとおりです。

```
None ( デフォルト )
OpenArrow
ClosedArrow
ROpenArrow      // Acrobat 6.0
RClosedArrow    // Acrobat 6.0
Butt           // Acrobat 6.0
Diamond
Circle
Square
Slash          // Acrobat 7.0
```

型

文字列

アクセス

R / W

注釈

Line、 PolyLine

例

[setProps](#) メソッドを参照してください。

attachIcon

5.0	D		C
-----	---	--	---

注釈の表示に使用するアイコンの名前。認識される値は、次のとおりです。

```
Paperclip
PushPin ( デフォルト )
Graph
Tag
```

型

文字列

アクセス

R / W

注釈

FileAttachment

author

5.0	D		C
-----	---	--	---

注釈の作成者を取得または設定します。

型

文字列

アクセス

R / W

注釈

すべて

例

[contents](#) プロパティを参照してください。

borderEffectIntensity

6.0	D		C
-----	---	--	---

境界線の効果を使用している場合の、効果の強度。雲型の長方形、多角形、橢円などの、雲形の大きさを表します。

型

数値

アクセス

R / W

注釈

すべて

borderEffectStyle

6.0	D		C
-----	---	--	---

境界線の効果のスタイル名（または空の文字列）。現在サポートされている境界線の効果は、空の文字列（なし）または「C」（雲型）のみです。

型

文字列

アクセス

R / W

注釈

すべて

callout

7.0	D		C
-----	---	--	---

テキストボックス注釈の引き出し線を指定する、4つまたは6つの数字の配列。詳しくは、『PDF Reference』バージョン1.7を参照してください。

型

配列

アクセス

R / W

注釈

FreeText

caretSymbol

6.0	D		C
-----	---	--	---

挿入記号注釈に関連付ける記号。テキスト編集の存在を示す視覚的な記号です。有効な値は、空の文字列（なし）、「P」（パラグラフ記号）、「S」（スペース記号）です。

型

文字列

アクセス

R / W

注釈

Caret

contents

5.0	D		C
-----	---	--	---

ポップアップウィンドウを持つ注釈の内容にアクセスします。音声注釈やファイル添付注釈の場合は、説明として表示されるテキストを表します。

型

文字列

アクセス

R / W

注釈

すべて

例

テキスト注釈を作成し、作成者と内容を指定します。

```
var annot = this.addAnnot({  
    page: 0,  
    type: "Text",  
    point: [400,500],  
    author: "A. C. Robat",  
    contents: "Call Smith to get help on this paragraph.",  
    noteIcon: "Help"  
});
```

Doc オブジェクトの [addAnnot](#) メソッドも参照してください。

creationDate

6.0			C
-----	--	--	---

注釈が作成された日時。

型

日付

アクセス

R

注釈

すべて

dash

5.0	D		C
-----	---	--	---

線と間隔のパターンを表す破線配列。破線の境界線を描画するために使用されます。例えば、[3, 2] という値を指定すると、3 ポイントの線と 2 ポイントの間隔が交互に描画されます。

破線配列を設定するには、style プロパティが D に設定されている必要があります。

型

配列

アクセス

R / W

注釈

FreeText、Line、PolyLine、Polygon、Circle、Square、Ink

例

この例では、境界を破線に変更します。annot は Annotation オブジェクトであると仮定しています。

```
annot.setProps({ style: "D", dash: [3,2] });
```

[delay](#) プロパティの例も参照してください。

delay

5.0			C
-----	--	--	---

true の場合、注釈のプロパティに対する変更はキューに入れられ、delay が false に設定されたときに実行されます（Field オブジェクトの delay プロパティに似ています）。

型

ブーリアン

アクセス

R / W

注釈

すべて

例

このコードでは、境界を破線に変更します。annot は Annotation オブジェクトであると仮定しています。

```
annot.delay=true;  
annot.style = "D";  
annot.dash = [4,3];  
annot.delay = false;
```

doc

5.0			C
-----	--	--	----------

この注釈が含まれている文書の Doc オブジェクト。

型

Doc オブジェクト

アクセス

R

注釈

すべて

例

注釈を作成して、その doc プロパティを使用します。

```
var inch = 72;  
var annot = this.addAnnot({  
    page: 0,  
    type: "Square",  
    rect: [1*inch, 3*inch, 2*inch, 3.5*inch]  
});  
/* 例えば "file:///C|/Adobe/Annots/myDoc.pdf" などを表示 */  
console.println(annot.doc.URL);
```

doCaption

7.0	D		C
-----	----------	--	----------

true の場合、線の外観にリッチコンテンツを表示します。

型

ブーリアン

アクセス

R / W

注釈

Line

例

[65 ページの points プロパティの例も参照してください。](#)

fillColor

5.0	D		C
-----	----------	--	----------

楕円、長方形、線、多角形、折れ線、テキストボックスの各注釈の背景色を設定します。値の定義には、 transparent、gray、RGB または CMYK カラーを使用します。カラー配列の定義や、このプロパティでの値の使用方法について詳しくは、[191 ページの「カラー配列」](#)を参照してください。

型

カラー

アクセス

R / W

注釈

Circle、Square、Line、Polygon、PolyLine、FreeText

例

橢円注釈を作成し、その背景色を設定します。

```
var annot = this.addAnnot(  
{  
    type: "Circle",  
    page: 0,  
    rect: [200,200,400,300],  
    author: "A. C. Robat",  
    name: "myCircle",  
    popupOpen: true,  
    popupRect: [200,100,400,200],  
    contents: "Hi World!",  
    strokeColor: color.red,  
    fillColor: ["RGB",1,1,.855]  
});
```

gestures

5.0			
-----	--	--	--

ストロークするパスを表す配列の配列。各配列には、デフォルトユーザースペースにおけるパス上の点を表す x 座標と y 座標が格納されています。描画時には、実装に応じた方法で、これらの点が直線または曲線で結ばれます。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

型

配列

アクセス

R / W

注釈

Ink

hidden

5.0			
-----	--	--	--

true の場合、注釈は表示されなくなり、ユーザは注釈を操作、表示、印刷できなくなります。

型

ブーリアン

アクセス

R / W

注釈

すべて

inReplyTo

6.0			
-----	--	--	--

空でない場合、この注釈の応答先である注釈の name 値を示します。

型

文字列

アクセス

R / W

注釈

すべて

intent

7.0			
-----	--	--	--

同じタイプのマークアップ注釈でも、このプロパティ（注釈の使用目的）の設定によって動作が変わります。例えば、引き出し線注釈は、intent が FreeTextCallout に設定されたテキストボックス注釈です。

このプロパティはすべての注釈で定義されていますが、intent に空以外の値を設定できるのは、現在のところ、テキストボックス注釈、多角形注釈、線注釈のみです。

型

文字列

アクセス

R / W

注釈

すべて

UI に用意されている、特別な外観の注釈を作成するためのツールを次の表に示します。

UI	注釈タイプ	Intent
引き出し線ツール	FreeText	FreeTextCallout
雲型ツール	Polygon	PolygonCloud
矢印ツール	Line	LineArrow
寸法線ツール	Line	LineDimension

leaderExtend

7.0	D		C
-----	----------	--	----------

線の両端から線に対して垂直方向に伸びた、補助線の延長部分の長さを指定します。補助線の延長部分は、補助線と逆の方向 (180° の方向) に伸びます。値は、常に 0 以上である必要があります。

デフォルトは 0 です (補助線の延長部分なし)。

型

数値

アクセス

R / W

注釈

Line

leaderLength

7.0	D		C
-----	----------	--	----------

線の両端から線に対して垂直方向に伸びた、補助線の長さを指定します。この値をマイナスにすると、補助線の伸びる方向が逆になります。

デフォルトは 0 です (補助線なし)。

型

数値

アクセス

R / W

注釈

Line

lineEnding

7.0			
-----	--	--	--

このプロパティは、引き出し線の線端の形状を決定します。このプロパティは、`intent` の値が `FreeTextCallout` であるテキストボックス注釈でのみ意味を持ちます。認識される値は、次のとおりです。

```
None (デフォルト)  
OpenArrow  
ClosedArrow  
ROpenArrow // Acrobat 6.0  
RClosedArrow // Acrobat 6.0  
Butt // Acrobat 6.0  
Diamond  
Circle  
Square  
Slash // Acrobat 7.0
```

型

文字列

アクセス

R / W

注釈

FreeText

lock

5.0			
-----	--	--	--

`true` の場合、注釈はロックされています。UI のプロパティダイアログボックスから注釈にアクセスできる点以外は、`readOnly` と同じです。

型

ブーリアン

アクセス

R / W

注釈

すべて

modDate

5.0			C
-----	--	--	---

注釈の最終更新日。

型

日付

アクセス

R / W

注釈

すべて

例

更新日をコンソールに出力します。

```
console.println(util.printd("mmmm dd, yyyy", annot.modDate));
```

name

5.0	D		C
-----	---	--	---

注釈の名前。Doc オブジェクトの `getAnnot` メソッドにこの値を指定すれば、注釈を取得してそのプロパティやメソッドにアクセスできるようになります。

型

文字列

アクセス

R / W

注釈

すべて

例

myNote という注釈を取得して、コメントを追加します。

```
var gannot = this.getAnnot(0, "myNote");
gannot.contents += "¥r¥rDon't forget to check with Smith";
```

notelcon

5.0			
-----	--	--	--

注釈の表示に使用するアイコンの名前。認識される値は、次のとおりです。

```
Check
Circle
Comment
Cross
Help
Insert
Key
NewParagraph
Note ( デフォルト )
Paragraph
RightArrow
RightPointer
Star
UpArrow
UpLeftArrow
```

型

文字列

アクセス

R / W

注釈

Text

例

[contents](#) プロパティを参照してください。

noView

5.0	D		C
-----	---	--	---

true の場合、注釈は非表示になりますが、注釈に外観がある場合、その外観を印刷することはできます。

型

ブーリアン

アクセス

R / W

注釈

すべて

例

[toggleNoView](#) プロパティを参照してください。

opacity

5.0	D		C
-----	---	--	---

注釈の描画に使用する不透明度の値。この値は、閉じている注釈のすべての可視要素（背景や境界線も含む）に適用されます。注釈を開いたときに表示されるポップアップウィンドウには適用されません。有効な値は、0.0 ~ 1.0 です。値を 0.5 にすると、注釈が半透明になります。

型

数値

アクセス

R / W

注釈

すべて

page

5.0			
-----	--	--	--

注釈が存在するページ。

型

整数

アクセス

R / W

注釈

すべて

例

次のコードでは、Annotation オブジェクトである `annot` を、現在のページからページ 3（ページ番号は 0 から数えます）に移動します。

```
annot.page = 2;
```

point

5.0			
-----	--	--	--

テキスト注釈、音声注釈、ファイル添付注釈のアイコンの、デフォルトユーザースペースにおける左上隅の座標を表す 2 つの数値の配列 `[xul, yul]`。

型

配列

アクセス

R / W

注釈

Text、Sound、FileAttachment

例

ヘルプアイコンを使用したノート注釈を、指定の座標に配置します。アイコンはポップアップノートの左上隅に配置されます。

```
var annot = this.addAnnot({
    page: 0,
    type: "Text",
    point: [400, 500],
    contents: "Call Smith to get help on this paragraph.",
    popupRect: [400, 400, 550, 500],
    popupOpen: true,
    noteIcon: "Help"
});
```

[noteIcon](#) プロパティと、Doc オブジェクトの [addAnnot](#) メソッドも参照してください。

points

5.0	④		⑤
-----	---	--	---

デフォルトユーザースペースにおける線の始点と終点の座標を表す、2 つの点の配列 $[[x_1, y_1], [x_2, y_2]]$ 。

型

配列

アクセス

R / W

注釈

Line

例

指定の 2 点間に線を引きます。

```
var annot = this.addAnnot({
    type: "Line",
    page: 0,
    author: "A. C. Robat",
    doCaption: true,
    contents: "Look at this again!",
    points: [[10, 40], [200, 200]],
});
```

[arrowBegin](#) プロパティ、[arrowEnd](#) プロパティ、[setProps](#) メソッド、Doc オブジェクトの [addAnnot](#) メソッドも参照してください。

popupOpen

5.0	D		C
-----	---	--	---

true の場合、注釈が存在するページが表示されると、テキストノートがポップアップして開きます。

型

ブーリアン

アクセス

R / W

注釈

FreeText、Sound、FileAttachment を除くすべて

例

[print](#) プロパティを参照してください。

popupRect

5.0	D		C
-----	---	--	---

親注釈に関連付けられているポップアップ注釈の矩形を表す、4つの数値の配列 [x_ll, y_ll, x_ur, y_ur]。これらの数値は、デフォルトユーザースペースにおける矩形の左下隅の x 座標、左下隅の y 座標、右上隅の x 座標、右上隅の y 座標を表します。これによって、ポップアップ注釈のページ上の位置が定義されます。

型

配列

アクセス

R / W

注釈

FreeText、Sound、FileAttachment を除くすべて

例

[print](#) プロパティを参照してください。

print

5.0	D		C
-----	---	--	---

注釈を印刷するか (`true`)、印刷しないか (`false`) を示します。

型

ブーリアン

アクセス

R / W

注釈

すべて

quads

5.0	D		C
-----	---	--	---

デフォルトユーザースペースにおける n 個の四角形の座標を表す、 $8 \times n$ の数値配列。各四角形は、注釈を構成する 1 つ以上の連続する単語を囲みます。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。1 つの単語の `quads` は、Doc オブジェクトの `getPageNthWordQuads` メソッドを呼び出して取得できます。

型

配列

アクセス

R / W

注釈

Highlight、Strikeout、Underline、Squiggly

例

Doc オブジェクトの [`getPageNthWordQuads`](#) メソッドを参照してください。

rect

5.0	D		C
-----	---	--	---

rect 配列は、ページ上の注釈の位置を定義する矩形を表す、4つの数値の配列 [x_ll, y_ll, x_ur, y_ur] です。これらの数値は、デフォルトユーザースペースにおける矩形の左下隅の x 座標、左下隅の y 座標、右上隅の x 座標、右上隅の y 座標を表します。[popupRect](#) プロパティも参照してください。

型

配列

アクセス

R / W

注釈

すべて

readOnly

5.0	D		C
-----	---	--	---

true の場合、注釈は表示専用となり、変更などは行えなくなります。

型

ブーリアン

アクセス

R / W

注釈

すべて

refType

7.0	D		C
-----	---	--	---

注釈の参照タイプ。このプロパティは、inReplyTo が単純なスレッドディスカッションの関係であるか、またはグループ関係であるかを示します。認識される値は、「R」と「Group」です。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

型

文字列

アクセス

R / W

注釈

すべて

richContents

6.0	D		C
-----	---	--	---

このプロパティは、注釈のテキストコンテンツと書式を取得します。リッチテキストコンテンツは、注釈のテキストコンテンツと書式が格納された `Span` オブジェクトの配列として表されます。

型

`Span` オブジェクトの配列

アクセス

R / W

注釈

Sound、FileAttachment を除くすべて

例

テキスト注釈を作成し、その注釈にリッチテキストコンテンツを追加します。

```
var annot = this.addAnnot ({
    page: 0,
    type: "Text",
    point: [72, 500],
    popupRect: [72, 500, 6*72, 500-2*72],
    popupOpen: true,
    noteIcon: "Help"
}) ;

var spans = new Array();
spans[0] = new Object();
spans[0].text = "Attention: ¥r";
spans[0].textColor = color.blue;
spans[0].textSize = 18;
```

```
spans[1] = new Object();
spans[1].text = "Adobe Acrobat 6.0¥r";
spans[1].textColor = color.red;
spans[1].textSize = 20;
spans[1].alignment = "center";

spans[2] = new Object();
spans[2].text = "will soon be here!";
spans[2].textColor = color.green;
spans[2].fontStyle = "italic";
spans[2].underline = true;
spans[2].alignment = "right";

// リッチフィールドにリッチテキストを指定
annot.richContents = spans;
```

Span オブジェクトの使用例については、Field オブジェクトの [richValue](#) メソッドと、event オブジェクトの [richValue](#)、[richChange](#)、[richChangeEx](#) の各メソッドも参照してください。

richDefaults

6.0	D		C
-----	----------	--	----------

このプロパティは、テキストボックス注釈のデフォルトのスタイル属性を定義します。詳しくは、Field オブジェクトの [defaultStyle](#) プロパティの説明を参照してください。

型

Span オブジェクト

アクセス

R / W

注釈

FreeText

rotate

5.0	D		C
-----	----------	--	----------

ページ上の注釈を反時計回りに回転させる角度（0、90、180、270）。このプロパティは、テキストボックス注釈でのみ意味を持ちます。

型

整数

アクセス

R / W

注釈

FreeText

seqNum

5.0			C
-----	--	--	---

ページ上にある注釈のシーケンス番号。読み取り専用です。

型

整数

アクセス

R

注釈

すべて

soundIcon

5.0	D		C
-----	---	--	---

音声注釈の表示に使用するアイコンの名前。有効な値は、「Speaker」です。

型

文字列

アクセス

R / W

注釈

Sound

state

6.0	D		C
-----	----------	--	----------

テキスト注釈の状態。このプロパティの値は、stateModel に依存します。Marked 状態モデルの場合、有効な値は Marked と Unmarked です。Review 状態モデルの場合、有効な値は Accepted、Rejected、Cancelled、Completed、None です。

型

文字列

アクセス

R / W

注釈

テキスト

stateModel

6.0	D		C
-----	----------	--	----------

Acrobat 6.0 から、作成者が独自の状態を注釈に関連付けられるようになりました。この状態は、IRT エントリで元の注釈を参照している別のテキスト注釈で指定します ([inReplyTo](#) プロパティを参照)。状態モデルには、「Marked」と「Review」の2種類が用意されています。

型

文字列

アクセス

R / W

注釈

Text

[getStateInModel](#) メソッドも参照してください。

strokeColor

5.0	D		C
-----	----------	--	----------

注釈の表示色を設定します。値の定義には、transparent、gray、RGB または CMYK カラーを使用します。テキストボックス注釈の場合、strokeColor は境界線とテキストの色を設定します。カラー配列の定義や、このプロパティでの値の使用方法について詳しくは、[191 ページの「カラー配列」](#) を参照してください。

型

カラー

アクセス

R / W

注釈

すべて

例

テキスト注釈を赤色に変更します。

```
var annot = this.addAnnot({type: "Text"});  
annot.strokeColor = color.red;
```

style

5.0			
-----	--	--	--

このプロパティは、境界線スタイルを取得および設定します。有効な値は、`s`（実線）と`d`（破線）です。`style` プロパティはすべての注釈で定義されていますが、有効に機能するのは、テキストボックス、楕円、長方形、折れ線、多角形、鉛筆の各注釈のみです。

型

文字列

アクセス

R / W

注釈

すべて

例については、[dash](#) プロパティを参照してください。

subject

6.0			
-----	--	--	--

注釈内容の簡潔な説明を示すテキスト。テキストは、ポップアップウィンドウのタイトルバーがあればそこに表示されるか、または、プロパティダイアログボックスに表示されます。

型

文字列

アクセス

R / W

注釈

すべて

textFont

5.0	D		C
-----	----------	--	----------

テキストボックス注釈に入力されたテキストのフォントを設定します。有効なフォントは、`font` オブジェクトのプロパティとして定義されています (Field オブジェクトの [textFont](#) プロパティを参照)。

テキストボックス注釈で使用するフォントを指定するには、`textFont` にフォントの PostScript 名を指定します。

型

文字列

アクセス

R / W

注釈

FreeText

例

Helvetica フォントを使用してテキストボックス注釈を作成します。

```
var annot = this.addAnnot({
  page: 0,
  type: "FreeText",
  textFont: font.Helv, // または、textFont: "Viva-Regular",
  textSize: 10,
  rect: [200, 300, 200+150, 300+3*12], // 3 つの行が収まる高さ
  width: 1,
  alignment: 1});
```

textSize

5.0			
-----	--	--	--

テキストボックス注釈のテキストサイズ（ポイント単位）。有効なテキストサイズは、0 または 4 ~ 144 の範囲です。0 は、すべてのテキストが注釈の矩形に収まる最大のポイントサイズを表します。

型

数値

アクセス

R / W

注釈

FreeText

例

[textFont](#) プロパティを参照してください。

toggleNoView

6.0			
-----	--	--	--

true の場合、注釈の上にマウスが置かれるか、または注釈が選択されると、[noView](#) フラグが切り替わります。

注釈の noView フラグと toggleNoView フラグを両方設定すると、通常の状況では注釈が表示されず、注釈の上にマウスが置かれるか、注釈が選択された場合にのみ表示されるようになります。

型

ブーリアン

アクセス

R / W

注釈

すべて

type

5.0			Ⓐ
-----	--	--	---

注釈のタイプ。注釈のタイプを設定できるのは、Doc オブジェクトの [addAnnot](#) メソッドでオブジェクトリテラル引数を指定するときのみです。有効な値は、次のとおりです。

Text
FreeText
Line
Square
Circle
Polygon
PolyLine
Highlight
Underline
Squiggly
StrikeOut
Stamp
Caret
Ink
FileAttachment
Sound

型

文字列

アクセス

R

注釈

すべて

vertices

6.0	Ⓓ		Ⓒ
-----	---	--	---

多角形注釈または折れ線注釈の、デフォルトユーザースペースにおける各頂点の座標を表す配列の配列。各座標は、水平座標、垂直座標の順に示されています。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

型

配列の配列

アクセス

R / W

注釈

Polygon、PolyLine

width

5.0			
-----	--	--	--

境界線の幅（ポイント単位）。この値が 0 の場合、境界線は描かれません。デフォルト値は 1 です。

型

数値

アクセス

R / W

注釈

Square、Circle、Line、Ink、FreeText

Annotation のメソッド

destroy

5.0			
-----	--	--	--

注釈を破棄してページから削除します。オブジェクトは無効になります。

例

ページ 0 の「FreeText」注釈をすべて削除します。

```
var annots = this.getAnnots({ nPage:0 });
for (var i = 0; i < annots.length; i++)
    if (annots[i].type == "FreeText") annots[i].destroy();
```

getProps

5.0			
-----	--	--	--

注釈のプロパティのコレクションを取得します。これは、注釈をコピーする場合に便利です。

戻り値

注釈のプロパティのオブジェクトリテラル。オブジェクトリテラルとは、Doc オブジェクトの addAnnot メソッドの引数と同じ形式で表される汎用オブジェクトのことです。

例 1

特定の注釈を文書のすべてのページにコピーします。

```
var annot = this.addAnnot ({
    page: 0,
    type: "Text",
    rect: [40, 40, 140, 140]
});

// 注釈のプロパティのコピーを作成
var copy_props = annot.getProps();

// 同じプロパティを持つ新規注釈を各ページに作成
var numpages = this.numPages;
for (var i=0; i < numpages; i++) {
    var copy_annot = this.addAnnot(copy_props);
    // ページ i に移動
    copy_annot.page=i;
}
```

例 2

注釈のプロパティとその値をすべて表示します。

```
var a = this.getAnnots(0); // ページ 0 の注釈をすべて取得
if ( a != null ) {
    var p = a[0].getProps(); // 最初の注釈のプロパティを取得
    for ( o in p ) console.println( o + " : " + p[o] );
}
```

getStateInModel

6.0			
-----	--	--	--

状態モデルにおける、注釈の現在の状態を取得します。[transitionToState](#) メソッドも参照してください。

パラメータ

cStateModel	注釈の状態を決定する状態モデル。
-------------	------------------

戻り値

注釈の現在の状態を示す識別子の配列を返します。

- 状態モデルが排他的に定義されている場合、状態は 1 つのみです（状態が設定されていない場合は、状態はありません）。
- 状態モデルが排他的でない場合は、複数の状態が存在する可能性があります（状態が設定されておらず、デフォルトもない場合、エントリはありません）。

例

この文書のすべてのページにあるすべての注釈について、そのステータスを報告します。

```
annots = this.getAnnots()
for ( var i= 0; i< annots.length; i++) {
    states = annots[i].getStateInModel("Review");
    if ( states.length > 0 ) {
        for(j = 0; j < states.length; j++)
        {
            var d = util.printd(2, states[j].modDate);
            var s = states[j].state;
            var a = states[j].author;

            console.println(annots[i].type + ": " + a + " "
                + s + " " + d + "on page "
                + (annots[i].page+1) );
        }
    }
}
```

setProps

5.0	D		C
-----	---	--	---

注釈の複数のプロパティを同時に設定します。

パラメータ

オブジェクトリテラル 作成する Annotation オブジェクトのプロパティ (type、rect、page など) を指定する汎用オブジェクト。Doc オブジェクトの addAnnot メソッドに指定するパラメータと同じです。

戻り値

Annotation オブジェクト

例

線注釈の様々なプロパティを設定します。

```
var annot = this.addAnnot({type: "Line"})
annot.setProps({
    page: 0,
    points: [[10,40], [200,200]],
    strokeColor: color.red,
    author: "A. C. Robat",
    contents: "Check with Jones on this point.",
    popupOpen: true,
    popupRect: [200, 100, 400, 200], // 矢印の先端に矩形を配置
    arrowBegin: "Diamond",
    arrowEnd: "OpenArrow"
});
```

transitionToState

6.0			
-----	--	--	--

注釈の状態を cState に移行します。状態の移行は、注釈の監査トレイルに記録されます。

[getStateInModel](#) メソッドも参照してください。

注意：マルチユーザ環境で状態が正常に機能するためには、すべてのユーザが同じ状態モデル定義を所有している必要があります。したがって、状態モデル定義は、すべてのユーザに配布したり、すべてのシステムにインストールしたりできるように、フォルダレベルの JavaScript ファイルに配置してください。

パラメータ

cStateModel	状態の移行に使用する状態モデル。cStateModel は、Collab の addStateModel メソッドを呼び出してあらかじめ追加しておく必要があります。
cState	移行後の状態。状態モデルで定義されている有効な状態であることが必要です。

例

カスタムの状態を定義して、注釈の状態を設定します。

```
try {
    // 文書を作成
    var myDoc = app.newDoc();
    // 注釈を作成
    var myAnnot = myDoc.addAnnot
    ({
        page: 0,
        type: "Text",
        point: [300,400],
        name: "myAnnot",
    });
    // 状態モデルを作成
    var myStates = new Object();
    myStates["initial"] = {cUIName: "Haven't reviewed it"};
    myStates["approved"] = {cUIName: "I approve"};
    myStates["rejected"] = {cUIName: "Forget it"};
    myStates["resubmit"] = {cUIName: "Make some changes"};
    Collab.addStateModel({
        cName: "ReviewStates",
        cUIName: "My Review",
        oStates: myStates,
        cDefault: "initial"
    });
} catch(e) { console.println(e); }
// 状態を変更
myAnnot.transitionToState("ReviewStates", "resubmit");
myAnnot.transitionToState("ReviewStates", "approved");
```

Annot3D

Annot3D オブジェクトは、特定の Acrobat 3D 注釈、つまり、Acrobat 3D ツールを使用して作成された注釈を表します。Annot3D オブジェクトは、Doc オブジェクトの [getAnnot3D](#) メソッドや [getAnnot3Ds](#) メソッドを使用して取得できます。

Annot3D のプロパティ

activated

7.0			
-----	--	--	--

注釈を 3D アートワークで表示するか (`true`)、単にポスター ボード画像で表示するか (`false`) を表すブーリアン値。

[context3D](#) プロパティを参照してください。

型

ブーリアン

アクセス

R / W

context3D

7.0			
-----	--	--	--

`activated` が `true` の場合、3D 注釈のコンテキスト（3D シーンを含む `global` オブジェクト）を返します（詳しくは、『JavaScript for Acrobat 3D Annotations API Reference』を参照）。`activated` が `false` の場合は、`undefined` を返します。

型

`global` オブジェクト

アクセス

R

innerRect

7.0			
-----	--	--	--

3D 注釈の 3D バウンディングボックス（3D アートワークが描画される領域）を表す、4 つの数字の配列 [x_{ll} , y_{ll} , x_{ur} , y_{ur}]。これらの数値は、注釈の座標系（左下隅が [0, 0]、右上隅が [高さ, 幅]）におけるバウンディングボックスの左下隅の x 座標、左下隅の y 座標、右上隅の x 座標、右上隅の y 座標を表します。

型

配列

アクセス

R

name

7.0			
-----	--	--	--

注釈の名前。

型

文字列

アクセス

R

page

7.0			
-----	--	--	--

注釈が含まれているページの番号（0 から数えます）。

型

整数

アクセス

R

rect

7.0			
-----	--	--	--

ページ上の注釈の位置を定義する矩形を表す、4つの数字の配列 $[x_{ll}, y_{ll}, x_{ur}, y_{ur}]$ を返します。これらの数値は、デフォルトユーザースペースにおける矩形の左下隅の x 座標、左下隅の y 座標、右上隅の x 座標、右上隅の y 座標を表します。

型

配列

アクセス

R / W

app

Acrobat アプリケーションを表す静的な JavaScript オブジェクト。このオブジェクトには、Acrobat 固有の関数に加えて、様々なユーティリティ関数や便利な関数が用意されています。

app のプロパティ

activeDocs

5.0			
-----	--	--	--

アクティブな各文書の Doc オブジェクトを含む配列。アクティブな文書が存在しない場合、`activeDocs` は何も返しません。これは、コア JavaScript の `d = new Array(0)` と同じ動作になります。

7.0 より前のバージョンの Acrobat では、コンソールで `d = app.activeDocs` というスクリプトを実行すると、`[object Global]` が返されていました。Acrobat 7.0 以降では、コンソールに `toString()` 値は出力されなくなりました。

注意：バージョンに関連する次の情報に注意してください。

- Acrobat 5.0 では、このプロパティは、アクティブな各文書の Doc オブジェクトを含む配列を返します。
- Acrobat 5.0.5 では、このプロパティに変更が加えられており、開いている文書のうち、Doc オブジェクトの `disclosed` プロパティが `true` に設定されている文書のみを対象にして、その Doc オブジェクトの配列を返します。
- 「アクセシビリティおよびフォーム機能のパッチ」を適用した Acrobat 5.0.5 と、Acrobat 6.0 以降の `activeDocs` は、バッヂ、コンソール、メニューイベントでは `disclosed` プロパティを無視して、アクティブな文書の Doc オブジェクトの配列を返します。その他のイベントでは、アクティブな文書のうち、`disclosed` が `true` に設定されている文書のみを対象として、その Doc オブジェクトの配列を返します。
- Acrobat 7.0 以降では、メニューイベントでの JavaScript の実行にも、セキュリティによる制限が加えられました。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

`app.activeDocs` が返す配列には、`app.openDoc` で `bHidden` パラメータを `true` に設定して開かれた文書がすべて含まれます。これは、前述のセキュリティ制限を受けます。

型

配列

アクセス

R

例

この例では、開いている文書の中から「myDoc」というタイトルの文書を検索し、Doc オブジェクトの `addField` メソッドを使用してこの文書にボタンを挿入します。この文書が `disclosed` であるべきかどうかは、このコードを実行する Acrobat のバージョンや、コードを実行する場所（コンソールか、「マウスボタンを放す」アクションかなど）によって決まります。

```
var d = app.activeDocs;
for (var i=0; i < d.length; i++)
if (d[i].info.Title == "myDoc") {
    var f = d[i].addField("myButton", "button", 0 , [20, 100, 100, 20]);
    f.setAction("MouseUp", "app.beep(0)");
    f.fillColor=color.gray;
}
```

calculate

X			
---	--	--	--

`true`（デフォルト値）の場合、計算を実行できます。`false` の場合、計算は許可されません。

このプロパティの使用は推奨されません。Doc オブジェクトの `calculate` プロパティを使用してください。

型

ブーリアン

アクセス

R / W

constants

7.0			
-----	--	--	--

様々な定数値を保持するためのラッパー オブジェクト。現在、このプロパティは、`align` という単一のプロパティを持つオブジェクトを返します。

`app.constants.align` には、整列のタイプを表す `left`、`center`、`right`、`top`、`bottom` というプロパティがあります。これらの値は、透かしを追加するときなどに、整列を指定するために使用できます。

型

オブジェクト

アクセス

R

例

例については、Doc オブジェクトの [addWatermarkFromFile](#) メソッドおよび [addWatermarkFromText](#) メソッドを参照してください。

focusRect

4.05			
------	--	--	--

フォーカスを示す矩形のオン／オフを切り替えます。フォーカスを示す矩形とは、ボタン、チェックボックス、ラジオボタン、署名を囲む薄い点線のことです、フォームフィールドにキーボードフォーカスがあることを示します。値が `true` の場合、フォーカスを示す矩形はオンになります。

型

ブーリアン

アクセス

R / W

例

フォーカスを示す矩形をオフにします。

```
app.focusRect = false;
```

formsVersion

4.0			
-----	--	--	--

ビューアフォームソフトウェアのバージョン番号。スクリプトの下位互換性を確保したい場合は、このプロパティを調べて、新しいバージョンのソフトウェアで導入されたオブジェクト、プロパティ、メソッドが使用できるかどうかを判断します。

型

数値

アクセス

R

例

```
if (typeof app.formsVersion != "undefined" && app.formsVersion >= 5.0)
{
    // ここでバージョン固有の処理を行う
    // 例えば、フルスクリーンモードへの切り替えなど
    app.fs.cursor = cursor.visible;
    app.fs.defaultTransition = "";
    app.fs.useTimer = false;
    app.fs.isFullScreen = !app.fs.isFullScreen;
}
else app.fullscreen = !app.fullscreen;
```

fromPDFConverters

6.0			
-----	--	--	--

ファイル形式変換 ID 文字列の配列。変換 ID 文字列は Doc オブジェクトの `saveAs` メソッドに渡すことができます。

型

配列

アクセス

R

例

現在サポートされているすべての変換 ID 文字列をリストします。

```
for ( var i = 0; i < app.fromPDFConverters.length; i++)
    console.println(app.fromPDFConverters[i]);
```

fs

5.0			
-----	--	--	--

FullScreen オブジェクト。これを使用して、フルスクリーンのプロパティにアクセスすることができます。

型

オブジェクト

アクセス

R

例

```
// このコードは、ビューアをフルスクリーン（プレゼンテーション）モードする。  
app.fs.isFullScreen = true;  
  
fullScreen.isFullScreen も参照してください。
```

fullscreen

X			
---	--	--	--

このプロパティの使用は推奨されません。代わりに `FullScreen` オブジェクトの `isFullScreen` プロパティを使用してください。`fs` メソッドは、`FullScreen` オブジェクトを返します。これを使用して、フルスクリーンのプロパティにアクセスできます。

ビューアをフルスクリーンモードにするか、通常の表示モードにするかを制御します。

注意：Web ブラウザに表示されている PDF 文書をフルスクリーンモードにすることはできません。ブラウザからフルスクリーンモードを起動することはできますが、ブラウザ自体にフルスクリーンモードは適用されません。Acrobat ビューアアプリケーションで開かれている文書があれば、その文書にのみフルスクリーンモードが適用されます。

型

ブーリアン

アクセス

R / W

language

3.01			
------	--	--	--

動作中の Acrobat ビューアで使用されている言語。次のいずれかの文字列になります。

文字列	言語
CHS	簡体字中国語
CHT	繁体字中国語
DAN	デンマーク語
DEU	ドイツ語
ENU	英語
ESP	スペイン語
FRA	フランス語
ITA	イタリア語

文字列	言語
KOR	韓国語
JPN	日本語
NLD	オランダ語
NOR	ノルウェー語
PTB	ポルトガル語（ブラジル）
SUO	フィンランド語
SVE	スウェーデン語

型

文字列

アクセス

R

media

6.0			
-----	--	--	--

マルチメディアプレーヤーの設定や制御に役立つ様々なプロパティやメソッドが用意されています。

このオブジェクトのプロパティやメソッドの一覧と、様々な使用例については、[app.media](#) オブジェクトを参照してください。

型

オブジェクト

アクセス

R

monitors

6.0			
-----	--	--	--

Monitors オブジェクト。これは、ユーザのシステムに接続されているモニタを表す 1 つ以上の Monitor オブジェクトの配列です。app.monitors にアクセスするたびに、この配列の最新のコピーが新たに返されます。

Monitors オブジェクトには、モニタを選択するためのメソッドもいくつか用意されています。または、JavaScript コードでこの配列の内容を明示的に調べることもできます。

型

Monitors オブジェクト

アクセス

R

例

ユーザのシステムに接続されているモニタの数をカウントします。

```
var monitors = app.monitors;  
console.println("There are " + monitors.length  
+ " monitor(s) connected to this system.");
```

numPlugIns

X			
---	--	--	--

注意：このメソッドの代わりに plugIns プロパティを使用してください。

Acrobat にロードされているプラグインの数を示します。

型

数値

アクセス

R

openInPlace

4.0	P		
-----	---	--	--

文書間のリンクが、同じウィンドウで開かれるか新しいウィンドウで開かれるかを示します。

型

ブーリアン

アクセス

R / W

例

文書間のリンクを同じウィンドウで開きます。

```
app.openInPlace = true;
```

platform

4.0			
-----	--	--	--

スクリプトが実行されているプラットフォーム。有効な値は、次の 3 つです。

WIN
MAC
UNIX

型

文字列

アクセス

R

plugins

5.0			
-----	--	--	--

ビューアに現在インストールされているプラグインを表す `PlugIn` オブジェクトの配列。

型

配列

アクセス

R

例

```
// PlugIn オブジェクトの配列を取得
var aPlugins = app.plugins;
// プラグインの数を取得
var nPlugins = aPlugins.length;
// すべてのプラグインの名前を列挙
for ( var i = 0; i < nPlugins; i++)
    console.println("Plugin #" + i + " is " + aPlugins[i].name);
```

printColorProfiles

6.0			X
-----	--	--	---

使用可能なプリンタカラースペースのリスト。それぞれの値は、`PrintParams` オブジェクトの `colorProfile` プロパティの値として使用できます。

型

文字列の配列

アクセス

R

例

使用可能なプリンタカラースペースのリストを出力します。

```
var l = app.printColorProfiles.length
for ( var i = 0; i < l; i++)
    console.println("(" + (i+1) + ") " + app.printColorProfiles[i]);
```

printerNames

6.0			
-----	--	--	--

使用可能なプリンタのリスト。それぞれの値は、`PrintParams` オブジェクトの `printerName` プロパティで使用できます。システムにプリンタがインストールされていない場合は、空の配列が返されます。

型

文字列の配列

アクセス

R

例

使用可能なプリンタのリストを出力します。

```
var l = app.printerNames.length
for ( var i = 0; i < l; i++)
    console.println("(" + (i+1) + ") " + app.printerNames[i]);
```

runtimeHighlight

6.0	P		
-----	----------	--	--

true の場合、フォームフィールドの背景色とホバーカラー（マウスポインタが範囲内に入ったときの色）が表示されます。

型

ブーリアン

アクセス

R / W

例

実行時の強調表示がオフ (false) の場合は何もせず、そうでない場合は設定を変更します。

```
if (!app.runtimeHighlight)
{
    app.runtimeHighlight = true;
    app.runtimeHighlightColor = color.red;
}
```

runtimeHighlightColor

6.0	P		
-----	----------	--	--

実行時のフォームフィールドの強調表示色を設定します。

このプロパティの値はカラー配列です（詳しくは、[191 ページの「カラー配列」](#)を参照）。

型

カラー配列

アクセス

R / W

例

[runtimeHighlight](#) プロパティを参照してください。

thermometer

6.0			
-----	--	--	--

Thermometer オブジェクトはステータスウィンドウとプログレスバーを組み合わせたもので、時間のかかる処理が進行中であることを示します。

型

オブジェクト

アクセス

R

例

[Thermometer](#) オブジェクトを参照してください。

toolbar

3.01	P		
------	----------	--	--

Acrobat の水平方向および垂直方向のツールバーの両方を、スクリプトで表示したり隠したりできます。外部ウィンドウ（Web ブラウザ内の Acrobat ウィンドウ）のツールバーを隠すことはできません。

型

ブーリアン

アクセス

R / W

例

ツールバーを非表示にします。

```
app.toolbar = false;
```

toolbarHorizontal

X	P		
----------	----------	--	--

注意：このプロパティは、Acrobat 5.0 以降で非推奨になりました。このプロパティは toolbar と同様に動作します。

Acrobat の水平方向のツールバーをスクリプトで表示したり隠したりできます。外部ウィンドウ（Web ブラウザ内の Acrobat ウィンドウ）のツールバーを隠すことはできません。

型

ブーリアン

アクセス

R / W

toolbarVertical

X	P		
---	---	--	--

注意：このプロパティは、Acrobat 5.0 以降で非推奨になりました。このプロパティは toolbar と同様に動作します。

Acrobat の垂直方向のツールバーをスクリプトで表示したり隠したりできます。外部ウィンドウ（Web ブラウザ内の Acrobat ウィンドウ）のツールバーを隠すことはできません。

型

ブーリアン

アクセス

R / W

viewerType

3.01			
------	--	--	--

動作中のビューアアプリケーションを表す文字列。次のいずれかの値になります。

値	説明
Reader	Acrobat Reader（バージョン 5.0 まで）または Adobe Reader（バージョン 5.1 以降）
Exchange	Adobe Acrobat（バージョン 6.0 よりも前）または Acrobat Standard（バージョン 6.0 以降）
Exchange-Pro	Acrobat Professional（バージョン 6.0 以降）

型

文字列

アクセス

R

viewerVariation

5.0			
-----	--	--	--

動作中のビューアアプリケーションの機能パッケージを示します。次のいずれかの値になります。

Reader
Fill-In
Business Tools
Full

型

文字列

アクセス

R

viewerVersion

4.0			
-----	--	--	--

現在のビューアアプリケーションのバージョン番号を示します。

型

数値

アクセス

R

app のメソッド

addMenuItem

5.0		⌚	
-----	--	---	--

メニューにメニュー項目を追加します。

注意：このメソッドを実行できるのは、アプリケーション初期化イベントまたはコンソールイベントのみです。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

[addSubMenu](#)、[execMenuItem](#)、[hideMenuItem](#)、[listMenuItems](#) の各メソッドも参照してください。

パラメータ

cName	言語に依存しない、メニュー項目の名前。この名前は、他のメソッド (<code>hideMenuItem</code> など) でこのメニュー項目にアクセスする際に使用できます。
cUser	(オプション) ユーザインターフェイスにメニュー項目の名前として表示する文字列（言語に依存する名前）。 <code>cUser</code> を指定しない場合は、 <code>cName</code> が使用されます。
cParent	親メニュー項目の名前。新しいメニュー項目は、このサブメニューに追加されます。 <code>cParent</code> にサブメニューが存在しない場合は、例外が発生します。 メニュー項目名は、 <code>listMenuItems</code> メソッドを使用して取得できます。
nPos	(オプション) メニュー項目を追加するサブメニュー内の位置。デフォルトではサブメニューの末尾に追加されます。 <code>nPos</code> に 0 を指定するとサブメニューの先頭に追加されます。Acrobat 6.0 以降では、言語に依存しないメニュー項目名を <code>nPos</code> の値として指定できるようになりました。 (Acrobat 6.0) <code>nPos</code> に文字列の値を指定すると、メニュー項目の名前（言語に依存しないメニュー項目名）を指定したものと解釈されます。新しいメニュー項目の挿入位置は、この項目名に基づいて決定されます。詳しくは、 bPrepend を参照してください。 アルファベット順に整列が行われるメニューでは、 <code>nPos</code> パラメータは無視されます。アルファベット順に整列が行われるメニューは、次のとおりです。 <ul style="list-style-type: none">表示／ナビゲーションパネルの最初のセクション。表示／ツールバーの最初のセクション。アドバンストサブメニューの最初のセクション。 <p>注意：<code>nPos</code> が数値の場合、ツールメニューでは <code>nPos</code> は無視されます。ツールメニューに追加したメニュー項目は、メニューの先頭に挿入されます。<code>nPos</code> によって挿入位置が決まるのは、<code>nPos</code> が別のユーザ定義のメニュー項目を参照する文字列である場合です。</p>
cExec	ユーザがメニュー項目を選択したときに評価する式を表す文字列。 注意： Acrobat 7.0 以降では、メニューイベントでの JavaScript の実行にも、セキュリティによる制限が加えられました。詳しくは、 32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」 を参照してください。
cEnable	(オプション) メニュー項目を使用可能にするかどうかを制御する式を表す文字列。デフォルトでは、メニュー項目は常に使用可能になります。メニュー項目を使用不可にするには、この式で <code>event.rc</code> を <code>false</code> に設定します。
cMarked	(オプション) メニュー項目の横にチェックを付けるかどうかを制御する式を表す文字列。メニュー項目のチェックを解除するには、この式で <code>event.rc</code> を <code>false</code> に設定します。チェックを付けるには <code>true</code> に設定します。デフォルトでは、メニュー項目にマークは付きません。
bPrepend	(オプション、Acrobat 6.0) <code>nPos</code> で指定した位置を基準として、新しいメニュー項目の挿入位置を指定します。デフォルト値は <code>false</code> です。 <code>bPrepend</code> が <code>true</code> の場合は、次の位置に挿入されます。 <ul style="list-style-type: none"><code>nPos</code> が文字列の場合は、指定した名前の項目の前に新しい項目が配置されます。<code>nPos</code> が数値の場合は、指定した番号の項目の前に新しい項目が配置されます。指定した名前の項目が見つからないか、<code>nPos</code> が 0 からリストの項目数までの範囲にない場合は、メニューの（末尾ではなく）先頭に新しい項目が挿入されます。 <p><code>bPrepend</code> は、グループの先頭にある項目の前に新しい項目を挿入したい場合に便利です。</p>

例 1

警告ダイアログボックスを開いてアクティブな文書のタイトルを表示するメニュー項目を、ファイルメニューの先頭に追加します。このメニューは、文書が開いている場合にのみ使用可能です。

```
app.addMenuItem({ cName: "Hello", cParent: "File",
    cExec: "app.alert(event.target.info.title, 3);",
    cEnable: "event.rc = (event.target != null);",
    nPos: 0
});
```

例 2 (Acrobat 6.0)

ファイルメニューに 2 つのメニュー項目を挿入します。1 つは「閉じる」メニュー項目の前に、もう 1 つは「閉じる」メニュー項目の後に挿入します。

```
// 「閉じる」メニュー項目の後に挿入（デフォルトの動作）
app.addMenuItem( { cName: "myItem1", cUser: "My Item 1", cParent:
    "File", cExec: "_myProc1()", nPos: "Close" });
// bPrepend を true に設定して「閉じる」メニュー項目の前に挿入
app.addMenuItem( { cName: "myItem2", cUser: "My Item 2", cParent:
    "File", cExec: "_myProc2()", nPos: "Close", bPrepend: true } );
```

addSubMenu



サブメニューを持つメニュー項目をアプリケーションに追加します。

[addMenuItem](#)、[execMenuItem](#)、[hideMenuItem](#)、[listMenuItems](#) の各メソッドも参照してください。

注意：このメソッドを実行できるのは、アプリケーション初期化イベントまたはコンソールイベントのみです。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cName	言語に依存しない、メニュー項目の名前。この言語に依存しない名前は、 hideMenuItem などでこのメニュー項目にアクセスする際に使用します。
cUser	(オプション) ユーザインターフェイスにメニュー項目の名前として表示する文字列（言語に依存する名前）。cUser を指定しない場合は、cName が使用されます。
cParent	サブメニューを追加する親メニュー項目の名前。 メニュー項目名は、 listMenuItems を使用して確認できます。
nPos	(オプション) サブメニューを追加する親のサブメニュー内の位置。デフォルトでは親のサブメニューの末尾に追加されます。nPos に 0 を指定すると親のサブメニューの先頭に追加されます。 アルファベット順に整列が行われるメニューでは、nPos パラメータは無視されます。アルファベット順に整列が行われるメニューは、次のとおりです。 <ul style="list-style-type: none">表示／ナビゲーションパネルの最初のセクション。表示／ツールバーの最初のセクション。アドバンストサブメニューの最初のセクション。 注意： nPos が数値の場合、ツールメニューでは nPos は無視されます。ツールメニューに追加したメニュー項目は、メニューの先頭に挿入されます。nPos によって挿入位置が決まるのは、nPos が別のユーザ定義のメニュー項目を参照する文字列である場合です。

例

[newDoc](#) メソッドを参照してください。

addToolBarButton

6.0			
-----	--	--	--

Acrobat のアドオンツールバーにボタンを追加します。

Acrobat でアクティブな文書（例えば docA.pdf とします）が開いているときにこのメソッドでボタンを追加した場合、docA.pdf が非アクティブになると閉じられると、追加したボタンは自動的に削除されます。非アクティブになってボタンが削除された場合は、docA.pdf が再びアクティブな文書になると、削除されたボタンが自動的にツールバーに追加されます。

ボタンのアイコンサイズは、20 × 20 ピクセルに制限されます。これより大きなサイズのアイコンが使用された場合は、例外が発生します。

注意： (Acrobat 7.0) このメソッドには、セキュリティに関連する様々な変更が加えされました。

addToolBarButton をコンソールまたはアプリケーションの初期化時に実行する場合は、セキュリティによる制限がないコンテキストとなるため、制限なく実行可能です。

セキュリティによる制限があるコンテキストでこのメソッドを呼び出すと、警告の「JavaScript ウィンドウ」がアドオンツールバーに表示されます。このツールバーはドッキングできません ([32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」を参照](#))。

[removeToolBarButton](#) も参照してください。

パラメータ

cName	言語に依存しない、ボタンの一意の識別子。言語に依存しない名前は、他のメソッド (removeToolBarButton など) でこのボタンにアクセスする際に使用します。
	注意： cName には、一意の値を使用してください。同じ名前を使用しないように、listToolbarButtons で、現在インストールされているすべてのツールバーボタン名を確認してください。
oIcon	Icon Stream オブジェクト。 Acrobat 7.0 以降では、cLabel が指定されている場合は、このパラメータはオプションになります。
cExec	ボタンが選択されたときに評価する式を表す文字列。
cEnable	(オプション) ツールボタンを使用可能にするかどうかを制御する式を表す文字列。デフォルトでは、ボタンは常に使用可能になります。ボタンを使用不可にするには、この式で event.rc を false に設定します。
cMarked	(オプション) ツールボタンにマークを付けるかどうかを制御する式を表す文字列。デフォルトでは、ボタンにマークは付きません。ボタンにマークを付けるには、この式で event.rc を true に設定します。
cTooltext	(オプション) ツールボタンの上にマウスが置かれたときにボタンのヘルプテキストとして表示するテキスト。デフォルトでは、ツールヒントは表示されません。 注意： 文字列が切り捨てられる場合があるので、cTooltext 文字列では拡張文字を使用しないでください。

nPos	(オプション) ツールバーに追加するボタンの挿入位置を表すボタン番号（この番号の前に挿入されます）。nPos が -1（デフォルト）の場合は、ツールバーの末尾にボタンが追加されます。
cLabel	(オプション、Acrobat 7.0) ボタンのアイコンの右側に表示するテキストラベル。デフォルトでは、ラベルは表示されません。

戻り値

undefined

例

ユーザのハードドライブにあるアイコングラフィックを使用して、ボタンを作成します。このスクリプトはコンソールから実行します。

```
// 文書を作成
var myDoc = app.newDoc();

// 指定のファイルからアイコン（20 × 20 ピクセル）を取り込む
myDoc.importIcon("myIcon", "/C/myIcon.jpg", 0);

// アイコンをストリームに変換
oIcon = util.iconStreamFromIcon(myDoc.getIcon("myIcon"));

// アイコンストリームを取得したので文書を閉じる
myDoc.closeDoc(true);

// ツールボタンを追加
app.addToolButton({
  cName: "myToolButton",
  oIcon: oIcon,
  cExec: "app.alert('Someone pressed me!')",
  cTooltext: "Push Me!",
  cEnable: true,
  nPos: 0
});

app.removeToolButton("myToolButton")
```

[util.iconStreamFromIcon](#) の例も参照してください。

alert

3.01			
------	--	--	--

警告ダイアログボックスを表示します。

パラメータ

cMsg	表示するメッセージを表す文字列。
nIcon	(オプション) アイコンの種類。有効な値は、次のとおりです。 0 — 警告（デフォルト） 1 — 注意 2 — 問い合わせ 3 — 情報
	注意： Macintosh では、注意と問い合わせは区別されません。
nType	(オプション) ボタンの組み合わせの種類。有効な値は、次のとおりです。 0 — OK（デフォルト） 1 — OK、キャンセル 2 — はい、いいえ 3 — はい、いいえ、キャンセル
cTitle	(オプション、Acrobat 6.0) ダイアログボックスのタイトル。指定しなかった場合は、「Adobe Acrobat」というタイトルが使用されます。
oDoc	(オプション、Acrobat 6.0) 警告に関連付ける Doc オブジェクト。
oCheckbox	(オプション、Acrobat 6.0) これを指定した場合は、警告ボックスの左下の領域にチェックボックスが作成されます。oCheckbox は、3 つのプロパティを持つ汎用 JavaScript オブジェクトです。最初の 2 つのプロパティ値は alert メソッドに渡されます。3 つ目のプロパティにはブーリアン値が渡されます。 cMsg — (オプション) チェックボックスとともに表示する文字列。指定しなかった場合は、デフォルトの文字列である「次回から表示しない」が表示されます。 bInitialValue — (オプション) true の場合、チェックボックスの初期状態としてチェックを付けます。デフォルトは false です。 bAfterValue — ダイアログボックスが閉じたときのチェックボックスの状態が、alert メソッドの終了時にこのプロパティに渡されます。true の場合は、警告ボックスが閉じたときにチェックボックスにチェックが付いていたことを表します。

戻り値

ユーザがクリックしたボタンの種類を示す nButton。

- 1 — OK
- 2 — キャンセル
- 3 — いいえ
- 4 — はい

例 1

単純な警告ボックスを表示します。

```
app.alert({  
    cMsg: "Error! Try again!",  
    cTitle: "Acme Testing Service"  
});
```

例 2

ユーザの許可があれば文書を閉じます。

```
// 「マウスボタンを放す」アクション
var nButton = app.alert({
    cMsg: "Do you want to close this document?",
    cTitle: "A message from A. C. Robat",
    nIcon: 2, nType: 2
});
if (nButton == 4) this.closeDoc();
```

例 3 (Acrobat 6.0)

ある文書で警告ボックスを作成し、別の文書の画面上に表示します。DocA と DocB の 2 つの文書があり、1 つはブラウザで開いており、もう 1 つはビューアで開いているとします。

```
// DocA の文書レベルでの宣言
var myAlertBoxes = new Object;
myAlertBoxes.oMyCheckbox = {
    cMsg: "Care to see this message again?",
    bAfterValue: false
}
```

次のコードは、DocA の「マウスボタンを放す」アクションです。変数 theOtherDoc は DocB の Doc オブジェクトです。この警告ボックスでは、同じ警告ボックスを再度表示するかどうかをユーザに確認しています。ユーザがチェックボックスにチェックを付けると、次回からこの警告は表示されなくなります。

```
if (!myAlertBoxes.oMyCheckbox.bAfterValue)
{
    app.alert({
        cMsg: "This is a message from the DocA?",
        cTitle: "A message from A. C. Robat",
        oDoc: theOtherDoc,
        oCheckbox: myAlertBoxes.oMyCheckbox
    });
}
```

beep

3.01			
------	--	--	--

システムの音を鳴らします。

注意：Macintosh および UNIX システムでは、音の種類は無視されます。

パラメータ

nType (オプション) 音の種類。値は、次のように音と関連付けられています。

- 0 — 警告
 - 1 — 注意
 - 2 — 問い合わせ
 - 3 — 情報
 - 4 — デフォルト (デフォルト値)
-

beginPriv

7.0			
-----	--	--	--

現在のスタックフレームの実行権限を引き上げて、セキュリティによる制限があるメソッドがセキュリティ例外を起こさずに実行できるようにします。このメソッドを正しく実行するためには、信頼済み関数の実行を表すフレームがスタックに存在し、そのフレームと現在実行中のフレームの間にあるすべてのフレーム（呼び出しを行っているフレームも含む）で信頼伝播関数が実行されている必要があります。

実行権限を元に戻すには、`app.endPriv` を使用します。信頼済み関数の作成には `app.trustedFunction` メソッドを使用します。信頼伝播関数の作成には `app.trustPropagatorFunction` を使用します。「スタッフレーム」という用語については、`app.trustedFunction` の説明の後にある記述を参照してください。

戻り値

成功した場合は `undefined`、失敗した場合は例外

例

使用方法の例は、[trustedFunction](#) および [trustPropagatorFunction](#) を参照してください。

browseForDoc

7.0			7.0
-----	--	--	-----

ファイルシステムブラウザを表示し、ユーザの応答に関する情報が格納されたオブジェクトを返します。

注意： このメソッドを実行できるのは、バッヂイベントまたはコンソールイベントのみです。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

パラメータ

<code>bSave</code>	(オプション) ブーリアン値。 <code>true</code> の場合は、保存操作用のファイルシステムブラウザが表示されます。デフォルトは <code>false</code> です。
<code>cFilenameInit</code>	(オプション) ファイルシステムブラウザに表示するデフォルトのファイル名を指定する文字列。
<code>cFSInit</code>	(オプション) ファイルシステムブラウザがデフォルトで対象とするファイルシステムを指定する文字列。サポートされている値は 2 つで、デフォルトのファイルシステムを表す空の文字列と、「CHTTP」です。デフォルト値は、デフォルトのファイルシステムです。Web サーバが WebDAV をサポートする場合にのみ、このパラメータは意味を持ちます。

戻り値

成功した場合は、次の 3 つのプロパティを持つオブジェクトが返されます。

プロパティ	説明
cFS	選択したファイルのファイルシステム名を表す文字列。
cPath	選択したファイルのパスを表す文字列。
cURL	選択したファイルの URL を表す文字列。

ユーザがキャンセルした場合、戻り値は `undefined` です。エラーの場合は例外が発生します。

例 1

文書を参照して、結果をコンソールに表示します。

```
var oRetn = app.browseForDoc({
    cFilenameInit: "myComDoc.pdf",
    cFSInit: "CHTTP",
});
if ( typeof oRetn != "undefined" )
    for ( var o in oRetn )
        console.println( "oRetn." + o + "=" + oRetn[o] );
else console.println("User cancelled!");
```

上記コードで、ユーザが WebDAV サーバ上のファイルを選択した場合の出力を次に示します。

```
oRetn.cFS=CHTTP
oRetn.cPath=http://www.example.com/WebDAV/myComDoc.pdf
oRetn.cURL=http://www.example.com/WebDAV/myComDoc.pdf
```

上記コードで、ユーザがデフォルトのファイルシステムでファイルを選択した場合の出力を次に示します。

```
oRetn.cFS=DOS
oRetn.cPath=/C/temp/myComDoc.pdf
oRetn.cURL=file:///C|/temp/myComDoc.pdf
```

このメソッドに続けて `app.openDoc` を使用すると、選択したファイルを開くことができます。

```
var myURL = (oRetn.cFS=="CHTTP") ? encodeURI(oRetn.cPath) : oRetn.cPath;
var oDoc = app.openDoc({cPath: myURL, cFS: oRetn.cFS});
```

注意：WebDAV サーバからファイルを取得する場合は、`cPath` をエスケープして `app.openDoc` に渡す必要があります。詳しい説明と例については、[app.openDoc](#) を参照してください。

例 2

文書の保存先/パスを指定して保存します。

```
var oRetn = app.browseForDoc({
    bSave: true,
    cFilenameInit: "myComDoc.pdf",
    cFSInit: "CHTTP",
});
if ( typeof oRetn != "undefined" ) this.saveAs({
    cFS: oRetn.cFS, cPath: oRetn.cPath, bPromptToOverwrite: false});
```

clearInterval

5.0			
-----	--	--	--

`setInterval` メソッドで設定された、登録済みのインターバルをキャンセルします。

[setTimeout](#) および [clearTimeOut](#) も参照してください。

パラメータ

oInterval	登録済みのインターバル。これをキャンセルします。
-----------	--------------------------

例

[setTimeout](#) を参照してください。

clearTimeOut

5.0			
-----	--	--	--

登録済みのタイムアウトインターバルをキャンセルします。これは、`setTimeout` で設定されたインターバルです。

[setInterval](#) および [clearInterval](#) も参照してください。

パラメータ

oTime	登録済みのタイムアウトインターバル。これをキャンセルします。
-------	--------------------------------

例

[setTimeout](#) を参照してください。

endPriv

7.0		⌚	
-----	--	---	--

`app.beginPriv` によって現在のスタックフレームに与えられた実行権限を元に戻します。セキュリティによる制限がないイベントで与えられた実行権限は、このコマンドの影響を受けません。

関連するメソッドとしては、`app.trustedFunction`、`app.trustPropagatorFunction`、`app.beginPriv` があります。

戻り値

成功した場合は `undefined`、失敗した場合は例外

例

使用方法の例は、[trustedFunction](#) および [trustPropagatorFunction](#) を参照してください。

execDialog

7.0			
-----	--	--	--

モーダルダイアログボックスを表示します。モーダルダイアログボックスが表示されると、そのダイアログボックスを閉じるまで、ユーザはホストアプリケーションを直接操作できなくなります。

`monitor` パラメータにはダイアログボックス記述子を指定します。これは汎用のオブジェクトリテラルで、様々なイベント用の一連のハンドラ関数と、ダイアログボックスの内容を記述する一連のプロパティで構成されます。

ダイアログボックスアイテムは、一意の 4 文字の文字列である `ItemID` によって識別されます。`ItemID` が必要になるのは、ダイアログボックスの記述内でその要素を参照する必要がある場合のみです（例えば、その要素の値を設定または取得したり、要素のハンドラを追加したり、要素を含むタブ順序を設定したりする場合など）。

注意： Acrobat のダイアログボックスと JavaScript によって作成されたダイアログボックスを区別できるように、文書レベルで追加されたダイアログボックスには「JavaScript ウィンドウ」というタイトルが付けられ、一番下に「警告：JavaScript ウィンドウ」というテキストが表示されます。

パラメータ

<code>monitor</code>	オブジェクトリテラル。一連のハンドラ (108 ページの「ダイアログボックスハンドラ」 を参照) と、ダイアログボックス要素を記述する <code>description</code> プロパティ (108 ページの「description プロパティ」 を参照) で構成されます。
<code>inheritDialog</code>	(オプション) このダイアログボックスを表示するときに再使用する <code>Dialog</code> オブジェクト。これを指定すると、新しいダイアログボックスが表示される前にダイアログボックスが消去されてしまうのを防ぐことができるので、一連のダイアログボックス (ウィザードなど) を表示する場合に便利です。デフォルトでは、ダイアログボックスは再使用されません。
<code>parentDoc</code>	(オプション) このダイアログボックスの親として使用する <code>Doc</code> オブジェクト。デフォルトの親は Acrobat アプリケーションです。

戻り値

ダイアログボックスを閉じるために使用された要素の `ItemID` を表す文字列。「OK」ボタンまたは「キャンセル」ボタンによってダイアログボックスが閉じられた場合、戻り値は「ok」または「cancel」になります。

注意： `app.execDialog` によって作成されたモーダルダイアログボックスがアクティブである間は、デバッグは無効です。

ダイアログボックスハンドラ

ダイアログボックスハンドラは、特定のダイアログボックスイベントが発生したときに呼び出されます。どのハンドラもオプションです。ハンドラには Dialog オブジェクトが渡されるので、これを使用してダイアログボックス内の値を取得したり設定したりできます。サポートされているハンドラを、次の表に示します。

ダイアログボックス ハンドラ	説明
initialize	ダイアログボックスが初期化されるときに呼び出されます。
validate	フィールドが変更されたときに、その値が許容できるか (<code>true</code> が返されます)、許容できないか (<code>false</code> が返されます) を判断するために呼び出されます。
commit	ダイアログボックスの「OK」がクリックされたときに呼び出されます。
destroy	ダイアログボックスが破棄されるときに呼び出されます。
ItemID	<p>ItemID というダイアログボックス要素が変更されたときに呼び出されます。テキストボックスの場合は、テキストボックスのフォーカスが解除されたときに呼び出されます。その他のコントロールについては、選択が変更されたときに呼び出されます。</p> <p>ItemID が JavaScript の識別子でない場合は、次のように、メソッドを定義するときに名前を二重引用符で囲む必要があります。</p> <pre>"bt:1": function () { }</pre> <p>ItemID が JavaScript の識別子である場合、二重引用符はオプションです。例えば、次の 2 つはどちらも正しいコードです。</p> <pre>"butn": function () { } butn: function () { }</pre>

description プロパティ

`description` プロパティは、ダイアログボックスを記述するプロパティが含まれたオブジェクトリテラルです。ダイアログボックスの要素は、`description` プロパティの `elements` プロパティで指定します。その要素でさらに `elements` プロパティを指定することで、サブ要素を持つことができます。

`description` プロパティのルートレベルで設定されるダイアログボックスプロパティを、次の表に示します。

プロパティ	型	説明
<code>name</code>	文字列	ダイアログボックスのタイトルバー。識別子ではないため、ユーザが読みやすい表現にしてください。
<code>first_tab</code>	文字列	タブの順序で最初のアイテムとなるダイアログボックスアイテムの ItemID。このダイアログボックスアイテムは、ダイアログボックスが作成されたときにアクティブになります。このプロパティは、タブの順序を設定するために必要です。後述の next_tab プロパティを参照してください。
<code>width</code>	数値	ピクセル単位で表したダイアログボックスの幅。幅が指定されていない場合、コンテンツの幅の合計が使用されます。
<code>height</code>	数値	ピクセル単位で表したダイアログボックスの高さ。高さが指定されていない場合、コンテンツの高さの合計が使用されます。
<code>char_width</code>	数値	文字数で表したダイアログボックスの幅。幅が指定されていない場合、コンテンツの幅の合計が使用されます。
<code>char_height</code>	数値	文字数で表したダイアログボックスの高さ。高さが指定されていない場合、コンテンツの高さの合計が使用されます。

プロパティ	型	説明
align_children	文字列	すべての子要素の整列方法。次のいずれかの値である必要があります。 「align_left」：左揃え 「align_center」：中央揃え 「align_right」：右揃え 「align_top」：上揃え 「align_fill」：親の幅を埋めるように整列します。オブジェクトの幅が拡大する場合があります。 「align_distribute」：親の幅いっぱいにコンテンツを配分します。 「align_row」：一定のベースラインに合わせて親の幅いっぱいにコンテンツを配分します。 「align_offscreen」：アイテムを互いに重なりあった状態で整列します。
elements	配列	このダイアログボックスに配置するダイアログボックス要素を表すオブジェクトリテラルの配列 (elements プロパティ を参照)。

elements プロパティ

ダイアログボックスの `elements` プロパティでは、次の一連のプロパティを持つオブジェクトリテラルを指定します。

プロパティ	型	説明
name	文字列	ダイアログボックス要素の表示名。識別子ではないため、ユーザが読みやすい表現にしてください。 注意： 「edit_text」タイプでは、このプロパティは無視されます。
item_id	文字列	ダイアログボックスの ItemID。一意の 4 文字の文字列です。
type	文字列	このダイアログボックス要素のタイプ。次のいずれかの文字列である必要があります。 「button」 - ボタン。 「check_box」 - チェックボックス。 「radio」 - ラジオボタン。 「list_box」 - リストボックス。 「hier_list_box」 - 階層型リストボックス。 「static_text」 - 静的テキスト。 「edit_text」 - 編集可能なテキストボックス。 「popup」 - ポップアップコントロール。 「ok」 - 「OK」 ボタン。 「ok_cancel」 - 「OK」 および 「キャンセル」 ボタン。 「ok_cancel_other」 - 「OK」、「キャンセル」、「その他」 ボタン。 「view」 - 一連のコントロールのコンテナ。 「cluster」 - 一連のコントロールのフレーム。 「gap」 - プレイスホルダ。

プロパティ	型	説明
next_tab	文字列	タブの順序で次のアイテムとなるダイアログボックスアイテムの ItemID。 注意： next_tab (または first_tab) プロパティで指定されていないダイアログボックスアイテムに、Tab キーでフォーカスが移動することはありません。タブの順序は、1 つのループを形成している必要があります。
width	数値	要素の幅をピクセルで指定します。幅が指定されていない場合、コンテンツの幅の合計が使用されます。
height	数値	要素の高さをピクセルで指定します。高さが指定されていない場合、コンテンツの高さの合計が使用されます。
char_width	数値	要素の幅を文字数で指定します。幅が指定されていない場合、コンテンツの幅の合計が使用されます。
char_height	数値	要素の高さを文字数で指定します。高さが指定されていない場合、コンテンツの高さの合計が使用されます。
font	文字列	この要素で使用するフォント。次のいずれかの文字列である必要があります。 「default」 - デフォルトフォント 「dialog」 - ダイアログボックスフォント 「palette」 - パレット（小）フォント
bold	ブーリアン	フォントが太字かどうかを指定します。
italic	ブーリアン	フォントが斜体かどうかを指定します。
alignment	文字列	この要素の整列方法を設定します。次のいずれかの値である必要があります。 「align_left」：左揃え 「align_center」：中央揃え 「align_right」：右揃え 「align_top」：上揃え 「align_fill」：親の幅を埋めるように整列します。オブジェクトの幅が拡大する場合があります。 「align_distribute」：親の幅いっぱいにコンテンツを配分します。 「align_row」：一定のベースラインに合わせて親の幅いっぱいにコンテンツを配分します。 「align_offscreen」：アイテムを互いに重なりあった状態で整列します。
align_children	文字列	すべての子要素の整列を設定します。有効な値は、alignment と同じです。
elements	配列	このダイアログボックス要素のサブ要素を表すオブジェクトリテラルの配列。プロパティは、この表で既に説明したものと同じです。

一部のダイアログボックス要素の追加属性

一部の要素タイプには、次に示す追加の属性があります。

element のタイプ	プロパティ	型	説明
static_text	multiline	ブーリアン	true の場合、この静的テキスト要素は複数行にわたります。 注意： Macintosh の場合、ボックスの内容を上下にスクロールするボタンが表示されるためには、height プロパティが 49 以上であることが必要です。
edit_text	multiline	ブーリアン	true の場合、この編集テキスト要素は複数行にわたります。
	readonly	ブーリアン	true の場合、このテキスト要素は読み取り専用です。 注意： このプロパティは、password が true に設定されている場合は無視されます。
	password	ブーリアン	true の場合、このテキスト要素はパスワードフィールドです。
	PopupEdit	ブーリアン	true の場合、これはポップアップ編集テキスト要素です。
	SpinEdit	ブーリアン	true の場合、これはスピン編集テキスト要素です。
radio	group_id	文字列	このラジオボタンが属するグループの名前。
ok、ok_cancel、ok_cancel_other	ok_name	文字列	「OK」ボタンの名前。
	cancel_name	文字列	「キャンセル」ボタンの名前。
	other_name	文字列	「その他」ボタンの名前。

例 1

次のダイアログボックス記述子は、文書レベルまたはフォルダレベルの JavaScript として使用できます。作成されるダイアログボックスには、姓名を入力するためのテキストフィールドが含まれています。「OK」をクリックすると、入力された名前がコンソールに表示されます。

```
var dialog1 = {

    initialize: function (dialog) {
        // 現在の日付を保持する静的テキストを作成
        var todayDate = dialog.store()["date"];
        todayDate = "Date: " + util.printd("mmmm dd, yyyy", new Date());
        dialog.load({ "date": todayDate });
    },
    commit:function (dialog) { // 「OK」が押されたときに呼び出される
        var results = dialog.store();
        // 収集したデータに対して何か処理を行う。次はその例
        console.println("Your name is " + results["fnam"]
            + " " + results["lnam"]);
    },
}
```

```
description:  
{  
    name: "Personal Data", // ダイアログボックスのタイトル  
    align_children: "align_left",  
    width: 350,  
    height: 200,  
    elements:  
    [  
        {  
            type: "cluster",  
            name: "Your Name",  
            align_children: "align_left",  
            elements:  
            [  
                {  
                    type: "view",  
                    align_children: "align_row",  
                    elements:  
                    [  
                        {  
                            type: "static_text",  
                            name: "First Name: "  
                        },  
                        {  
                            item_id: "fnam",  
                            type: "edit_text",  
                            alignment: "align_fill",  
                            width: 300,  
                            height: 20  
                        }  
                    ]  
                },  
                {  
                    type: "view",  
                    align_children: "align_row",  
                    elements:  
                    [  
                        {  
                            type: "static_text",  
                            name: "Last Name: "  
                        },  
                        {  
                            item_id: "lnam",  
                            type: "edit_text",  
                            alignment: "align_fill",  
                            width: 300,  
                            height: 20  
                        }  
                    ]  
                },  
                {  
                    type: "static_text",  
                    name: "Date: ",  
                }  
            ]  
        }  
    ]  
}
```

```
        char_width: 25,
        item_id: "date"
    },
]
},
{
    alignment: "align_right",
    type: "ok_cancel",
    ok_name: "Ok",
    cancel_name: "Cancel"
}
]
}
};
```

次の行は、ボタンの「マウスボタンを放す」アクションや、メニューアクションなどから実行できます。

```
app.execDialog(dialog1);
```

例 2

次の例では、チェックボックスとラジオボタンフィールドを使用します。このコードは文書レベルの JavaScript として使用できます。

```
var dialog2 =
{
    initialize: function(dialog) {
        // ラジオボタンフィールドのデフォルト値を設定
        dialog.load({ "rd01": true });
        this.hasPet = false;
        // ラジオボタンフィールドを無効にする
        dialog.enable({
            "rd01" : this.hasPet,
            "rd02" : this.hasPet,
            "rd03" : this.hasPet
        });
    },
    commit: function(dialog) {
        // ユーザが「OK」を押すと、このハンドラが最初に起動する
        console.println("commit");
        var results = dialog.store();
        // データに対して何か処理を行う。次はその例
        var hasPet = (this.hasPet) ? "have" : "don't have";
        console.println("You " + hasPet + " a pet.");
        if (this.hasPet)
            console.println("You have " + this.getNumPets(results)
                + " pet(s).");
    },
    getNumPets: function (results) {
        for ( var i=1; i<=3; i++ ) {
            if ( results["rd0"+i] ) {
                switch (i) {
                    case 1:
                        var nPets = "one";
                        break;
                    case 2:
                        nPets = "two";
                        break;
                    case 3:
                        nPets = "three";
                        break;
                }
                return nPets;
            }
        }
    }
};
```

```
        case 2:
            var nPets = "two";
            break;
        case 3:
            var nPets = "three or more";
    }
}
},
return nPets;
},
ok: function(dialog) {
    // commit の後、「OK」ボタンのハンドラが最初に渡される
    console.println("Ok!");
},
ckbx: function (dialog) {
    // チェックボックスを処理し、ユーザがペットを持っている場合は、ラジオボタンをオンにする
    this.hasPet = !this.hasPet;
    dialog.enable({
        "rd01" : this.hasPet,
        "rd02" : this.hasPet,
        "rd03" : this.hasPet
    });
},
cancel: function(dialog) { // 「キャンセル」ボタンを処理
    console.println("Cancel!");
},
other: function(dialog){ // 「その他」ボタンを処理
    app.alert("Thanks for pressing me!");
    dialog.end("other"); // ダイアログボックスを閉じ、「other」を返す
},
// ダイアログボックスの記述
description:
{
    name: "More Personal Information",
    elements:
    [
        {
            type: "view",
            align_children: "align_left",
            elements:
            [
                {
                    type: "static_text",
                    name: "Personal Information",
                    bold: true,
                    font: "dialog",
                    char_width: 30,
                    height: 20
                },
                {
                    type: "check_box",
                    item_id: "ckbx",
                    name: "Pet Owner"
                }
            ]
        }
    ]
}
```

```

        type: "view",
        align_children: "align_row",
        elements:
        [
            {
                type: "static_text",
                name: "Number of pets: "
            },
            {
                type: "radio",
                item_id: "rd01",
                group_id: "rado",
                name: "One"
            },
            {
                type: "radio",
                item_id: "rd02",
                group_id: "rado",
                name: "Two",
            },
            {
                type: "radio",
                item_id: "rd03",
                group_id: "rado",
                name: "Three or more",
            }
        ]
    }
},
{
    type: "gap", // ラジオフィールドとボタンの間に
    height: 10 // わずかな垂直の間隔をあける
},
{
    type: "ok_cancel_other",
    ok_name: "Ok",
    cancel_name: "Cancel",
    other_name: "Press Me"
}
]
}
};
```

次の行は、ボタンの「マウスボタンを放す」アクションや、メニューアクションなどから実行できます。

```
var retn = app.execDialog(dialog2);
```

`retn` の値は、ユーザが「OK」をクリックした場合は「ok」、「Cancel」をクリックした場合は「cancel」、「Press Me」ボタンをクリックした場合は「other」になります。

例 3

この例では、リストボックスを使用します。

```
var dialog3 = {
    // このハンドラは、ダイアログボックスが作成されたときに呼び出される
    initialize: function(dialog) {
        this.loadDefaults(dialog);
    },
    // このハンドラは、「OK」がクリックされたときに呼び出される
    commit: function(dialog) {
        // dialog.load と dialog.store の動作について詳しくは、
        // Dialog オブジェクトを参照
        var elements = dialog.store()["subl"];
        // データに対して何か処理を行う
    },
    // 「butn」ボタンがクリックされたときのハンドラ
    butn: function(dialog) {
        var elements = dialog.store()["subl"]
        for(var i in elements) {
            if (elements[i] > 0) {
                app.alert("You chose ¥" + i
                    + "¥", which has a value of " + elements[i] );
            }
        }
    },
    loadDefaults: function (dialog) {
        dialog.load({
            subl:
            {
                "Acrobat Professional": +1,
                "Acrobat Standard": -2,
                "Adobe Reader": -3
            }
        })
    },
    // ダイアログボックスの記述
    description:
    {
        name: "Adobe Acrobat Products", // ダイアログボックスのタイトル
        elements: // 子要素の配列
        [
            {
                type: "view",
                align_children: "align_left",
                elements: // 子要素の配列
                [
                    {
                        type: "cluster",
                        name: "Select",
                        elements: // 子要素の配列
                        [
                            {
                                type: "static_text",
                                name: "Select Acrobat you use",
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```

```
        font: "default"
    },
{
    type: "list_box",
    item_id: "subl",
    width: 200,
    height: 60
},
{
    type: "button",
    item_id: "butn",
    name: "Press Me"
}
],
{
    type: "ok_cancel"
}
]
}
]
```

ここで、次のコードを実行します。

```
app.execDialog(dialog3);
```

この例で、type: "list_box" を type: "popup" に置き換えて、height の指定を削除すると、リストボックスではなくポップアップになります。

例 4

この例では、階層型リストボックスを使用します。ダイアログボックスが開いたあと、階層リストが表示されます。ユーザが選択を行った後、「Select」ボタンをクリックすると、ユーザが選択した場所にジャンプします。Doc オブジェクトは、ダイアログボックスのプロパティとしてダイアログボックスに渡しています。

```
var dialog4 = {
    initialize: function(dialog) {
        dialog.load({
            subl:
            {
                "Chapter 1":
                {
                    "Section 1":
                    {
                        "SubSection 1": -1,
                        "SubSection 2": -2,
                    },
                    "Section 2":
                    {
                        "SubSection 1": -3,
                        "SubSection 2": -4,
                    }
                },
                "Chapter 3": -5,
            }
        });
    }
};
```

```
        "Chapter 4": -6
    }
})
},
subl: function(dialog) {
    console.println("Selection Box Hit");
},
getHierChoice: function (e)
{
    if (typeof e == "object") {
        for ( var i in e ) {
            if ( typeof e[i] == "object" ) {
                var retn = this.getHierChoice(e[i]);
                if ( retn ) {
                    retn.label = i + ", " + retn.label;
                    return retn;
                }
                // e[i] > 0 の場合、選択したアイテムが見つかった
            } else if ( e[i] > 0 ) return { label:i, value: e[i] };
        }
    } else {
        if ( e[i] > 0 ) return e[i];
    }
},
butn: function (dialog)
{
    var element = dialog.store()["subl"]
    var retn = this.getHierChoice(element);
    if ( retn ) {
        // 選択されたアイテムの完全な名前をコンソールに表示する
        console.println("The selection you've chosen is ¥""
            + retn.label + "¥", its value is " + retn.value );
        dialog.end("ok");
        //this.doc は、この文書の doc オブジェクト
        this.doc.gotoNamedDest ("dest"+retn.value);
    }
    else app.alert("Please make a selection, or cancel");
},
cncl: function (dialog) { dialog.end("cancel") },
// ダイアログボックスの記述
description:
{
    name: "My Novel",
    elements:
    [
        {
            type: "view",
            align_children: "align_left",
            elements:
            [
                {
                    type: "cluster",
                    name: "Book Headings",

```

```
elements:
[
  [
    {
      type: "static_text",
      name: "Make a selection",
    },
    [
      {
        type: "hier_list_box",
        item_id: "subl",
        char_width: 20,
        height: 200
      }
    ]
  ],
  [
    {
      type: "view",
      align_children: "align_row",
      elements:
      [
        [
          {
            type: "button",
            item_id: "cncl",
            name: "Cancel"
          },
          {
            item_id: "butn",
            type: "button",
            name: "Select"
          }
        ]
      ]
    }
  ]
];
}
```

次の関数では、Doc オブジェクトをダイアログボックスに関連付け、そのダイアログボックスオブジェクトを `app.execDialog` メソッドに渡しています。dialog4 オブジェクトとこの関数は、文書レベルで使用できます。

```
function dotheDialog(dialog,doc)
{
  dialog.doc = doc;
  var retn = app.execDialog( dialog )
}
```

最後に、「マウスボタンを放す」アクションなどで次のスクリプトを実行します。

```
dotheDialog( dialog4, this );
```

例 5

[145 ページの「例 2」](#) を参照してください。この例では、セキュリティによる制限があるコンテキストで、その制限を解除してコードを実行する方法が示されています。

execMenuItem

4.0			
-----	--	--	--

指定のメニュー項目を実行します。

Acrobat 5.0 以降では、以下に示されている制限に従って、`app.execMenuItem("SaveAs")` を呼び出すことができます。「名前を付けて保存」（「別名で保存」）メニュー項目を実行すると、フォルダとファイル名を選択するダイアログボックスが表示され、現在のファイルをユーザのハードドライブに保存できるようになります。「文書」環境設定で「Web 表示用に最適化して保存」にチェックが付いている場合は、現在のファイルがリニアライズされて保存されます。

注意： (Acrobat 7.0) 次のコードは、以前のバージョンの Acrobat では、バッチイベント、コンソールイベント、メニューイベントでのみ実行可能でした。

```
app.execMenuItem("SaveAs");
```

Acrobat 7.0 ではこの制限がなくなり、「マウスボタンを放す」イベントなどで `app.execMenuItem("SaveAs")` を実行できるようになりました。

ユーザ環境設定で「Web 表示用に最適化して保存」を設定している場合、「名前を付けて保存」（「別名で保存」）操作を行うと、フォームオブジェクトは破棄され、作成した Field オブジェクトは無効になります。保存操作の直後にそれらのオブジェクトにアクセスしようとすると、例外が発生します。この後の例を参照してください。

セキュリティ上の理由により、「終了」メニュー項目をスクリプトで実行することはできません。Acrobat 6.0 以降では、「貼り付け」（「ペースト」）メニュー項目をスクリプトで実行することはできません。

(Acrobat 8.0) `execMenuItem` メソッドで実行できるメニュー項目は、セーフメニューのリストに登録されている項目に制限されています。セーフメニューのリストに登録されていないメニュー項目を `execMenuItem` メソッドで実行した場合、メソッドは失敗しますが例外は発生しません。セーフメニューの内容は、今後のリリースで削除されたり、動作が変更されたりする場合があります。

セーフメニューのリストを表示するには、フォームボタンを作成し、ボタンのプロパティダイアログボックスで「アクション」タブを選択します。アクションを選択リストから「メニュー項目を実行」を選択します。「追加」ボタンをクリックすると、メニュー項目ダイアログボックスが開いてセーフメニューのリストが表示されます。

`app.execMenuItem` メソッドは、セキュリティによる制限がないコンテキスト（コンソールやバッチシーケンスなど）では制限なく実行できます。フォルダレベルの JavaScript でも、信頼済み関数を使用して権限を引き上げれば、制限なく `app.execMenuItem` を実行できます。次の例 4 を参照してください。

`app.execMenuItem` を制限なく実行する別の方法として、署名と証明を行う方法があります。文書の作成者が文書に署名して証明を行えば、セキュリティによる制限があるコンテキストで、その制限を解除してメソッドを実行できるようになります。ただし、文書の利用者が、信頼済み証明書の一覧に作成者の証明書を登録し、文書に埋め込まれた JavaScript プログラムの実行を許可するように作成者を信頼する必要があります。

[addMenuItem](#)、[addSubMenu](#)、[hideMenuItem](#) も参照してください。[listMenuItems](#) を使用すると、すべてのメニュー項目の名前がコンソールに表示されます。

パラメータ

cMenuItem	実行するメニュー項目。 メニュー項目名のリストは、listMenuItems を使用して取得できます。
oDoc	(オプション、Acrobat 7.0) oDoc は、非表示でない文書の Doc オブジェクトです (Doc オブジェクトの hidden プロパティを参照)。このパラメータを指定して execMenuItem を実行すると、その文書のコンテキストでメニュー項目が実行されます。

例 1

この例では、ファイル／開くメニュー項目を実行します。開くファイルを指定するように求めるダイアログボックスが表示されます。

```
app.execMenuItem("Open");
```

例 2 (Acrobat 5.0)

この例では、app.execMenuItem("SaveAs") の実行後にフォームオブジェクトが破棄されることを示します (前の説明を参照)。

```
var f = this.getField("myField");
// リニアライズして保存するように環境設定が指定されていると仮定
app.execMenuItem("SaveAs");
// 例外が発生し、フィールドは更新されない
f.value = 3;
```

例 3 (Acrobat 5.0)

app.execMenuItem("SaveAs") の実行後は、Field オブジェクトを再度取得する必要があります。

```
var f = this.getField("myField");
// リニアライズして保存するように環境設定が指定されていると仮定
app.execMenuItem("SaveAs");
// リニアライズして保存した後、フィールドを再度取得
var f = getField("myField");
// フィールド値は 3 に更新される
f.value = 3;
```

例 4 (Acrobat 8.0)

信頼済み関数を使用して、フォルダレベルの JavaScript で app.execMenuItem を実行します。

```
myTrustedMenu = app.trustedFunction( function( name )
{
    app.beginPriv();
    app.execMenuItem(name);
    app.endPriv();
});
```

Acrobat または Adobe Reader を再起動すると、次のようなスクリプトを

```
myTrustedMenu("PropertyToolbar");
```

セキュリティによる制限があるコンテキスト (「マウスポタンを放す」アクションなど) で実行できるようになります。このスクリプトは、プロパティツールバーの表示／非表示を切り替えます。

getNthPlugInName

X			
---	--	--	--

注意：このメソッドの代わりに plugIns プロパティを使用してください。

ビューアにロードされている n 番目のプラグイン名を取得します。

パラメータ

nIndex	ビューアにロードされている n 番目のプラグイン。
--------	---------------------------

戻り値

nIndex 番目のプラグイン名である cName。

getPath

6.0		S	
-----	--	---	--

インストール時に作成されたフォルダのパスを返します。アプリケーションフォルダとユーザフォルダは区別されます。パスが存在しない場合は、GeneralError という例外 ([Error オブジェクトを参照](#)) が発生します。

注意：(Acrobat 7.0) このメソッドを実行できるのは、バッヂイベントまたはコンソールイベントのみです ([event オブジェクトを参照](#))。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

パラメータ

cCategory	(オプション) フォルダ検索のカテゴリー。有効な値は、app (デフォルト) と user です。
cFolder	(オプション) プラットフォームに依存しない、フォルダを示す文字列。有効な値は、次のとおりです。 root、eBooks、preferences、sequences、documents、 javascript、stamps、dictionaries、plugIns spPlugIns、help、temp、messages、resource、update デフォルトは root です。

戻り値

パラメータで指定したフォルダのパス。フォルダが存在しない場合は、例外が発生します。

例 1

ユーザの Sequences フォルダのパスを検索します。

```
try {
    var userBatch = app.getPath("user", "sequences");
} catch(e) {
    var userBatch = "User has not defined any custom batch sequences";
}
console.println(userBatch);
```

例 2

Windows プラットフォームで文書を作成して、My Documents フォルダに保存します。

```
var myDoc = app.newDoc();
var myPath = app.getPath("user", "documents") + "/myDoc.pdf"
myDoc.saveAs(myPath);
myDoc.closeDoc();
```

goBack

3.01			
------	--	--	--

ビュースタックにある前のビューに戻ります。これは、Acrobat ツールバーの「前の画面」ボタンをクリックするのと同じです。

例

戻るボタンを作成します。このコードをバッチシーケンスに組み込めば、選択した PDF 文書にナビゲーションボタンを追加するなどの処理が行えます。

```
var aRect = this.getPageBox();
var width = aRect[2] - aRect[0];
// 高さが 12 ポイントで幅が 16 ポイントの長方形を最下部の中央に配置
rect = [width/2-8, 10, width/2+8, 22];
f = this.addField("goBack", "button", 0, rect);
f.textFont="Wingdings";
f.textSize=0;
f.buttonSetCaption("\u2190"); // 左矢印
f.setAction("MouseUp", "app.goBack()"); // アクションを追加
```

goForward

3.01			
------	--	--	--

ビュースタックにある次のビューに進みます。これは、Acrobat ツールバーの「次の画面」ボタンをクリックするのと同じです。

例

app.[goBack](#) の例を参照してください。

hideMenuItem

4.0			
-----	--	--	--

指定のメニュー項目を削除します。

[addMenuItem](#)、[addSubMenu](#)、[execMenuItem](#)、[listMenuItems](#) も参照してください。

注意：このメソッドを実行できるのは、コンソールイベントまたはアプリケーション初期化イベントのみです。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cName	削除するメニュー項目名。 メニュー項目名は、 listMenuItems を使用して確認できます。
-------	--

hideToolbarButton

4.0			
-----	--	--	--

指定のツールバーボタンを削除します。

注意：このメソッドを実行できるのは、コンソールイベントまたはアプリケーション初期化イベントのみです。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cName	削除するツールバーの名前。 ツールバー項目名は、 listToolbarButtons を使用して確認できます。
-------	---

例

次のスクリプトが含まれている `myConfig.js` というファイルを、フォルダレベルの JavaScript フォルダに配置します。

```
app.hideToolbarButton("Hand");
```

ビューアを起動すると、手のひらアイコンが表示されなくなります。

launchURL

7.0			
-----	--	--	--

ブラウザウィンドウで URL を開きます。

パラメータ

cURL	開く URL を指定する文字列。
bNewFrame	(オプション) <code>true</code> の場合は、ブラウザアプリケーションの新しいウィンドウで URL を開きます。デフォルトは <code>false</code> です。

戻り値

成功した場合は、`undefined` という値が返されます。失敗した場合は例外が発生します。

例 1

```
app.launchURL("http://www.example.com/", true);
```

例 2

メニューシステムにヘルプ項目を追加します。このコードは、フォルダレベルの JavaScript ファイルとして実行するか、JavaScript デバッガコンソールから実行する必要があります。

```
app.addMenuItem({  
    cName: "myHelp", cUser: "Online myHelp",  
    cParent: "Help",  
    cExec: "app.launchURL('www.example.com/myhelp.html');",  
    nPos: 0  
});
```

関連するメソッドとしては、[openDoc](#) や Doc オブジェクトの [getURL](#) メソッドがあります。

listMenuItems

5.0			
-----	--	--	--

Acrobat 6.0 以降では、メニュー階層を表す `TreeItem` オブジェクトの配列を返します。

Acrobat 6.0 よりも前のバージョンでは、メニュー項目名のリストをコンソールに表示していました。

[addMenuItem](#)、[addSubMenu](#)、[execMenuItem](#)、[hideMenuItem](#) も参照してください。

戻り値

`TreeItem` オブジェクトの配列。

TreeItem オブジェクト

メニュー項目またはツールバー項目の階層を表す汎用 JavaScript オブジェクト。これらのオブジェクトの配列は、`app.listMenuItems` や `app.listToolbarButtons` で返されます (Acrobat 6.0 以降)。このオブジェクトのプロパティは、次のとおりです。

cName	(オプション) メニュー項目またはツールバーボタンの名前。
oChildren	(オプション) サブメニューまたはフライアウトボタンを含む <code>TreeItem</code> オブジェクトの配列。

例 1

すべてのメニュー項目の名前をコンソールに表示します。

```
var menuItems = app.listMenuItems()  
for( var i in menuItems)  
    console.println(menuItems[i] + "\n")
```

例 2

すべてのメニュー項目をコンソールにわかりやすく表示します。

```
function FancyMenuItemList(m, nLevel)  
{  
    var s = "";  
    for (var i = 0; i < nLevel; i++) s += " ";  
    console.println(s + "+" + m.cName);  
    if (m.oChildren != null)  
        for (var i = 0; i < m.oChildren.length; i++)  
            FancyMenuItemList(m.oChildren[i], nLevel + 1);  
}  
var m = app.listMenuItems();  
for (var i=0; i < m.length; i++) FancyMenuItemList(m[i], 0);
```

listToolbarButtons

5.0			
-----	--	--	--

Acrobat 6.0 以降では、ツールバー階層（フライアウトツールバーを含む）を表す `TreeItem` オブジェクトの配列を返します。

Acrobat 6.0 よりも前のバージョンでは、ツールバーボタン名のリストをコンソールに表示していました。

(Acrobat 8.0) 環境設定ダイアログボックスの「文書」分類で、「各文書を独立したウィンドウに表示（再起動が必要）」という項目にチェックが付いている場合、`listToolbarButtons` で `TreeItem` オブジェクトの配列が返されるためには、文書ウィンドウが空であることが必要です。

戻り値

`TreeItem` オブジェクトの配列。

例

すべてのツールバーの名前をコンソールに表示します。

```
var toolbarItems = app.listToolbarButtons()  
for( var i in toolbarItems)  
    console.println(toolbarItems[i] + "\n")
```

[hideToolbarButton](#) メソッドも参照してください。

mailGetAddrs

6.0			
-----	--	--	--

注意：このメソッドは、Windows でのみ使用できます。

電子メールの受信者を選択できるアドレス帳のダイアログボックスが表示されます。オプションとして、`cTo`、`cCc`、`cBcc` にセミコロンで区切ったアドレス文字列を指定して、ダイアログボックスに表示することができます。`bCc` ブーリアンと `bBcc` ブーリアンでは、CC 受信者と BCC 受信者を選択できるかどうかを指定します。

[mailMsg](#)、Doc オブジェクトの [mailDoc](#) および [mailForm](#) メソッド、FDF オブジェクトの [mail](#) メソッド、Report オブジェクトの [mail](#) メソッドも参照してください。

注意：(Acrobat 7.0) このメソッドを実行できるのは、バッチイベントまたはコンソールイベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

<code>cTo</code>	(オプション) セミコロンで区切った To アドレスのリスト。
<code>cCc</code>	(オプション) セミコロンで区切った CC アドレスのリスト。
<code>cBcc</code>	(オプション) セミコロンで区切った BCC アドレスのリスト。
<code>cCaption</code>	(オプション) アドレスダイアログボックスのキャプションバーに表示する文字列。
<code>bCc</code>	(オプション) ユーザが CC 受信者を選択できるかどうかを示すブーリアン値。
<code>bBcc</code>	(オプション) ユーザが BCC 受信者を選択できるかどうかを示すブーリアン値。このブーリアンは、 <code>bCc</code> が <code>true</code> の場合にのみ使用してください。それ以外の場合に使用すると、メソッドは失敗します (<code>undefined</code> を返します)。

戻り値

失敗した（ユーザがキャンセルした）場合、`undefined` を返します。成功した場合は、To、CC、BCC の 3 つの文字列を含む配列を返します。

例

ユーザが 2 つのメールアドレスを指定できるようにします。

```
var attempts = 2;
while (attempts > 0)
{
    var recipients = app.mailGetAddrs
    ({
        cCaption: "Select Recipients, Please",
        bBcc: false
    })
}
```

```
if (typeof recipients == "undefined" ) {  
    if (--attempts == 1)  
        app.alert("You did not choose any recipients, try again");  
    } else break;  
}  
if (attempts == 0)  
    app.alert("Cancelling the mail message");  
else {  
    // メールを送信するための JavaScript 文  
}
```

mailMsg

4.0			X
-----	--	--	---

電子メールメッセージを送信します。ユーザ操作を要求するかどうかを選択できます。

Doc オブジェクトの [mailDoc](#) メソッドと [mailForm](#) メソッド、FDF オブジェクトの [mail](#) メソッド、Report オブジェクトの [mail](#) メソッドも参照してください。

注意 : Windows の場合 : このメソッドを使用するためには、クライアントマシンのデフォルトのメールプログラムで MAPI が有効になっている必要があります。

パラメータ

bUI ユーザ操作が必要かどうかを示します。true の場合、メーラーの新規メッセージウィンドウが表示され、他のパラメータ値が初期値として使用されます。false の場合、cTo パラメータが必須で、その他のパラメータはオプションです。

注意 : (Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、bUI パラメータは無視され、デフォルトの true の動作になります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

cTo セミコロンで区切ったアドレスのリスト。

cCc (オプション) セミコロンで区切った CC アドレスのリスト。

cBcc (オプション) セミコロンで区切った BCC アドレスのリスト。

cSubject (オプション) 件名のテキスト。長さの制限は 64 KB です。

cMsg (オプション) メールメッセージのテキスト。長さの制限は 64 KB です。

例

メーラーの新規メッセージウィンドウを表示します。

```
app.mailMsg(true);
```

fun1@example.com と fun2@example.com にメッセージを送信します。

```
app.mailMsg(false, "fun1@example.com; fun2@example.com", "", "",  
"This is the subject", "This is the body of the mail.");
```

フォームデータを含むメッセージを作成することができます。

```
var cMyMsg = "Below are the current budget figures:¥n¥n";
cMyMsg += "Date Compiled: " + this.getField("date").value + "¥n";
cMyMsg += "Current Estimate: " + this.getField("budget").value + "¥n";
app.mailMsg({
    bUI: true,
    cTo: "myBoss@example.com",
    cSubject: "The latest budget figures",
    cMsg: cMyMsg
} );
```

newDoc

5.0			
-----	--	--	--

新しい文書を作成し、その Doc オブジェクトを返します。オプションのパラメータでは、文書のメディアボックスの寸法をポイント単位で指定できます。

注意：このメソッドを実行できるのは、バッチャイントまたはコンソールイベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

nWidth	(オプション) 新しい文書の幅（ポイント単位）。デフォルト値は 612 です。
nHeight	(オプション) 新しい文書の高さ（ポイント単位）。デフォルト値は 792 です。

戻り値

新しく作成された文書のオブジェクト。

例

Acrobat のファイルメニューに「New」という項目を追加します。「New」には、「Letter」、「A4」、「Custom」という 3 つのサブメニュー項目を追加します。このスクリプトは、フォルダレベル JavaScript の.js ファイルで使用します。

```
app.addSubMenu({ cName: "New", cParent: "File", nPos: 0 })
app.addItem({ cName: "Letter", cParent: "New", cExec:
    "app.newDoc();"});
app.addItem({ cName: "A4", cParent: "New", cExec:
    "app.newDoc(420,595)"});
app.addItem({ cName: "Custom...", cParent: "New", cExec:
    "var nWidth = app.response({ cQuestion:'Enter Width in Points',¥
        cTitle:      'Custom Page Size'});"
    +"if (nWidth == null) nWidth = 612;"}
    +"var nHeight = app.response({ cQuestion:'Enter Height in Points',¥
        cTitle: 'Custom Page Size'});"
    +"if (nHeight == null) nHeight = 792;"}
    +"app.newDoc({ nWidth: nWidth, nHeight: nHeight })"});
```

このスクリプトは、7.0 より前のバージョンの Acrobat で動作します。Acrobat 7.0 の場合、正しく動作するためには、「JavaScript」環境設定の「メニュー項目の JavaScript 実行権限を有効にする」にチェックが付いている必要があります。

この項目にチェックが付いていない場合は、`trustedFunction` を介して `app.newDoc` メソッドを実行する必要があります。Acrobat 7.0 以降では、メニューイベントによる JavaScript の実行はセキュリティによって制限されています。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

前述の例と同じ処理は、次のように記述できます。

```
trustedNewDoc = app.trustedFunction( function (nWidth, nHeight)
{
    app.beginPriv();
    switch( arguments.length ) {
        case 2:
            app.newDoc( nWidth, nHeight );
            break;
        case 1:
            app.newDoc( nWidth );
            break;
        default:
            app.newDoc();
    }
    app.endPriv();
})
app.addSubMenu({ cName: "New", cParent: "File", nPos: 0 })
app.addMenuItem({ cName: "Letter", cParent: "New", cExec:
    "trustedNewDoc();"});
app.addMenuItem({ cName: "A4", cParent: "New", cExec:
    "trustedNewDoc(420,595)"});
app.addMenuItem({ cName: "Custom...", cParent: "New", cExec:
    "var nWidth = app.response({ cQuestion:'Enter Width in Points',¥
        cTitle: 'Custom Page Size'});"
    +"if (nWidth == null) nWidth = 612;"}
    +"var nHeight = app.response({ cQuestion:'Enter Height in Points',¥
        cTitle: 'Custom Page Size'});"
    +"if (nHeight == null) nHeight = 792;"}
    +"trustedNewDoc(nWidth, nHeight) "});
```

このコードには、不完全な部分があります。「Custom」メニュー項目の処理では、空の文字列、小さすぎる値、大きすぎる値などが入力されないように確認するためのコードを挿入したほうがよいでしょう。現在の制約については、『PDF Reference』バージョン 1.7 を参照してください。

例

空の文書を作成し、Doc オブジェクトを取得して、透かしを挿入します。

```
var myNewDoc = app.newDoc();
myNewDoc.addWatermarkFromText ("Confidential",0,font.Helv,24,color.red);
```

この例では、Doc オブジェクトの [addWatermarkFromText](#) メソッドを使用しています。

newFDF

6.0			
-----	--	--	--

データが含まれていない新しい FDF オブジェクトを作成します。

注意：このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。Adobe Reader では使用できません。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

戻り値

新しい FDF オブジェクト。

例

FDF を作成して、PDF ファイルを埋め込みます。

```
var fdf = app.newFDF();
fdf.addEmbeddedFile( "/c/myPDFs/myFile.pdf", 1);
fdf.save( "/c/myFDFs/myFile.fdf" );
```

この例は、[app.openFDF](#) の説明の後に、続きが記載されています。

openDoc

5.0			
-----	--	--	--

指定の PDF 文書を開き、その Doc オブジェクトを返します。このオブジェクトを使用して、新たに開いた文書のメソッドを呼び出したり、プロパティの取得や設定を行うことができます。

注意： バッチシーケンスの実行時にはモーダルダイアログボックスが開くため、処理中はユーザインターフェイスによる処理が妨げられます。したがって、このメソッドをバッチシーケンスで実行することはできません。

このメソッドを使用して HTML 文書を開くと例外が発生し、無効な Doc オブジェクトが返されます。例外を検出するには、`app.openDoc` を `try/catch` 構文で囲みます。次の[例 2](#) を参照してください。

パラメータ

cPath	デバイスに依存しない、開く文書のパス。 <code>oDoc</code> を指定した場合は、 <code>oDoc</code> に対する相対パスを指定することができます。対象の文書は、デフォルトのファイルシステムでアクセス可能である必要があります。
cFS	注意： <code>cFS</code> が「CHTTP」に設定されている場合は、コア JavaScript のグローバル関数の <code>encodeURI</code> などを使用して、 <code>cPath</code> 文字列をエスケープする必要があります。次の 例 5 (Acrobat 7.0) を参照してください。
oDoc	(オプション) 相対パスの <code>cPath</code> を解決するための基準となる Doc オブジェクト。デフォルトのファイルシステムでアクセス可能である必要があります。
bHidden	(オプション、Acrobat 7.0) ブーリアン値。 <code>true</code> の場合、ウィンドウを非表示にして PDF ファイルを開きます。デフォルトは <code>false</code> です。

bUseConv	(オプション、Acrobat 7.0) ブーリアン値。cPath で PDF 以外のファイルを参照した場合に使用されます。true の場合は、PDF 以外のファイルを PDF 文書に変換します。デフォルトは false です。
cDest	<p>注意：(Acrobat 7.0) bUseConv を true に設定できるのは、コンソールイベントまたはバッチャイベントのみです。32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」 も参照してください。</p> <p>(オプション、Acrobat 8.0) 文書内の移動先の名前。このパラメータを指定すると、PDF 文書内の指定の名前の移動先が開かれます。名前付きの移動先とその作成方法について詳しくは、『PDF Reference』バージョン 1.7 を参照してください。</p>

戻り値

Doc オブジェクトまたは null。

- Acrobat 5.0 では、Doc オブジェクトを返します。
- Acrobat 5.0.5 では、対象文書の disclosed プロパティが true である場合は Doc オブジェクト、true でない場合は null を返します。
- 「アクセシビリティおよびフォーム機能のパッチ」を適用した Acrobat 5.0.5 と、Acrobat 6.0 以降の openDoc では、次のようにになります。
 - openDoc は、バッチ、コンソール、メニューイベントでは disclosed プロパティを無視して、cPath で指定されたファイルの Doc オブジェクトを返します。
 - その他のイベントでは、disclosed が true の場合は Doc オブジェクト、それ以外の場合は null を返します。

例 1

この例では、別の文書を開いて、テキストフィールドにプロンプトメッセージを挿入し、そのフィールドにフォーカスを設定してから、現在の文書を閉じます。

```
var otherDoc = app.openDoc("/c/temp/myDoc.pdf");
otherDoc.getField("name").value="Enter your name here: "
otherDoc.getField("name").setFocus();
this.closeDoc();
```

次の例は、パスが相対パスになったことを除いて、前の例と同じです。

```
var otherDoc = app.openDoc("myDoc.pdf", this);
otherDoc.getField("name").value="Enter your name here: "
otherDoc.getField("name").setFocus();
this.closeDoc();
```

この例では、Doc オブジェクトの [closeDoc](#) メソッドと Field オブジェクトの [setFocus](#) メソッドを使用しています。

例 2

ユーザのハードドライブ上の HTML 文書を開いて PDF に変換します。

```
try {
    app.openDoc("/c/myWeb/myHomePage.html");
} catch (e) {};
```

例 3 (Acrobat 7.0)

非表示の PDF 文書を開き、そこから情報を抽出して、閉じます。

```
oDoc = app.openDoc({  
    cPath: "/C/myDocs/myInfo.pdf",  
    bHidden: true  
});  
var v = oDoc.getField("myTextField").value;  
this.getField("yourTextField").value = v;  
oDoc.closeDoc();
```

例 4 (Acrobat 7.0)

PDF 以外のファイルを PDF 文書に変換して、開きます。次のスクリプトはコンソールから正しく実行することができます。

```
app.openDoc({  
    cPath: "/c/temp/myPic.jpg",  
    bUseConv: true  
})
```

例 5 (Acrobat 7.0)

WebDAV サーバからファイルを開きます。app.openDoc メソッドには、ファイルのパスをエスケープして渡す必要があります。

```
var myURL = encodeURI("http://www.example.com/My Folder/ComDoc.pdf");  
app.openDoc({cPath: myURL, cFS: "CHTTP"});  
  
app.browseForDoc も参照してください。
```

例 6 (Acrobat 8.0)

文書を開いて、指定の名前の移動先にジャンプします。

```
app.openDoc({ cPath: "/c/temp/myDoc.pdf", cDest: "myDest" });
```

8.0 より前のバージョンでは、disclosed でない文書でこのジャンプを行うことはできませんでした。
[gotoNamedDest](#) の例を参照してください。

openFDF

6.0			
-----	--	--	--

指定のファイルを開いて新しい FDF オブジェクトを作成します。FDF オブジェクトには、その FDF ファイルに含まれているデータを操作するためのメソッドやプロパティが用意されています。

注意：このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。

パラメータ

cDIPath デバイスに依存しない、開くファイルのパス。

戻り値

開いた FDF ファイルの FDF オブジェクト。

例

FDF ファイルを作成して、PDF ファイルを埋め込みます。

```
var fdf = app.newFDF();
fdf.addEmbeddedFile( "/c/myPDFs/myFile.pdf", 1);
fdf.save( "/c/myFDFs/myFile.fdf" ); // この FDF を保存して閉じる

// FDF ファイルを開いて別の PDF 文書を埋め込む
var fdf = app.openFDF( "/c/myFDFs/myFile.fdf" );
fdf.addEmbeddedFile( "/c/myPDFs/myOtherFile.pdf", 1);
fdf.save( "/c/myFDFs/myFile.fdf" ); // この FDF を保存して閉じる
```

別の使用例については、FDF オブジェクトの [signatureSign](#) メソッドを参照してください。

popUpMenu

5.0			
-----	--	--	--

注意：このメソッドの代わりに、popUpMenuEx を使用してください。

現在のマウス位置に、指定の項目を含むポップアップメニューを作成します。

パラメータ

cItem	(オプション) 引数が文字列の場合、メニュー項目としてメニューに表示されます。「-」というメニュー項目名は、メニューのセパレータとして予約されています。
Array	(オプション) 引数が配列の場合、配列の最初の要素が親メニュー項目となるサブメニューとして表示されます。配列の中にさらにサブメニューを含めることができます。

戻り値

選択されたメニュー項目名か、項目が選択されなかった場合は null。

例

文書の章を示すポップアップメニューを作成します。

```
var cChoice = app.popUpMenu("Introduction", "-", "Chapter 1",
    [ "Chapter 2", "Chapter 2 Start", "Chapter 2 Middle",
    ["Chapter 2 End", "The End"]]);
app.alert("You chose the $" + cChoice + "$ menu item");
```

popUpMenuEx

6.0			
-----	--	--	--

現在のマウス位置にポップアップメニューを作成します。パラメータには、ポップアップメニューのメニュー項目を表す `MenuItem` オブジェクトを指定します。

`popUpMenu` の代わりに、このメソッドを使用してください。

パラメータ

1つ以上の `MenuItem` オブジェクト（以下を参照）。

戻り値

選択されたメニュー項目の `cReturn` 値か、その項目に対して `cReturn` が指定されていない場合は `cName`。何も選択されなかった場合は `null` を返します。

MenuItem オブジェクト

この汎用 JavaScript オブジェクトは、メニュー項目を表します。このオブジェクトのプロパティは、次のとおりです。

プロパティ	説明
<code>cName</code>	メニュー項目の名前。メニュー項目には、この文字列が表示されます。「-」という値は、メニューのセパレータとして予約されています。
<code>bMarked</code>	(オプション) 項目にチェックを付けるかどうかを示すブーリアン値。デフォルトは <code>false</code> (マークなし) です。
<code>bEnabled</code>	(オプション) 項目を有効にするかグレーアウトするかを示すブーリアン値。デフォルトは <code>true</code> (有効) です。
<code>cReturn</code>	(オプション) メニュー項目が選択されたときに返される文字列。デフォルトは <code>cName</code> の値です。
<code>oSubMenu</code>	(オプション) サブメニュー項目を表す <code>MenuItem</code> オブジェクト、または、 <code>MenuItem</code> オブジェクトを要素として持つサブメニュー項目の配列。

例 1

`popUpMenuEx` メソッドのすべての機能を示します。

```
var cChoice = app.popUpMenuEx
(
  {cName: "Item 1", bMarked:true, bEnabled:false},
  {cName: "-"},
  {cName: "Item 2", oSubMenu:
    [ {cName: "Item 2, Submenu 1"}, 
      {
        cName: "Item 2, Submenu 2",
        ...
      }
    ]
  }
)
```

```
        oSubMenu: {cName:"Item 2, Submenu 2, Subsubmenu 1", cReturn: "0"}
    }
],
},
{cName: "Item 3"},
{cName: "Item 4", bMarked:true, cReturn: "1"}
)
app.alert("You chose the ¥" + cChoice + "¥ menu item");
```

例 2

popupMenuEx メソッドは、MenuItem オブジェクトのリストをパラメータに取ります。JavaScript 変数をパラメータとして渡すことはできません。JavaScript 変数を使用したい場合は、メニュー項目の配列を作成してから、コア JavaScript に用意されている Function オブジェクトの apply メソッドを使用します。このメソッドを使用すれば、配列として引数を渡すことができます。

```
// ポップアップメニューのプロパティを 1 つの配列として宣言
var aParams = [
    {cName: "Adobe Web Page", cReturn: "www.adobe.com"}, 
    {cName: "-"}, 
    {cName: "The Adobe Acrobat family", 
        cReturn: "http://www.adobe.com/products/Acrobat/main.html"}, 
    {cName: "Adobe Reader", 
        cReturn: "http://www.adobe.com/products/Acrobat/readstep2.html"}];
// 関数 app.popUpMenuEx を、パラメータの配列 aParams を使用して
// app オブジェクトに適用
var cChoice = app.popUpMenuEx.apply( app, aParams );
if ( cChoice != null ) app.launchURL(cChoice);
```

removeToolBar

6.0			
-----	--	--	--

追加されたボタンをツールバーから削除します。

注意：(Acrobat 7.0) addToolBar メソッドで追加したツールボタンを削除するには、addToolBar を実行したときと同じコンテキストで removeToolBar を実行する必要があります。

ボタンの追加時に Acrobat で文書が開かれていなかった場合は、ボタンを削除するときにも Acrobat で文書が開かれていなければなりません。次の[例 2](#)を参照してください。

同様に、ツールボタンの追加時に特定の文書がアクティブになっていた場合は、removeToolBar を使用してボタンを削除するときにも同じ文書がアクティブになっている必要があります。

ツールボタンを追加したときにアクティブであった文書を閉じると、そのボタンは自動的に削除されます。また、[addToolBar](#) の説明の後の注も参照してください。

パラメータ

cName	言語に依存しない、addToolBar で使用された識別子。
-------	--------------------------------

例 1

[addToolButton](#) の例を参照してください。

例 2

この例では、`addToolButton` と同じコンテキストでのツールボタンの削除方法を示します。最初は、Acrobat で開いている文書はありません。コンソールから次のコードを実行します。

```
app.addToolButton({cName: "button1", cExec:"app.alert('pressed')",
    cTooltext:"Button1"});
```

Acrobat で PDF 文書を開き、コンソールから次の行を実行します。

```
app.removeToolButton({cName: "button1"});
```

例外が発生し、ボタンの削除は行われません。

PDF 文書を閉じて、`removeToolButton` スクリプトを再度実行すると、ボタンが削除されます。

response

3.01			
------	--	--	--

質問、および質問に応答するための入力フィールドを含むダイアログボックスを表示します。

パラメータ

cQuestion	ユーザに表示する質問。
cTitle	(オプション) ダイアログボックスのタイトル。
cDefault	(オプション) 質問に対する応答のデフォルト値。指定しなかった場合、デフォルト値は表示されません。
bPassword	(オプション) <code>true</code> の場合、ユーザが入力する内容をアスタリスク (*) または黒丸 (•) でマスクし、他人に見られないようにします。デフォルトは <code>false</code> です。
cLabel	(オプション、Acrobat 6.0) 入力フィールドの左側に表示する短い文字列。

戻り値

ユーザの応答を含む文字列。ユーザが「キャンセル」をクリックした場合、応答は `null` オブジェクトになります。

例

ユーザに質問し、得られた回答を表示します。

```
var cResponse = app.response({
    cQuestion: "How are you today?",
    cTitle: "Your Health Status",
    cDefault: "Fine",
    cLabel: "Response:"
});
if (cResponse == null)
    app.alert("Thanks for trying anyway.");
else
    app.alert("You responded, ¥"+cResponse+"¥", to the health "
        + "question.", 3);
```

setInterval

5.0			
-----	--	--	--

JavaScript スクリプトと時間を指定します。指定の時間が経過するたびにこのスクリプトが実行されます。

このメソッドの戻り値は、JavaScript 変数に格納する必要があります。格納しないと、interval オブジェクトがガーベッジコレクションによって解放され、クロックが停止します。

実行の繰り返しを停止するには、返された interval オブジェクトを `clearInterval` に渡します。

注意：Acrobat 7.05 以降では、`setInterval` を呼び出したスクリプトが含まれている文書を閉じると、インターバルが自動的に停止します（まだ停止していない場合）。

文書レベルの JavaScript ダイアログボックスを開いたり閉じたりすると、JavaScript インタプリタによって文書レベルの JavaScript が再読み込みされ、文書レベルの変数がすべて再初期化されます。文書レベルの変数を使用した JavaScript 式を `setInterval` や `setTimeout` に登録していた場合は、このようにして文書レベルの変数がリセットされると、JavaScript エラーが発生する可能性があります。

[clearInterval](#)、[setTimeout](#)、[clearTimeout](#) も参照してください。

パラメータ

cExpr 実行する JavaScript スクリプト。

nMilliseconds 時間（ミリ秒単位）。

戻り値

interval オブジェクト。

例

「Color」というフィールドで、1秒ごとに色が変化する単純なアニメーションを作成します。

```
function DoIt() {
    var f = this.getField("Color");
    var nColor = (timeout.count++ % 10 / 10);
    // 様々な赤色を作成
    var aColor = new Array("RGB", nColor, 0, 0);
    f.fillColor = aColor;
}
// 戻り値を変数に保存
timeout = app.setInterval("DoIt()", 1000);
// DoIt() がカウントを続けられるように、timeout オブジェクトに
// プロパティを追加
timeout.count = 0;
```

他の例については、[setTimeOut](#) を参照してください。

setTimeOut

5.0			
-----	--	--	--

JavaScript スクリプトと時間を指定します。指定の時間が経過した後に、1回のみこのスクリプトが実行されます。

このメソッドの戻り値は、JavaScript 変数に格納する必要があります。格納しないと、timeout オブジェクトがガーベッジコレクションによって解放され、クロックが停止します。

タイムアウトイベントをキャンセルするには、返された timeout オブジェクトを `clearTimeOut` に渡します。

注意：Acrobat 7.05 以降では、`setInterval` を呼び出したスクリプトが含まれている文書を閉じると、インターバルが自動的に停止します（まだ停止していない場合）。

文書レベルの JavaScript ダイアログボックスを開いたり閉じたりすると、JavaScript インタプリタによって文書レベルの JavaScript が再読み込みされ、文書レベルの変数がすべて再初期化されます。文書レベルの変数を使用した JavaScript 式を `setInterval` や `setTimeOut` に登録していた場合は、このようにして文書レベルの変数がリセットされると、JavaScript エラーが発生する可能性があります。

[clearTimeOut](#)、[setInterval](#)、[clearInterval](#) も参照してください。

パラメータ

cExpr	実行する JavaScript スクリプト。
nMilliseconds	時間（ミリ秒単位）。

戻り値

timeout オブジェクト

例

単純なスクロールテキストを作成します。「marquee」というテキストフィールドがあるものと仮定しています。このフィールドのデフォルト値は「Adobe Acrobat version 8.0 will soon be here!」です。

```
// 文書レベルの JavaScript 関数
function runMarquee() {
    var f = this.getField("marquee");
    var cStr = f.value;
    // フィールド値を取得
    var aStr = cStr.split("");
    aStr.push(aStr.shift());           // 配列に変換
    cStr = aStr.join("");             // 最初の文字を最後に移動
    f.value = cStr;                  // 文字列に戻す
    f.value = cStr;                  // フィールドに新しい値を設定
}

// 「Go」ボタンに「マウスボタンを放す」アクションを挿入
run = app.setInterval("runMarquee()", 100);
// 1 分後に停止
stopRun = app.setTimeout("app.clearInterval(run)", 6000);

// 「Stop」ボタンに「マウスボタンを放す」アクションを挿入
try {
    app.clearInterval(run);
    app.clearTimeout(stopRun);
} catch (e) {}
```

try/catch 構文で「Stop」ボタンのコードをより安全なものにしています。ユーザが「Go」ボタンをクリックせずに「Stop」ボタンをクリックすると、run と stopRun は未定義なので、「Stop」のコードで例外が発生します。しかし、例外が発生した場合には catch コードが実行されるので、ユーザが最初に「Stop」をクリックしても問題は起きません。

trustedFunction

7.0			
-----	--	--	--

関数を「信頼済み」にします。信頼済み関数にすることで、その関数のスタックフレームの現在の権限レベルを明示的に引き上げることができます。セキュリティによる制限があるメソッド（セキュリティによる制限がないコンテキストでしか通常は実行できないメソッド）がスタックフレーム（関数の本体）に含まれている場合は、権限レベルを引き上げることで、セキュリティによる制限があるコンテキストでもその制限を解除してメソッドが実行できるようになります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

関数定義の本体では、次の例に示すように、セキュリティによる制限がないコンテキストでしか通常は実行できないコードを、app.beginPriv メソッドの呼び出しと app.endPriv メソッドの呼び出しで囲む必要があります。

注意：このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。

パラメータ

oFunc	「信頼済み」にする関数を指定する関数オブジェクト。
-------	---------------------------

戻り値

成功した場合は、渡されたものと同じ関数オブジェクトを返します。このメソッドが正常に実行されると、関数オブジェクトは信頼済みになります。エラーが起きた場合は、NotAllowedError が発生します。

構文

このメソッドは、次の 2 つのいずれかの方法で呼び出すことができます。

```
myTrustedFunction = app.trustedFunction(  
    function()  
    {  
        <function body>  
    }  
) ;
```

または

```
function myOtherTrustedFunction()  
{  
    <function body>  
};  
app.trustedFunction(myOtherTrustedFunction) ;
```

次の例と app.trustPropagatorFunction メソッドの例で、信頼済み関数の概念とその主な注意点について、詳細なコメントを付けながら説明します。

例 1

app.newDoc は、セキュリティによる制限があるメソッド（セキュリティによる制限がないコンテキストでしか通常は実行できないメソッド）です。例えば、「マウスボタンを放す」イベントから実行することは、通常はできません。次の例のように、信頼済み関数を作成すれば、「マウスボタンを放す」イベントから実行できるようになります。

ユーザ（またはアプリケーション）の JavaScript フォルダ内にある .js ファイルに、次のスクリプトを記述します。

```
trustedNewDoc = app.trustedFunction( function (nWidth, nHeight)  
{  
    // この上に追加のコードを記述可能  
    app.beginPriv(); // 明示的に権限を引き上げる  
    app.newDoc( nWidth, nHeight );  
    app.endPriv();  
    // この下に追加のコードを記述可能  
})
```

Acrobat を再起動すると、任意の場所から trustedNewDoc 関数を実行できるようになります。例えば、ボタンの「マウスボタンを放す」アクションに次のスクリプトを割り当てて、200 ポイント × 200 ポイントの新規文書を作成することができます。

```
trustedNewDoc( 200, 200 );
```

app.newDoc(200,200) は、セキュリティによる制限があるメソッドなので、「マウスボタンを放す」イベントから実行することは通常はできません。信頼済み関数に組み込むことによって、新規文書の作成を行えるようにしています。

注意：この例では、単純な実装を行っています。この信頼済み関数を改良して、app.newDoc メソッドのように、2 つの引数をオプションにすることができます。

この `trustedNewDoc` 関数は、メニュー項目として実行することもできます。

```
app.addMenuItem( {  
    cName: "myTrustedNewDoc",  
    cUser: "New Doc", cParent: "Tools",  
    cExec: "trustedNewDoc(200,200)", nPos: 0  
} );
```

この例でも、`trustedNewDoc` を改良することができます。例えば、いくつかの `app.response` ダイアログボックスを表示したり、すべてのオプションを完備した単一の `app.execDialog` ダイアログボックスを表示して、ユーザが新規文書のサイズを入力できるようにすることができます。

注意：`app.newDoc` を `app.beginPriv` と `app.endPriv` で囲んでいない場合、セキュリティによる制限があるコンテキストから `trustedNewDoc` を実行すると失敗し、例外が発生します。この例のように、明示的に権限レベルを引き上げる必要があります。

例 2

`app.activeDocs` プロパティは、状況によって動作が異なります。

- コンソールイベントやバッヂイベントでは、すべてのアクティブな文書の配列を返します。
- セキュリティによる制限があるコンテキストでは、アクティブな文書のうち、`disclosed` プロパティが `true` に設定されている文書のみの配列を返します。

セキュリティによる制限があるコンテキストでこの制約を回避したい場合は、信頼済み関数を定義して、`activeDocs` の権限レベルを引き上げます。ユーザ（またはアプリケーション）の JavaScript フォルダ内にある `.js` ファイルで、次の関数を定義します。

```
trustedActiveDocs = app.trustedFunction (   
    function()  
    {  
        app.beginPriv(); // 明示的に権限を引き上げる  
        var d = app.activeDocs;  
        app.endPriv();  
        return d;  
    }  
)
```

次のコードは、フォームボタンの「マウスボタンを放す」アクションから実行できます。

```
var d = trustedActiveDocs();  
console.println("There are d = " + d.length  
+ " files open in the viewer.")  
for ( var i=0; i< d.length; i++)  
    console.println((i+1) + ". " + d[i].documentFileName )
```

コンソールには、`disclosed` であるかどうかに関係なく、ビューアで開かれているすべての文書の番号とファイル名が表示されます。

例 3

信頼済み関数では、現在の権限レベルを明示的に引き上げることができますが、その対象となるのは、その関数のスタックフレームだけです。これに関連する問題を、次の例に示します。

次のコードでは、信頼済み関数のモジュール化を進めています。

```
function mySaveAs(doc, path)
{
    doc.saveAs(doc, path);
}
myFunc = app.trustedFunction( function (doc, path)
{
    // セキュリティによる制限があるコードや、セキュリティによる制限がないコードを記述
    app.beginPriv();
    mySaveAs(doc, path);
    app.endPriv();
    // セキュリティによる制限があるコードや、セキュリティによる制限がないコードを記述
}
```

このコードでは問題が発生します。なぜならば、セキュリティによる制限があるコード (doc.saveAs(doc, path)) が、呼び出し元の信頼済み関数 (myFunc) ではなく、信頼済みでない関数 (mySaveAs) のスタックフレーム (関数本体) で実行されるからです。したがって、セキュリティによる制限があるコンテキストで myFunc を実行すると、例外が発生します。

解決策として、mySaveAs を信頼済み関数にする方法が考えられます。そうすれば、myFunc を実行しても例外が発生しなくなります。しかし、この解決策には問題があります。システムにこの関数が存在することを知つていれば、セキュリティによる制限がある doc.saveAs という関数を、セキュリティによる制限があるコンテキストで誰でも実行できるようになってしまいます。

また、単に beginPriv と endPriv で doc.saveAs(doc, path) を囲むことはできません。セキュリティによる制限があるコンテキストで myFunc を実行すると、mySaveAs 関数の本体に挿入した app.beginPriv で例外が発生します。これは、mySaveAs が信頼済みでないので、権限レベルを引き上げることができないからです。

以上の説明をまとめると、次のような特性を持つ関数が必要であることがわかります。

- 信頼済み関数から呼び出せる。
- それ自身は信頼済みでなく、したがって、セキュリティによる制限があるコンテキストから直接呼び出すことはできない。

これらの条件を満たすのが、信頼伝播関数です（次の [trustPropagatorFunction](#) を参照）。

trustPropagatorFunction

7.0		⌚	
-----	--	---	--

関数を「信頼伝播」関数にします。信頼伝播関数とは、それ自身は信頼済みでなく、信頼済み関数から呼び出された場合に信頼を継承することができる関数のことです。

信頼伝播関数が継承するのは信頼だけで、権限は継承しません。したがって、app.trustedFunction × ソッドと同様に、関数本体に記述されている、セキュリティによる制限がないコンテキストで通常は実行できないコードを、app.beginPriv と app.endPriv で囲む必要があります。

信頼伝播関数は、ユーティリティ関数のようなものとして使用できます。信頼伝播関数は、信頼済み関数や他の信頼伝播関数から呼び出すことはできますが、セキュリティによる制限があるコンテキストで信頼済みでない関数から呼び出すことはできません。

注意： アプリケーションの JavaScript フォルダ内にある .js ファイルで定義されている関数は、暗黙のうちに信頼伝播関数と見なされます。ユーザの JavaScript フォルダ内にある .js ファイルで定義されている関数は、暗黙の信頼伝播関数とは見なされません。

このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。

構文

このメソッドは、次の 2 つのいずれかの方法で呼び出すことができます。

```
myPropagatorFunction = app.trustPropagatorFunction(  
    function()  
    {  
        <function body>  
    }  
) ;
```

または

```
function myOtherPropagatorFunction()  
{  
    <function body>  
};  
app.trustPropagatorFunction(myOtherPropagatorFunction);
```

パラメータ

oFunc	信頼伝播関数にする関数オブジェクト。
-------	--------------------

戻り値

成功した場合は、渡されたものと同じ関数オブジェクトを返します。このメソッドが正常に実行されると、関数オブジェクトは信頼伝播関数になります。エラーが起きた場合は、NotAllowedError が発生します。

例 1

この例の背景情報については、[143 ページの「例 3」](#) を参照してください。

この例では、フォルダにファイルを保存するための信頼伝播関数として、mySaveAs を定義しています。また、様々なタスク（セキュリティによる制限があるコードやセキュリティによる制限がないコードなど）を実行するための信頼済み関数として、myTrustedSpecialTaskFunc を定義しています。mySaveAs 関数は、セキュリティによる制限があるコンテキストから直接呼び出すことはできません。

```
mySaveAs = app.trustPropagatorFunction(function(doc, path)  
{  
    app.beginPriv();  
    doc.saveAs(path);  
    app.endPriv();  
})  
myTrustedSpecialTaskFunc = app.trustedFunction(function(doc, path)  
{
```

```
// この上に、セキュリティによる制限があるコードや、セキュリティによる制限がないコードを記述
app.beginPriv();
    mySaveAs(doc, path);
app.endPriv();
// この下に、セキュリティによる制限があるコードや、セキュリティによる制限がないコードを記述
});
```

次のコードを、ボタンの「マウスボタンを放す」アクションなどに割り当てれば、指定のパスに現在の文書を保存できます。

```
myTrustedSpecialTaskFunc(this, "/c/temp/mySavedDoc.pdf");
```

例 2

次の例では、`app.execDialog` メソッドを使用して簡単なダイアログボックスを作成し、セキュリティによる制限があるコードを実行します。

このダイアログボックスでは、ユーザの名前を尋ね、ローカルハードドライブ（またはネットワークドライブ）上の文書を指定するように求めます。「OK」をクリックすると、選択したファイルがビューアに読み込まれ、文書のプロパティの作成者フィールドにユーザの名前が設定されます（名前の設定は、作成者フィールドが空の場合にのみ行われます）。このダイアログボックスには、セキュリティによる制限がある情報である `identity.email` の値も表示されます。

セキュリティによる制限があるコードはすべて、`beginPriv` と `endPriv` で囲まれています。

この例で使用されている `ANTrustPropagateAll` 関数は、ダイアログボックスでセキュリティによる制限があるコードを使用する場合に役に立ちます。この関数は、1 個のオブジェクトを引数に取り、そのオブジェクト内の各関数を信頼伝播関数にして、オブジェクトを返します。

```
myDialog = app.trustedFunction(function()
{
    app.beginPriv();
    var dialog = ANTrustPropagateAll({
        initialize:function(dialog) {
            this.data = {}; // ダイアログボックスデータを保持するためのオブジェクト
            app.beginPriv();
            dialog.load({ "emai": "Email: " + identity.email });
            app.endPriv();
        },
        commit:function (dialog) { // 「OK」が押されたときに呼び出される
            var results = dialog.store();
            console.println("Your name is " + results["name"]);
            this.data.name = results["name"];
        },
        brws: function (dialog) {
            app.beginPriv();
            var oRetn = app/browseForDoc();
            if ( typeof oRetn != "undefined")
                this.data.oRetn = oRetn;
            app.endPriv();
        },
    });
});
```

```
doDialog:function() {
    app.beginPriv();
    var retn = app.execDialog(this);
    app.endPriv();
    return retn;
},
description: {
    name: "Open File & Populate Info Field",
    align_children: "align_left",
    elements:
    [
        {
            type: "view",
            align_children: "align_left",
            elements:
            [
                {
                    type: "view",
                    align_children: "align_row",
                    elements:
                    [
                        {
                            type: "static_text",
                            name: "Name: "
                        },
                        {
                            item_id: "name",
                            type: "edit_text",
                            alignment: "align_fill",
                            width: 300,
                            height: 20
                        }
                    ]
                },
                {
                    type: "static_text",
                    item_id: "emai",
                    name: "Email: ",
                    char_width: 25
                },
                {
                    type: "gap",
                    height: 10
                },
                {
                    type: "view",
                    align_children: "align_row",
                    elements:
                    [
                        {
                            type: "button",
                            name: "Browse",
                            item_id: "brws"
                        }
                    ]
                }
            ]
        }
    ]
}
```

```
        type: "ok_cancel",
        ok_name: "Ok",
        cancel_name: "Cancel"
    }
]
}
]
}
})
);
app.endPriv();
try { // ユーザが Esc キーを押した場合の対策
    // すべての設定が完了した後、オブジェクトダイアログボックスで指定した
    // doDialog 関数を使用してダイアログボックスを実行
    var retn = dialog.doDialog();
    app.beginPriv();
    // ユーザが「OK」をクリックし、oRetn データがある場合、
    // 制限があるメソッド app.openDoc() を使用して、要求されたファイルを読み込む
    if ( (retn == "ok") && dialog.data.oRetn ) {
        var oDoc = app.openDoc({
            cPath: dialog.data.oRetn.cPath,
            cFS: dialog.data.oRetn.cFS
        });
        if ( !oDoc.info.Author )
            oDoc.info.Author = dialog.data.name;
    }
    app.endPriv();
} catch(e) {}
})
```

このダイアログボックスはボタンから起動することができますが、メニュー項目やツールバーボタンから起動するほうが適切です。例えば、次のコードをユーザの JavaScript ファイルに配置して、ツールメニューにメニュー項目を追加します。

```
app.addMenuItem( { cName: "myDialog", cUser: "My Cool Dialog",
    cParent: "Tools", cExec: "myDialog()", nPos: 0 } );
```

app.media

このオブジェクトには、マルチメディアデータを処理する JavaScript コードに役立つプロパティや関数が用意されています。

app.media のいくつかのプロパティは、様々なプロパティで使用できる値をリストした列挙型のオブジェクトです。この値は、今後のバージョンの Acrobat で追加される可能性があるので、JavaScript コードを作成する際は、ここにリストされていない値にも対応できるようにしてください。同様に、動作環境として想定しているバージョンよりも古いバージョンの Acrobat で JavaScript コードが実行された場合は、そのバージョンで使用可能な値を使用する必要があります。

app.media のプロパティ

align

6.0			
-----	--	--	--

MediaSettings.floating.align プロパティの値が列挙されています。整列は、MediaSettings.floating.over プロパティで指定されているウィンドウを基準として行われます (app.media.[over](#) の値を参照)。

有効な値を、次の表に示します。

値	フローティングウィンドウの位置
app.media.align.topLeft	左上隅
app.media.align.topCenter	上端の中央
app.media.align.topRight	右上隅
app.media.align.centerLeft	左端の中央
app.media.align.center	中央
app.media.align.centerRight	右端の中央
app.media.align.bottomLeft	左下隅
app.media.align.bottomCenter	下端の中央
app.media.align.bottomRight	右下隅

型

オブジェクト (列挙型)

アクセス

R

canResize

6.0			
-----	--	--	--

`MediaSettings.floating.canResize` プロパティの値が列挙されています。これは、フローティングウィンドウのサイズをユーザが変更できるかどうかを指定します。

それらの値を、次の表に示します。

値	説明
<code>app.media.canResize.no</code>	サイズ変更できません
<code>app.media.canResize.keepRatio</code>	縦横比を維持したサイズ変更のみが可能です
<code>app.media.canResize.yes</code>	縦横比を維持しないサイズ変更が可能です

型

オブジェクト（列挙型）

アクセス

R

closeReason

6.0			
-----	--	--	--

Close イベントの `event.reason` プロパティの値が列挙されています。値は、次のとおりです。

```
app.media.closeReason.general
app.media.closeReason.error
app.media.closeReason.done
app.media.closeReason.stop
app.media.closeReason.play
app.media.closeReason.uiGeneral
app.media.closeReason.uiScreen
app.media.closeReason.uiEdit
app.media.closeReason.docClose
app.media.closeReason.docSave
app.media.closeReason.docChange
```

EventListener オブジェクトの [afterClose](#) メソッドや [onClose](#) メソッドを参照してください。

型

オブジェクト（列挙型）

アクセス

R

defaultVisible

6.0			
-----	--	--	--

このプロパティは `true` と定義されています。これは、`MediaSettings.visible` のデフォルト値です

型

ブーリアン

アクセス

R

ifOffScreen

6.0			
-----	--	--	--

`MediaSettings.floating.ifOffScreen` プロパティの値が列挙されています。これは、フローティング ウィンドウの全体または一部が画面の外に出た場合のアクションを指定します。

それらの値と説明を、次の表に示します。

値	説明
<code>app.media.ifOffScreen.allow</code>	何もアクションを行いません
<code>app.media.ifOffScreen.forceOnScreen</code>	画面の中に収まるように、移動やサイズ変更を行います
<code>app.media.ifOffScreen.cancel</code>	メディアクリップの再生をキャンセルします

型

オブジェクト（列挙型）

アクセス

R

layout

6.0			
-----	--	--	--

`MediaSettings.layout` プロパティの値が列挙されています。

それらの値と説明を、次の表に示します。

値	説明
<code>app.media.layout.meet</code>	すべてのコンテンツが収まるように拡大縮小されます。縦横比は維持され、クリッピングは行われず、背景は塗りつぶされます。
<code>app.media.layout.slice</code>	ウィンドウ内がすべて埋め尽くされるように拡大縮小されます。縦横比は維持され、X または Y が必要に応じてクリッピングされます。
<code>app.media.layout.fill</code>	ウィンドウ内がすべて埋め尽くされるように、X と Y が別個の倍率で拡大縮小されます。
<code>app.media.layout.scroll</code>	通常のサイズ（スクロール付き）。
<code>app.media.layout.hidden</code>	通常のサイズ（クリッピングあり）。
<code>app.media.layout.standard</code>	プレーヤーのデフォルト設定を使用します。

型

オブジェクト（列挙型）

アクセス

R

monitorType

6.0			
-----	--	--	--

`MediaSettings.monitorType` プロパティの値が列挙されています。

それらの値と説明を、次の表に示します。

値	説明
<code>app.media.monitorType.document</code>	文書ウィンドウの最も大きいセクションが含まれているモニタ
<code>app.media.monitorType.nonDocument</code>	文書ウィンドウの最も小さいセクションが含まれているモニタ
<code>app.media.monitorType.primary</code>	プライマリモニタ
<code>app.media.monitorType.bestColor</code>	色深度が最も深いモニタ

値	説明
app.media.monitorType.largest	領域が最も広いモニタ（平方ピクセル単位）
app.media.monitorType.tallest	高さが最も高いモニタ（ピクセル単位）
app.media.monitorType.widest	幅が最も広いモニタ（ピクセル単位）

型

オブジェクト（列挙型）

アクセス

R

openCode

6.0			
-----	--	--	--

MediaPlayer.open の戻り値の code プロパティの値が列挙されています。値は、次のとおりです。

```
app.media.openCode.success
app.media.openCode.failGeneral
app.media.openCode.failSecurityWindow
app.media.openCode.failPlayerMixed
app.media.openCode.failPlayerSecurityPrompt
app.media.openCode.failPlayerNotFound
app.media.openCode.failPlayerMimeType
app.media.openCode.failPlayerSecurity
app.media.openCode.failPlayerData
```

型

オブジェクト（列挙型）

アクセス

R

over

6.0			
-----	--	--	--

`MediaSettings.floating.over` プロパティの値が列挙されています。これは、フローティングウィンドウの整列に使用されます。`app.media.align` を参照してください。

値	説明
<code>app.media.over.pageWindow</code>	文書（ページ）ウィンドウを基準としてフローティングウィンドウを整列します
<code>app.media.over.appWindow</code>	アプリケーションウィンドウを基準としてフローティングウィンドウを整列します
<code>app.media.over.desktop</code>	仮想デスクトップ全体を基準としてフローティングウィンドウを整列します
<code>app.media.over.monitor</code>	（選択されている）モニタ画面を基準としてフローティングウィンドウを整列します

型

オブジェクト（列挙型）

アクセス

R

pageEventNames

6.0			
-----	--	--	--

ページレベルアクションの `event.name` プロパティの値が列挙されています。ユーザによる直接アクションを表すイベント名は、ここには含まれていません。この列挙型は、ページレベルアクションとユーザアクションを区別するために使用されます。値は、次のとおりです。

```
app.media.pageEventNames.Open
app.media.pageEventNames.Close
app.media.pageEventNames.InView
app.media.pageEventNames.OutView
```

型

オブジェクト（列挙型）

アクセス

R

例

`app.media.pageEventNames` を使用して、ページレベルアクションとユーザによる直接アクションを区別できます。次のスクリプトは、文書内の任意の場所から呼び出せる、フォルダレベルまたは文書レベルの JavaScript です。

```
function myMMfunction () {
    if ( app.media.pageEventNames[event.name] ) {
        console.println("Page Event: " + event.name);
        ...
    } else {
        console.println("User Generated Event: " + event.name);
        ...
    }
}
```

raiseCode

6.0			
-----	--	--	--

例外が発生した場合の `error.raiseCode` プロパティの値が列挙されています。このプロパティは、`error.name` が「RaiseError」の場合にのみ存在します。ここに示されていない値が使用されることもあります。

```
app.media.raiseCode.fileNotFound
app.media.raiseCode.fileOpenFailed
```

型

オブジェクト（列挙型）

アクセス

R

raiseSystem

6.0			
-----	--	--	--

例外が発生した場合の `error.raiseSystem` プロパティの値が列挙されています。このプロパティは、`error.name` が「RaiseError」の場合にのみ存在します。

```
app.media.raiseSystem.fileError
```

これ以外の値が使用されることもあります。

型

オブジェクト（列挙型）

アクセス

R

renditionType

6.0			
-----	--	--	--

`Rendition.type` の値が列挙されています。それらの値と説明を、次に示します。

値	説明
<code>app.media.renditionType.unknown</code>	このバージョンの Acrobat では認識されないタイプ
<code>app.media.renditionType.media</code>	メディアレンディション
<code>app.media.renditionType.selector</code>	レンディションセレクタ

型

オブジェクト（列挙型）

アクセス

R

status

6.0			
-----	--	--	--

`Status` イベントの `event.media.code` プロパティの値が列挙されています ([onStatus](#) および [afterStatus](#) を参照)。ほとんどの値は、`event.text` プロパティで追加情報が提供されます。値は、次のとおりです。

値	説明
<code>app.media.status.clear</code>	空の文字列（このステータスイベントは任意のメッセージをクリアします）
<code>app.media.status.message</code>	一般的なメッセージ
<code>app.media.status.contacting</code>	接続しているホスト名
<code>app.media.status.buffering</code>	進捗メッセージ、または何もない
<code>app.media.status.init</code>	初期化しているエンジンの名前
<code>app.media.status.seeking</code>	空の文字列

`event.media.status` コードのほかに、現在のステータスを表す前述の `event.media.text` という文字列もあります。

型

オブジェクト (列挙型)

アクセス

R

trace

6.0			
-----	--	--	--

このプロパティを `true` に設定すると、プレーヤーの作成中やイベントのディスパッチ中に、トレースメッセージが JavaScript コンソールに出力されます。

注意： `app.media.trace` は、テスト目的でのみ用意されているプロパティです。このプロパティは、実際に配布する PDF ファイルでは使用しないでください。これは、今後のバージョンの Acrobat で変更される予定です。

型

ブーリアン

アクセス

R / W

version

6.0			
-----	--	--	--

マルチメディア API のバージョン番号。現在は 7.0 です。

型

数値

アクセス

R

windowType

6.0			
-----	--	--	--

`MediaSettings.windowType` プロパティの値が列挙されています。それらの値を、次の表に示します。

値	説明
<code>app.media.windowType.docked</code>	PDF ページにドッキング
<code>app.media.windowType.floating</code>	フローティング（ポップアップ）ウィンドウ
<code>app.media.windowType.fullScreen</code>	フルスクリーンモード

型

オブジェクト（列挙型）

アクセス

R

app.media のメソッド

addStockEvents

6.0			
-----	--	--	--

`MediaPlayer` オブジェクトにストックイベントリスナを追加し、後で削除できるように、`player.stockEvents` にそれらのリスナへの参照を設定します。

オプションの `annot` パラメータを指定した場合は、その注釈への参照が `MediaPlayer.annot` に保存されます。プレーヤーが `MediaPlayer.open` で開かれると、この注釈にもストックイベントリスナが追加され、`annot.player` にプレーヤーへの参照が設定されます。

パラメータ

player	必須の <code>MediaPlayer</code> オブジェクト
annot	（オプション） <code>screenAnnot</code> オブジェクト

ストックイベントリスナには、フォーカスの処理などの、Acrobat の標準的な動作が用意されています。

`app.media.trace` が `true` の場合は、ストックイベントリスナとともにデバッグトレースリスナも含まれます。

`addStockEvents` で追加したイベントリスナを削除するには、`removeStockEvents` メソッドを使用します。

`app.media.createPlayer` メソッドや `app.media.openPlayer` メソッドの内部で `addStockEvents` が呼び出されているので、このメソッドを自分で呼び出す必要はありません。

alertFileNotFound

6.0			
-----	--	--	--

ファイルが見つからないことを示す標準の警告を表示します。オプションで、「次回から表示しない」チェックボックスも提供できます。

パラメータ

<code>oDoc</code>	<code>oDoc</code> は、警告を関連付ける文書です
<code>cFilename</code>	<code>cFilename</code> は、見つからないファイルの名前です
<code>bCanSkipAlert</code>	(オプション) <code>bCanSkipAlert</code> が <code>true</code> で、ユーザがチェックボックスにチェックを付けると、 <code>true</code> が返されます。それ以外の場合は <code>false</code> が返されます。デフォルトは <code>false</code> です。

戻り値

`bCanSkipAlert` が `true` で、ユーザがチェックボックスにチェックを付けると、`true` が返されます。それ以外の場合は `false` が返されます。

例

```
if ( !doNotNotify )
{
    var bRetn = app.media.alertFileNotFound(this, cFileClip, true );
    var doNotNotify = bRetn;
}
```

alertSelectFailed

6.0			
-----	--	--	--

`rendition.select` が失敗したことを示す標準の警告を表示します。

パラメータ

<code>oDoc</code>	警告を関連付ける文書
<code>oRejects</code>	(オプション) 指定した場合、この <code>MediaReject</code> オブジェクトの配列が <code>PlayerInfoList.select</code> で返されます。
<code>bCanSkipAlert</code>	(オプション) <code>true</code> で、ユーザがチェックボックスにチェックを付けると、 <code>true</code> が返されます。それ以外の場合は <code>false</code> が返されます。デフォルトは <code>false</code> です。
<code>bFromUser</code>	(オプション) 警告メッセージの内容に影響を与えるブーリアン値。ユーザによる直接的なアクションによってこのコードが呼び出された場合は、 <code>true</code> にします。その他のアクション（しおりの選択など）によってこのコードが呼び出された場合は、 <code>false</code> にします。デフォルトは <code>false</code> です。

戻り値

`bCanSkipAlert` が `true` で、ユーザがチェックボックスにチェックを付けると、`true` が返されます。それ以外の場合は `false` が返されます。

注意：`rendition.select` で使用可能なプレーヤーが見つからず、`select` の `bWantRejects` というパラメータが `true` に設定されている場合は、戻り値の `MediaSelection` オブジェクトに `MediaReject` オブジェクトの配列が含まれています。この配列を、このメソッドの `oRejects` パラメータに渡すことができます。すると、Web にアクセスして適切なプレーヤーをダウンロードするように求めるメッセージが、`alertSelectFailed` メソッドによってユーザに表示されます。

例

メディアクリップが再生できないというメッセージを、チェックボックス付きで表示します。

```
var bRetn = app.media.alertSelectFailed({  
    oDoc: this,  
    bCanSkipAlert: true  
});
```

argsDWIM

6.0			
-----	--	--	--

このメソッドは、`app.media.createPlayer`、`app.media.openPlayer`、`app.media.startPlayer` 関数で使用される、「Do What I Mean」機能（ユーザの意図を汲み取って適切な処理を行う機能）の一種です。このメソッドに `PlayerArgs` オブジェクトを渡すと、指定されていないプロパティにデフォルト値（`event` オブジェクトから取得されたもの）が指定されます。したがって、前述の関数を、引数を持たないレンディションアクションイベントハンドラとして使用したり、明示的な引数を持つカスタムの JavaScript で使用したりすることができます。

パラメータ

args `PlayerArgs` オブジェクト (`app.media.createPlayer` を参照)。

戻り値

`PlayerArgs` オブジェクト

例

使用例については、[163 ページの「例 1」](#) を参照してください。

canPlayOrAlert

6.0			
-----	--	--	--

メディアの再生が許可されているかどうかを確認し、許可されている場合は `true` を返します。再生が許可されていない場合はユーザに警告し、`false` を返します。

パラメータ

args PlayerArgs オブジェクト (`app.media.createPlayer` を参照)。

戻り値

メディアの再生が許可されている場合は `true`、許可されていない場合は `false`。

注意： `createPlayer` メソッドでは、プレーヤーの作成を試みる前に、この関数を呼び出しています。
`createPlayer` の代わりに自分で作成したコードを使用する場合は、`canPlayOrAlert` を呼び出して、再生が許可されない状況（マルチメディアオーサリングモードなど）でユーザに警告することができます。

`args` オブジェクトのプロパティの中で使用されるのは `doc` のみなので、次のようにすることができます。

```
// myDoc には Doc が保存されている
if( app.media.canPlayOrAlert({ doc: myDoc })
/* OK であれば、プレーヤーを作成する */;
```

このコードでは、「オーサリングモードではメディアを再生できません」などの適切な警告が表示されます。

computeFloatWinRect

6.0			
-----	--	--	--

パラメータで指定した必要な矩形を画面座標で計算して返します。

パラメータ

doc 文書の Doc オブジェクト。

floating MediaSettings.floating によって返されるオブジェクトのフローティングパラメータ。

monitorType 使用するモニタを示す数値。`app.media.monitorType` というプロパティを参照してください。

uiSize (オプション) ユーザインターフェイスのサイズ。`MediaPlayer.uiSize` で返される、サイズを表す 4 つの数値の配列 [w, x, y, z]。

戻り値

画面座標での矩形。

例

フローティングウィンドウの矩形を取得します。

```
var floating =
{
  over: app.media.over.monitor,
  align: app.media.align.center,
  canResize: app.media.canResize.no,
```

```
    hasClose: false,  
    hasTitle: true,  
    width: 400,  
    height: 400  
}  
var rect = app.media.computeFloatWinRect  
(this, floating, app.media.monitorType.primary);
```

constrainRectToScreen

6.0			
-----	--	--	--

画面座標の矩形を返します。一部のモニタでは、矩形全体を表示するために移動やサイズ変更が行われます。anchorPt を指定すると、rect が縮小される場合に anchorPt (点を表す 2 つの数値の配列 [x, y]) に向かって均等に縮小されます。

パラメータ

rect	目的の矩形の画面座標を表す 4 つの数値の配列。
anchorPt	(オプション) アンカーポイントを表す 2 つの数値の配列 [x, y]。

戻り値

画面座標での矩形。

createPlayer

6.0			
-----	--	--	--

プレーヤーを開かずに、MediaPlayer オブジェクトを作成します。

注意： プレーヤーを開くには、MediaPlayer.open を使用します。createPlayer ではなく app.media.openPlayer を使用すれば、この 2 つの手順を一度で行えます。

レンディションアクション（例えば、マルチメディアのプロパティダイアログボックスの「アクション」タブで入力したカスタムの JavaScript）で createPlayer を呼び出すと、そのアクションの event オブジェクトからデフォルト値が取得されます。この場合、レンディションアクションの値をオーバーライドする必要がない限り、args パラメータを指定する必要はありません。createPlayer は、argsDWIM を呼び出して、event オブジェクトと args ([PlayerArgs オブジェクト](#)を参照) パラメータを処理します。

PlayerArgs オブジェクトの noStockEvents が true に設定されていない限り、MediaPlayer オブジェクトには、Acrobat と適切にやり取りを行うために必要な標準の動作を提供するストックイベントリスナが用意されます。他のイベントリスナは、PlayerArgs オブジェクトで指定することもできますし、MediaPlayer.events.add を使用して後で追加することもできます。

args.annot.player が開いている MediaPlayer である場合、createPlayer はそのプレーヤーを閉じ、それによってイベントが発生します。

パラメータ

args	PlayerArgs オブジェクト。必須およびオプションのプロパティについては、次を参照してください。
	関連するレンディションを持つレンディションアクションの中で <code>createPlayer</code> を実行する場合、このパラメータはオプションです。この場合、そのプロパティは、デフォルトおよび UI で選択されたオプションに指定されます。そうでない場合、このパラメータは必須です。

PlayerArgs オブジェクト

プロパティ	型	説明
doc	オブジェクト	文書の Doc オブジェクト。 <code>annot</code> と <code>rendition</code> を両方省略した場合は必須です（例えば URL の再生など）。
annot	オブジェクト	ScreenAnnot オブジェクト。 <code>event</code> オブジェクトにある場合や、 <code>MediaSettings.page</code> が指定されている場合を除いて、ドッキング表示の再生では必須です。新しいプレーヤーは、この注釈に関連付けられます。プレーヤーに注釈が既に関連付けられている場合、プレーヤーは停止され、閉じられます。
rendition	オブジェクト	(オプション) Rendition オブジェクト (<code>MediaRendition</code> か <code>RenditionList</code> のいずれか)。 <code>event</code> オブジェクトに <code>rendition</code> がある場合や、URL が指定されている場合を除いて、必須です。
URL	文字列	URL または <code>rendition</code> が必要です。URL のほうが優先されます。
mimeType	文字列	(オプション) URL が存在しない場合は無視されます。URL が存在する場合は、 <code>app.media.getPlayers</code> で返される <code>mimeType</code> または <code>settings.players</code> が必要です。
settings	オブジェクト	(オプション) <code>MediaSettings</code> オブジェクト。 <code>rendition</code> の設定をオーバーライドします。
events	オブジェクト	(オプション) <code>EventListener</code> オブジェクト。ストックイベントが使用される場合はオプション。ストックイベントの後に追加されます。
noStockEvents	ブーリアン	(オプション) <code>true</code> の場合、ストックイベントを使用しません。デフォルトは <code>false</code> です。
fromUser	ブーリアン	(オプション) ユーザによる直接的なアクションによってこのコードが呼び出される場合は、 <code>true</code> にします。それ以外の場合は、 <code>false</code> にします。デフォルトは <code>event</code> オブジェクトに依存します。
showAltText	ブーリアン	(オプション) <code>true</code> の場合、メディアが再生できないときに代替テキスト (<code>Rendition.altText</code> を参照) を表示します。デフォルトは <code>true</code> です。
showEmptyAltText	ブーリアン	(オプション) <code>true</code> で、代替テキスト (<code>Rendition.altText</code> を参照) が空の場合、空のボックスとして代替テキストを表示します。 <code>false</code> の場合は警告が表示されます。 デフォルト値は、 <code>fromUser</code> が <code>false</code> の場合は <code>true</code> で、 <code>fromUser</code> が <code>true</code> の場合は <code>false</code> です。

戻り値

MediaPlayer オブジェクト

例 1

次のコードは、`openPlayer` の定義です。この内で `createPlayer` が使用されています。

```
app.media.openPlayer = function( args )
{
    var player = null;
    try
    {
        args = app.media.argsDWIM( args );

        player = app.media.createPlayer( args );
        if( player )
        {
            var result = player.open();
            if( result.code != app.media.openCode.success )
            {
                player = null;
                app.media.alert(
                    ( "Open", args, { code: result.code } ) );
            }
            else if( player.visible )
                player.setFocus(); // Focus イベントを発生
        }
    }
    catch( e )
    {
        player = null;
        app.media.alert( 'Exception', args, { error: e } );
    }

    return player;
}
```

例 2

`PlayerArgs` の使用例については、[openPlayer](#) の説明の最後にある例を参照してください。

getAltTextData

6.0			
-----	--	--	--

指定のテキストの代替テキストデータを表す MediaData オブジェクト (`MediaSettings`[data](#) を参照) を返します。このオブジェクトは、プレーヤーを作成して代替テキストを表示するのに使用できます。

パラメータ

cAltText	代替テキストデータとして使用する文字列。
----------	----------------------

戻り値

MediaSettings オブジェクト (MediaSettings[data](#) を参照)。

例

getAltTextSettings メソッドの例を参照してください。

getAltTextSettings

6.0			
-----	--	--	--

PlayerArgs オブジェクト (少なくとも settings、showAltText、showEmptyAltText の各プロパティが設定されている必要があります) と、rendition.select で返される selection オブジェクトを引数として取り、使用可能な最初の代替テキストレンディションを検索します (存在する場合)。そして、代替テキストの再生に適した新しい MediaSettings オブジェクトを作成して返します。代替テキストレンディションが存在しない場合は、null を返します。

パラメータ

args	PlayerArgs オブジェクト
selection	MediaSelection オブジェクト

戻り値

MediaSettings オブジェクトまたは null

例

この例では、レンディションの代替テキストを再生します。このコードでは、フローティングではなく、PDF 画面に埋め込んだ状態で代替テキストを再生しますが、フローティングウィンドウで再生するように変更することができます。

```
var rendition = this.media.getRendition("myClip");
var settings = rendition.getPlaySettings();
var args = {
    settings: settings,
    showAltText: true,
    showEmptyAltText: true
};
var selection = rendition.select();
settings = app.media.getAltTextSettings( args, selection );

// 次の行のコメントを解除すれば、カスタムの代替テキストを再生できる
// settings.data = app.media.getAltTextData("A. C. Robat");

// 次のコードのコメントを解除すれば、代替テキスト再生用のフローティング
// ウィンドウが取得される
/*
settings.windowType = app.media.windowType.floating
settings.floating = {
    canResize: app.media.canResize.keepRatio,
```

```
    hasClose: true,
    width: 400,
    height: 100
} */

// 次に、代替テキストを再生する openPlayer で使用する
// args パラメータを定義
args = {
    rendition: rendition,
    annot: this.media.getAnnot({nPage: 0, cAnnotTitle:"myScreen"}),
    settings: settings
};
app.media.openPlayer(args);
```

getAnnotStockEvents

6.0			
-----	--	--	--

画面注釈を Acrobat で通常再生するのに必要なストックイベントリスナを含んだ event オブジェクトを返します。ストックイベントリスナには、フォーカスの処理などの、Acrobat の標準的な動作が用意されています。

app.media.trace が true の場合は、ストックイベントリスナとともにデバッグトレースリスナも含まれます。

パラメータ

settings	ウィンドウのタイプ (app.media. windowType を参照) に対応する数値。
----------	--

戻り値

event オブジェクト

getAnnotTraceEvents

6.0			
-----	--	--	--

イベントがディスパッチされたときのデバッグトレースを提供するイベントリスナを含んだ Events オブジェクトを返します。

戻り値

Events オブジェクト

getPlayers

6.0			
-----	--	--	--

PlayerInfoList オブジェクトを返します。これは、利用可能なメディアプレーヤーを表す PlayerInfo オブジェクトの配列です。

PlayerInfoList は、その select メソッドでフィルタリングでき、createPlayer でメディアプレーヤーを作成するときに settings.players プロパティの中で使用できます。

詳しくは、[PlayerInfoList](#) オブジェクトおよび [PlayerInfo](#) オブジェクトを参照してください。

パラメータ

cMimeType	(オプション) 「audio/wav」などの、オプションの MIME タイプ。cMimeType を省略すると、リストには利用可能なプレーヤーがすべて含まれます。cMimeType を指定すると、リストにはその MIME タイプを扱えるプレーヤーのみが含まれます。
-----------	--

戻り値

PlayerInfoList オブジェクト

例 1

デバッグコンソールに MP3 プレーヤーをリストします。

```
var mp = app.media.getPlayers("audio/mp3")
for ( var i = 0; i < mp.length; i++ ) {
    console.println("mp[" + i + "] Properties");
    for ( var p in mp[i] ) console.println(p + ": " + mp[i][p]);
}
```

例 2

Flash メディアを再生可能なプレーヤーを、MIME タイプを指定して選択します。このコードは、レンディションに関連付けられているレンディションアクションとして実行されるものと仮定しています（したがって、createPlayer の引数は必要ありません）。

```
var player = app.media.createPlayer();
player.settings.players
    = app.media.getPlayers( "application/x-shockwave-flash" );
player.open();
```

getPlayerStockEvents

6.0			
-----	--	--	--

メディアプレーヤーを Acrobat で通常再生するのに必要なストックイベントリスナを含んだ Events オブジェクトを返します。ストックイベントリスナには、フォーカスの処理などの、Acrobat の標準的な動作が用意されています。

それらのストックイベントをメディアプレーヤーに追加するには、MediaPlayer.events.add を使用します。

app.media.createPlayer メソッドや app.media.openPlayer メソッドの内部で getPlayerStockEvents が自動的に呼び出されているので、すべてのストックイベントリスナを明示的にセットアップするコードを作成する場合を除き、このメソッドを自分で呼び出す必要はありません。

app.media.trace が true の場合は、ストックイベントリスナとともにデバッグトレースリスナも含まれます。

パラメータ

settings MediaSettings オブジェクト。

戻り値

Events オブジェクト

getPlayerTraceEvents

6.0			
-----	--	--	--

イベントがディスパッチされたときのデバッグトレースを提供するイベントリスナを含んだ Events オブジェクトを返します。

戻り値

Events オブジェクト

getRenditionSettings

6.0			
-----	--	--	--

Rendition.select を呼び出して MediaSelection オブジェクトを取得し、MediaSelection.rendition.getPlaySettings を呼び出して再生用の MediaSettings オブジェクトを取得します。このいずれかが失敗した場合は、getAltTextSettings メソッドを呼び出して、代替テキスト再生用の MediaSettings オブジェクトを取得します。最後に、MediaSettings オブジェクトを返すか、getAltTextSettings で null が返された場合（代替テキストが指定されていないか許可されていない場合）は null を返します。

パラメータ

args PlayerArgs オブジェクト。

戻り値

MediaSettings オブジェクトまたは null

例

[171 ページの「例 3」](#) を参照してください。

getURLData

6.0			
-----	--	--	--

指定の URL にある、指定の MIME タイプのデータを表す MediaData オブジェクト（MediaSettings [data](#) を参照）を返します。この MediaData オブジェクトは、その URL のデータにアクセスするプレーヤーを作成するのに使用できます。

パラメータ

cURL	メディアデータの取得先 URL。
cMimeType	(オプション) データの MIME タイプ。

戻り値

MediaData オブジェクト

例

インターネットからメディアクリップを取得して、フローティングウィンドウでそれを再生します。

```
var myURLClip = "http://www.example.com/myClip.mpg";
var args = {
    URL: myURLClip,
    mimeType: "video/x-mpg",
    doc: this,
    settings: {
        players: app.media.getPlayers("video/x-mpg"),
        windowType: app.media.windowType.floating,
        data: app.media.getURLData(myURLClip,"video/x-mpg"),
        floating: { height: 400, width: 600 }
    }
}
app.media.openPlayer(args);
```

getURLSettings

6.0			
-----	--	--	--

settings プロパティを持つ PlayerArgs オブジェクトを引数に取り、URL の再生に適した MediaSettings オブジェクトを返します。settings プロパティには URL プロパティが含まれている必要があります。また、mimeType プロパティを指定することができます。出力される settings オブジェクトにコピーするその他の設定も含めることができます。

パラメータ

args	PlayerArgs オブジェクト。
------	--------------------

戻り値

MediaSettings オブジェクト

例 1

前の例と同じです。getURLSettings は getURLData を呼び出して、返された MediaData オブジェクトを setting の data プロパティに挿入して返します。

```
var myURLClip = "http://www.example.com/myClip.mpg";
args = {
    URL: myURLClip,
    mimeType: "video/x-mpg",
    doc: this,
    settings:
    {
        players: app.media.getPlayers("video/x-mpg"),
        windowType: app.media.windowType.floating,
        floating: { height: 400, width: 600 }
    }
};
settings = app.media.getURLSettings(args)
args.settings = settings;
app.media.openPlayer(args);
```

例 2

次の例は、コンボボックスのカスタムのキーストロークアクションです。このコンボボックスは、ストリーム配信の音声／動画 Web サイトの単純なプレイリストです。リスト内の各要素の書き出し値は、次のように、「URL,MIME タイプ」の形式になっています。

```
http://www.example.com/streaming/radio.aspx,video/x-ms-asx
```

次のコードでは、2 要素の配列に書き出し値を分割しています。最初の要素が URL で、2 番目の要素が MIME タイプになります。動画の場合は、「myScreen」という画面注釈に表示されます。それ以外の場合は、音声のみが聞こえます。

```
if (!event.willCommit)
{
    var aURLMime = event.changeEx.split(",")
    console.println("aURLMime[0] = " + aURLMime[0]);
    console.println("aURLMime[1] = " + aURLMime[1]);
    var args = {
        annot:this.media.getAnnot({ nPage:0,cAnnotTitle: "myScreen" }),
        URL: aURLMime[0],
        mimeType: aURLMime[1],
        doc: this,
        settings: {
            players: app.media.getPlayers(aURLMime[1]),
            windowType: app.media.windowType.docked
        }
    };
    settings = app.media.getURLSettings(args);
    args.settings = settings;
    var player = app.media.openPlayer(args);
}
```

getWindowSizeBorderSize

6.0			
-----	--	--	--

パラメータで指定したプロパティを持つフローティングウィンドウで使用されている、左、上、右、下の境界のサイズ（ピクセル単位）を表す4つの数値からなる配列を返します。

`hasTitle` および `hasClose` パラメータはブーリアン値で、`canResize` は `app.media.canResize` の任意の値を取ります。

これらのパラメータは、`MediaSettings.floating` オブジェクトのプロパティと同じ名前を持っているので、単一のパラメータとしてフローティングオブジェクトを単純に渡すことができます。

```
var size = doc.media.getWindowWindowSizeBorderSize( settings.floating );
```

パラメータ

`hasTitle` (オプション) デフォルトは `true` です。

`hasClose` (オプション) デフォルトは `true` です。

`canResize` (オプション) デフォルトは `app.media.canResize.no` です。

戻り値

4つの数値の配列。

openPlayer

6.0			
-----	--	--	--

`app.media.createPlayer` を呼び出して `MediaPlayer` オブジェクトを作成してから、`MediaPlayer.open` を呼び出してプレーヤーを開きます。

このメソッドによって、いくつかのイベントが発生します。例えば、Ready ([onReady](#) および [afterReady](#) を参照)、Play ([onPlay](#) および [afterPlay](#) を参照)、Focus ([onFocus](#) および [afterFocus](#) を参照) などが発生する場合があります。これらのイベントの一般的な説明については、Event Listener オブジェクトを参照してください。

このメソッドが失敗した場合はユーザに警告が表示され、`null` が返されます。例外は発生しません。

パラメータ

`args` (オプション) `PlayerArgs` オブジェクト。

戻り値

`MediaPlayer` オブジェクト。失敗した場合は `null`。

例 1

この単純な例は、画面注釈のマルチメディアのプロパティダイアログボックスの「アクション」タブで使用するカスタム JavaScript です。画面注釈の UI で指定したパラメータをオーバーライドするために、args パラメータを渡しています。

```
app.media.openPlayer();
```

settings.repeat をオーバーライドします。repeat が 1 に設定されている場合は 2 に変更し、1 でない場合は 1 に設定します。

```
var nRepeat =
  ( event.action.rendition.getPlaySettings().repeat == 1 ) ? 2 : 1;
var args = { settings: { repeat: nRepeat } };
app.media.openPlayer(args);
```

event.action.rendition について詳しくは、[event](#) オブジェクトを参照してください。この例では、Rendition.getPlaySettings を使用して、再生するレンディション（画面注釈に関連するレンディション）の設定にアクセスしています。

例 2

次のスクリプトは、フォームボタンの「マウスボタンを放す」アクションから実行します。画面注釈で、ドッキングされたメディアクリップを再生します。

```
app.media.openPlayer({
  rendition: this.media.getRendition( "myClip" ),
  annot: this.media.getAnnot( {nPge:0,cAnnotTitle:"myScreen"} ),
  settings: { windowType: app.media.windowType.docked }
});
```

例 3

この例は、画面注釈のマルチメディアのプロパティの「アクション」タブで使用するカスタム JavaScript です。ユーザが注釈をクリックすると、ランダムに選択されたムービークリップが再生されます。

```
// このコードは、文書レベル JavaScript の最上位に置く
var myRenditions = new Array();
myRenditions[0] = "myClip1";
myRenditions[1] = "myClip2";
myRenditions[2] = "myClip3";

// このコードは ScreenAnnot のカスタム JavaScript。すべての
// レンディションはドッキングされ、ScreenAnnot で再生される。
var l = myRenditions.length;
randomIndex = Math.floor( Math.random() * l ) % l;

var rendition = this.media.getRendition(myRenditions[randomIndex]);
var settings = app.media.getRenditionSettings({ rendition: rendition });

var args = { rendition: rendition, settings: settings }
app.media.openPlayer(args);
```

removeStockEvents

6.0			
-----	--	--	--

MediaPlayer オブジェクトおよび関連付けられているすべての ScreenAnnot オブジェクトからすべてのストックイベントリスナを削除し、player.stockEvents、player.annot、annot.stockEvents、annot.player プロパティを削除します。これによって、過去に呼び出した addStockEvents の効果が取り消されます。

パラメータ

player	MediaPlayer オブジェクト
--------	--------------------

startPlayer

6.0			
-----	--	--	--

PlayerArgs オブジェクトに注釈があるかどうか、および開かれているプレーヤーをその注釈が持っているかどうかを確認します。確認できた場合は、プレーヤーの player.play を呼び出して、再生を開始または再開します。確認できなかった場合は、app.media.openPlayer を呼び出して新しい MediaPlayer オブジェクトを作成し開きます。詳しくは、[openPlayer](#) を参照してください。

注意： app.media.startPlayer は、Acrobat ユーザインターフェイスを使用してマルチメディア注釈やレンディションを作成したときの、デフォルトの「マウスボタンを放す」アクションです。カスタムの JavaScript は指定しないでください。

パラメータ

args	(オプション) PlayerArgs オブジェクト
------	---------------------------

戻り値

MediaPlayer オブジェクト。失敗した場合は null。

例

フォームボタンから画面注釈を開始します。

```
var args = {
    rendition: this.media.getRendition( "myClip" ),
    annot: this.media.getAnnot({ nPage: 0, cAnnotTitle: "myScreen" }),
};
app.media.startPlayer(args);
```

Bookmark

Bookmark オブジェクトは、「しおり」タブに表示されるツリーのノードを表します。しおりは、目的のトピックに素早く移動するための目次としてよく使用されます。

Bookmark のプロパティ

children

5.0			
-----	--	--	--

しおりツリー内でこのしおりの子になっている Bookmark オブジェクトの配列。このしおりに子がない場合、このプロパティの値は null になります。

[parent](#) プロパティおよび [bookmarkRoot](#) プロパティも参照してください。

型

配列または null

アクセス

R

例

文書内のすべてのしおりを表示します。

```
function DumpBookmark(bkm, nLevel)
{
    var s = "";
    for (var i = 0; i < nLevel; i++) s += " ";
    console.println(s + "+" + bkm.name);
    if (bkm.children != null)
        for (var i = 0; i < bkm.children.length; i++)
            DumpBookmark(bkm.children[i], nLevel + 1);
}
console.clear(); console.show();
console.println("Dumping all bookmarks in the document.");
DumpBookmark(this.bookmarkRoot, 0);
```

color

5.0	◐		☒
-----	---	--	---

しおりの色を示します。グレー、RGB または CMYK カラーで値を定義します。カラー配列の定義や、このプロパティでの値の使用方法について詳しくは、[191 ページの「カラー配列」](#) を参照してください。[style](#) プロパティも参照してください。

型

配列

アクセス

R / W (Adobe Reader : R のみ)

例

次のスクリプトは、トップレベルのしおりの色を赤色、緑色、青色と変化させます。

```
var bkm = this.bookmarkRoot.children[0];
bkm.color = color.black;
var C = new Array(1, 0, 0);
var run = app.setInterval(
    'bkm.color = ["RGB",C[0],C[1],C[2]]; C.push(C.shift());', 1000);
var stoprun=app.setTimeout(
    "app.clearInterval(run); bkm.color=color.black",12000);
```

doc

5.0			
-----	--	--	--

しおりが存在する Doc。

型

オブジェクト

アクセス

R

name

5.0	D		X
-----	---	--	---

「しおり」タブに表示するしおりのテキスト文字列。

型

文字列

アクセス

R / W (Adobe Reader : R のみ)

例

最上位にあるしおりを太字にします。

```
var bkm = this.bookmarkRoot.children[0];
console.println( "Top-level bookmark name: " + bkm.name );
```

[children](#) プロパティの例でも name プロパティを使用しています。

open

5.0	D		X
-----	----------	--	---

「しおり」タブでしおりの子を表示する（開く）か、子のサブツリーを非表示にする（閉じる）かを指定します。

型

ブーリアン

アクセス

R / W (Adobe Reader : R のみ)

parent

5.0			
-----	--	--	--

親のしおり。そのしおりがルートに該当する場合は null。[children](#) プロパティおよび [bookmarkRoot](#) プロパティも参照してください。

型

オブジェクトまたは null

アクセス

R

style

5.0	D		X
-----	----------	--	---

しおりのフォントスタイルを示します。0 は標準、1 はイタリック、2 はボールド、3 はボールドイタリックを表します。[color](#) プロパティも参照してください。

型

整数

アクセス

R / W (Adobe Reader : R のみ)

例

次のコードは、最上位にあるしおりをボールドにします。

```
var bkm = this.bookmarkRoot.children[0];  
bkm.style = 2;
```

Bookmark のメソッド

createChild

5.0	⊕		✗
-----	---	--	---

指定の位置に新しい子しおりを作成します。

[children](#) プロパティ、[insertChild](#) メソッド、[remove](#) メソッドも参照してください。

パラメータ

cName	「しおり」タブに表示するしおりの名前。
cExpr	(オプション) ユーザがしおりをクリックするたびに評価する式。これは、JavaScript アクションを持つしおりを作成することと同じです。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。デフォルトでは指定されていません。
nIndex	(オプション) 新しい子を作成する位置を指定する、子しおりの配列におけるインデックス (0 から数えます)。デフォルトは 0 です。

例

「しおり」タブの一番上に、文書の次のページに移動するしおりを作成します。

```
this.bookmarkRoot.createChild("Next Page", "this.pageNum++");
```

execute

5.0			
-----	--	--	--

しおりに関連付けられているアクションを実行します。アクションには様々なタイプがあります。主なアクションタイプのリストについては、『PDF Reference』バージョン 1.7 を参照してください。[createChild](#) メソッドも参照してください。

例

しおりの簡易検索を行います。しおりが見つかった場合は、そのしおりに関連付けられているアクションを実行します。

```
// 文書レベルまたはフォルダレベルの JavaScript
function searchBookmarks(bkm, nLevel, bkmName)
{
    if ( bkm.name == bkmName ) return bkm;
    if (bkm.children != null) {
        for (var i = 0; i < bkm.children.length; i++)
        {
            var bkMark = searchBookmarks(
                bkm.children[i], nLevel + 1, bkmName);
            if ( bkMark != null ) break;
        }
        return bkMark;
    }
    return null;
}
// この関数は定義し直して、より高度な比較を行うのが望ましい。
function bmkCompare( name1, name2 )
{
    return ( name1 == name2 );
}
```

次のコードで検索を行います。このコードは、フィールドレベルの JavaScript またはメニューアクションとして実行できます。

```
var bkmName = app.response({
    cQuestion: "Enter the name of the bookmark to find",
    cTitle: "Bookmark Search and Execute"
});
if ( bkmName != null ) {
    var bkm = searchBookmarks(this.bookmarkRoot, 0, bkmName );
    if ( bkm != null ) bkm.execute();
    else app.alert("Bookmark not found");
}
```

insertChild

5.0	D		X
-----	----------	--	---

指定のしおりを子として挿入します。指定のしおりが既にツリーに存在する場合は、一度ツリーから切り離して挿入し直します。また、挿入の循環が確認され、循環している場合は挿入されません。これにより、しおりがそれ自体の子または孫として挿入されるのを防ぐことができます。[children](#) プロパティ、[createChild](#) メソッド、[remove](#) メソッドも参照してください。

パラメータ

oBookmark	子として追加する bookmark オブジェクト。
nIndex	(オプション) 新しい子を挿入する位置を指定する、子しおりの配列におけるインデックス (0 から数えます)。デフォルトは 0 です。

例

最初の子しおりをしおりの最後に移動します。

```
var bm = bookmarkRoot.children[0];
bookmarkRoot.insertChild(bm, bookmarkRoot.children.length);
```

remove

5.0	D		X
-----	---	--	---

しおりとそのすべての子を、しおりツリーから削除します。[children](#) プロパティ、[createChild](#) メソッド、[insertChild](#) メソッドも参照してください。

例

すべてのしおりを文書から削除します。

```
bookmarkRoot.remove();
```

setAction

6.0			X
-----	--	--	---

しおりに JavaScript アクションを設定します。

Doc の [addRequirement](#) メソッドと [setPageAction](#) メソッド、Field オブジェクトの [setAction](#) メソッドも参照してください。

注意：このメソッドを実行すると、このしおりで既に定義されているアクションが上書きされます。

パラメータ

cScript	ユーザがしおりをクリックするたびに実行する JavaScript 式を定義します。
---------	---

例

一番上のしおりにアクションを割り当てます。

```
var bm = bookmarkRoot.children[0]
bm.setAction("app.beep(0);");
```

catalog

Acrobat Catalog プラグインで提供されている機能にアクセスするための静的なオブジェクト。catalog オブジェクトにアクセスするには、このプラグインがインストールされている必要があります。

注意：Catalog プラグイン（と catalog オブジェクト）を使用できるのは、Acrobat Professional のみです。

[Index](#) オブジェクト（Catalog プラグインによって提供される様々なインデックス処理を呼び出すために使用します）および [CatalogJob](#) オブジェクトも参照してください。

catalog のプロパティ

isIdle

6.0			P
-----	--	--	---

Catalog がインデックス処理の最中でない場合、`true` を返します。

型

ブーリアン

アクセス

R

jobs

6.0			P
-----	--	--	---

Catalog 処理に関する情報を取得します。Catalog には、各 Acrobat セッションにおける保留中、進行中、完了済みのジョブのリストが保持されています。CatalogJob オブジェクトの配列を返します。

型

配列

アクセス

R

catalog のメソッド

getIndex

6.0			P
-----	--	--	---

Catalog インデックスのパスを指定して index オブジェクトを取得します。返されたオブジェクトを使用して、インデックスの作成や削除などの様々なインデックス処理を実行できます。

パラメータ

cDIPath	デバイスに依存しない、Catalog インデックスのパス。
---------	-------------------------------

戻り値

Index オブジェクト。

remove

6.0			P
-----	--	--	---

指定の CatalogJob オブジェクトを Catalog のジョブリストから削除します。Catalog には、各 Acrobat セッションにおける保留中、進行中、完了済みのジョブのリストが保持されています。

パラメータ

oJob	削除する CatalogJob オブジェクト。このオブジェクトは、jobs プロパティや Index オブジェクトの様々なメソッドによって返されます。
------	---

例

保留中の処理のうち、インデックスの再構成に関するものをすべて削除します。

```
if (typeof catalog != undefined) {
    for (var i=0; i<catalog.jobs.length; i++) {
        var job = catalog.jobs[i];
        console.println("Index: ", job.path);

        if (job.status == "Pending" && job.type == "Rebuild")
            catalog.remove(job);
    }
}
```

CatalogJob

この汎用 JavaScript オブジェクトは、Catalog に送信されたジョブに関する情報を提供します。このオブジェクトは、Index オブジェクトの `build` メソッドや `catalog.jobs` プロパティから取得し、`catalog.remove` に渡します。

CatalogJob のプロパティ

path

デバイスに依存しない、ジョブの対象インデックスのパス。

型

文字列

アクセス

R

type

ジョブのインデックス処理のタイプ。有効な値は、次のとおりです。

Build
Rebuild
Delete

型

文字列

アクセス

R

status

インデックス処理のステータス。有効な値は、次のとおりです。

Pending
Processing
Completed
CompletedWithErrors

型

文字列

アクセス

R

Certificate

Certificate オブジェクトを使用すると、X.509 公開鍵証明書のプロパティに読み取り専用でアクセスできます。

関連するオブジェクトやメソッドとしては、次のものがあります。

security オブジェクト : [importFromFile](#) および [getSecurityPolicies](#)

DirConnection オブジェクト : [search](#)

Field オブジェクト : [signatureInfo](#)

FDF オブジェクト : [signatureValidate](#)

RDN オブジェクト

[Usage オブジェクト](#)

注意：このオブジェクトには、セキュリティ制限はありません。

Certificate のプロパティ

binary

5.0			
-----	--	--	--

証明書を表すバイトデータ。16進エンコードされた文字列として表されます。

型

文字列

アクセス

R

issuerDN

5.0			
-----	--	--	--

証明書の発行者の識別名。RDN オブジェクトとして返されます。

型

RDN オブジェクト

アクセス

R

keyUsage

6.0			
-----	--	--	--

証明書の鍵使用エクステンションの値を示す文字列の配列。有効な値は、次のとおりです。

kDigitalSignature	kDataEncipherment	kCRLSign
kNonRepudiation	kKeyAgreement	kEncipherOnly
kKeyEncipherment	kKeyCertSign	kDecipherOnly

型

文字列の配列

アクセス

R

MD5Hash

5.0			
-----	--	--	--

証明書の MD5 ダイジェスト。16 進エンコードされた文字列として表されます。これにより、この証明書が一意に識別されます。

型

文字列

アクセス

R

privateKeyValidityEnd

8.0			
-----	--	--	--

この証明書に関連付けられている秘密鍵の有効期間の終了日。PKUP エクステンションが存在しないか、エクステンションにこのプロパティが存在しない場合は、証明書自体の有効期間の終了日を表します。デジタル ID で署名を行うには、署名の日付が privateKeyValidityEnd の日付より前である必要があります。

型

Date オブジェクト

アクセス

R

privateKeyValidityStart

8.0			
-----	--	--	--

この証明書に関連付けられている秘密鍵の有効期間の開始日。証明書に PKUP (Private Key Usage Period) エクステンションが存在しない場合は、証明書自身の有効期間の開始日を表します。デジタル ID で署名を行うには、署名の日付が privateKeyValidityStart の日付より後である必要があります。

型

Date オブジェクト

アクセス

R

SHA1Hash

5.0			
-----	--	--	--

証明書の SHA1 ダイジェスト。16 進エンコードされた文字列として表されます。これにより、この証明書が一意に識別されます。

型

文字列

アクセス

R

serialNumber

5.0			
-----	--	--	--

この証明書の一意の識別子。issuerDN と併用されます。

型

文字列

アクセス

R

subjectCN

5.0			
-----	--	--	--

署名者の共通名。

型

文字列

アクセス

R

subjectDN

5.0			
-----	--	--	--

署名者の識別名。 RDN オブジェクトとして返されます。

型

RDN オブジェクト

アクセス

R

ubRights

7.0			
-----	--	--	--

この証明書で有効にできるアプリケーション権限。汎用の Rights オブジェクトとして返されます。

型

Rights オブジェクト

アクセス

R

Rights オブジェクト

`Rights` オブジェクトには、次のプロパティがあります。

プロパティ	型	アクセス	説明
mode	文字列	R	<p>有効な値は、次のとおりです。</p> <p>Evaluation — この文書のこの証明書で有効にされた権限は、この証明書が有効である間有効になります。</p> <p>Production — この文書のこの証明書で有効にされた権限は、永久に有効になります。</p> <p>現在、この値は Adobe の PDF ビューアでは使用されていません。</p>
rights	文字列の配列	R	<p>この証明書で有効にできるアプリケーション権限を示す文字列の配列。有効な値は、次のとおりです。</p> <p>FormFillInAndSave — フォームに入力する権限（署名フィールドを除く）と、変更されたファイルを保存する権限。</p> <p>FormImportExport — フォームデータの取り込みや書き出しを行う権限。</p> <p>FormAddDelete — フォームフィールドの追加や削除を行う権限。</p> <p>SubmitStandalone — ブラウザ内ではなく、単体の Acrobat に開いたフォームからデータを送信する権限。</p> <p>SpawnTemplate — ページテンプレートからページを生成する権限。</p> <p>Signing — 文書内の既存のフォームフィールドに署名する権限。</p> <p>AnnotModify — 注釈の作成、削除、変更を行う権限。</p> <p>AnnotImportExport — 注釈の取り込みや書き出しを行う権限。</p> <p>BarcodePlaintext — フォームフィールドの外観をプレーンテキストバーコードでエンコードする権限。</p> <p>AnnotOnline — オンラインでの注釈付けを許可します。文書内の任意の注釈をサーバにアップロードしたり、サーバから注釈をダウンロードしたりできます。それらの注釈を文書に追加することは許可されません。</p> <p>FormOnline — フォーム固有のオンラインメカニズム（SOAP や Active Data Object など）を有効にします。</p> <p>EFModify — 名前付きの埋め込みファイルの作成、削除、変更、取り込みを行う権限。ファイル添付注釈には適用されません。</p>

usage

6.0			
-----	--	--	--

この証明書を Acrobat 環境で使用する場合の用途が `Usage` オブジェクトとして返されます。

型

Usage オブジェクト

アクセス

R

Usage オブジェクト

この汎用 JavaScript オブジェクトは、`certificate.usage` プロパティの値として使用される、証明書の使用目的を表します。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
<code>endUserSigning</code>	ブーリアン	R	エンドユーザがこの証明書を使用して署名できる場合は <code>true</code> 。
<code>endUserEncryption</code>	ブーリアン	R	エンドユーザがこの証明書を使用して暗号化できる場合は <code>true</code> 。

例

現在開かれている文書を、アドレス帳のすべての人に対して暗号化します。アドレス帳に含まれているエントリのうち、署名専用証明書を持つエントリ、CA 証明書を持つエントリ、証明書を持たないエントリ、暗号化に適さないエントリは、最終的な受信者リストから除外されます。

```
var eng = security.getHandler( "Adobe.AAB" );
var dc = eng.directories[0].connect();
var recipients = dc.search();

var filteredRecipients = new Array();
for( i = 0; i < recipients.length; ++i ) {
    if( recipients[i].defaultEncryptCert &&
        recipients[i].defaultEncryptCert.usage.endUserEncryption ) {
        filteredRecipients[filteredRecipients.length] = recipients[i];
        continue;
    }
    if(recipients[i].certificates) {
        for( j = 0; j < recipients[i].certificates.length; ++j )
            if( recipients[i].certificates[j].usage.endUserEncryption ) {
                filteredRecipients[filteredRecipients.length]
                    = recipients[i];
                continue;
            }
    }
}
this.encryptForRecipients({
    oGroups: [{userEntities: filteredRecipients}]
});
```

validityEnd

7.0.5			
-------	--	--	--

証明書の有効期間の終了日。デジタル ID で署名を行うには、署名の日付が `validityEnd` の日付より前である必要があります。

型

Date オブジェクト

アクセス

R

validityStart

7.0.5			
-------	--	--	--

証明書の有効期間の開始日。デジタル ID で署名を行うには、署名の日付が `validityStart` の日付より後である必要があります。

型

Date オブジェクト

アクセス

R

Collab

この静的なオブジェクトは、コラボレーション機能を提供します。

Collab のメソッド

addStateModel

6.0			
-----	--	--	--

新しい状態モデルを Acrobat に追加します。状態モデルとは、注釈に設定可能な一連の状態のことです（注釈の状態の取得や設定について詳しくは、[Annotation](#) オブジェクトを参照）。文書レビューのワークフローを状態モデルで表せば、それを使用してレビュー管理が行えます。

[removeStateModel](#)、[getStateInModel](#)、[transitionToState](#) も参照してください。

パラメータ

cName	言語に依存しない、状態モデルの一意の識別子。
cUIName	ユーザインターフェイスに表示する状態モデルの名前。この値はローカライズされた文字列である必要があります。
oStates	状態モデルを構成する状態。states オブジェクトで表します。
cDefault	(オプション) 状態が設定されていない場合にデフォルトの状態として使用する、モデル内の状態の 1 つ。このデフォルトオプションは、デフォルトの状態がないために存在します。
bHidden	(オプション) 状態モデルのユーザインターフェイスで、状態モデルを非表示にするかどうかを指定します。デフォルトは false です（状態モデルは表示されます）。
bHistory	(オプション) 状態モデルの監査履歴を保持するかどうかを指定します。監査履歴を保持する場合、ファイルサイズが拡大します。デフォルトは true です。

States オブジェクト

この汎用オブジェクトは、状態モデルを構成する一連の状態を表し、oStates パラメータの値として利用されます。このオブジェクトリテラルでは、一意の状態識別子を要素として指定し、次のプロパティを持つオブジェクトを値として指定します。

プロパティ	説明
cUIName	状態の UI (表示名)。
oIcon	(オプション) 状態の UI に表示する icon Stream オブジェクト。

例

「ReviewStates」 という一意の名前を持つ新しい状態モデルを追加します。

```
Collab.addStateModel({
  cName: "ReviewStates",
  cUIName: "My Review",
```

```
oStates:  
{  
    "initial": {cUIName: "Haven't reviewed it"},  
    "approved": {cUIName: "I approve"},  
    "rejected": {cUIName: "Forget it"},  
    "resubmit": {cUIName: "Make some changes"}  
},  
cDefault: "initial"  
});
```

状態モデルは、Collab.removeStateModel を使用して削除できます。

documentToStream

7.0.5			S
-------	--	--	---

Doc のコピーを保存し、そのコンテンツをストリームオブジェクトとして返します。

このメソッドを呼び出しても、元の文書は変更されず、dirty プロパティは変化しません。

パラメータ

oDocument	Doc。
-----------	------

戻り値

ReadStream オブジェクト。

removeStateModel

6.0			
-----	--	--	--

addStateModel で追加した状態モデルを削除します。状態モデルを削除しても、注釈に関連付けられている状態情報は削除されません。したがって、削除したモデルを再度追加すれば、注釈のすべての状態情報を引き続き利用することができます。

[addStateModel](#)、[getStateInModel](#)、[transitionToState](#) も参照してください。

パラメータ

cName	addStateModel で状態モデルを追加したときに使用した、言語に依存しない一意の識別子。
-------	--

例

addStateModel の例に続いて、状態モデル「ReviewStates」を削除します。

```
// 状態モデルを削除  
Collab.removeStateModel("ReviewStates");
```

color

color オブジェクトは、基本色を定義する静的なオブジェクトです。プロパティやメソッドにカラー配列を渡す必要があるときは、このオブジェクトを使用します。

カラー配列

JavaScript では、色は 1、2、4 または 5 個の要素を持つ配列として表され、それぞれ透明、グレー、RGB、CMYK カラースペースに対応します。配列の最初の要素は、カラースペースのタイプを示す文字列です。それ以降の要素は、0 以上 1 以下の数値です。例えば、赤色は `["RGB", 1, 0, 0]` と表すことができます。

無効な文字列が含まれているカラー配列や、必要な要素が欠落しているカラー配列は、黒色として解釈されます。

カラースペース	文字列	追加要素の数	説明
透明	"T"	0	透明カラースペースは、色がまったくなく、そのフィールドの下にある部分が透けて見えます。
グレー	"G"	1	単一の値（無色の明度）で色が表現されます。このカラースペースでは 0 が黒色、1 が白色で、その中間値はグレーの明暗を表します。例えば、.5 は中間のグレーを表します。
RGB	"RGB"	3	色は 3 つの値で表され、それぞれ赤（Red）、緑（Green）、青（Blue）の各成分の強度を表します。RGB は、ディスプレイ装置で広く使用されています。ディスプレイ装置は、通常、赤、緑、青を組み合わせて色を表現します。
CMYK	"CMYK"	4	色は 4 つの値で表され、それぞれシアン（Cyan）、マゼンタ（Magenta）、黄（Yellow）、黒（black）の各成分の濃度を表します。これらの色は、4 色カラー印刷で使用されているインキの色で、このカラースペースはカラープリンタで広く使用されています。必要な色はシアン、マゼンタ、黄のみですが、印刷には黒も使用するのが普通です。これは、黒を使用したほうがシアン、マゼンタ、黄のインキを混ぜ合わせるよりも鮮やかな黒を表現でき、他のインキに比べて価格も低いからです。

color のプロパティ

color オブジェクトは、次の色を定義します。

Color オブジェクト	キーワード	同等の JavaScript	バージョン
透明	<code>color.transparent</code>	<code>["T"]</code>	
黒	<code>color.black</code>	<code>["G", 0]</code>	

Color オブジェクト	キーワード	同等の JavaScript	バージョン
白	color.white	["G", 1]	
赤	color.red	["RGB", 1, 0, 0]	
緑	color.green	["RGB", 0, 1, 0]	
青	color.blue	["RGB", 0, 0, 1]	
シアン	color.cyan	["CMYK", 1, 0, 0, 0]	
マゼンタ	color.magenta	["CMYK", 0, 1, 0, 0]	
イエロー	color.yellow	["CMYK", 0, 0, 1, 0]	
ダークグレー	color.dkGray	["G", 0.25]	4.0
グレー	color.gray	["G", 0.5]	4.0
ライトグレー	color.ltGray	["G", 0.75]	4.0

例

この例では、フィールドが負の値である場合はフィールドのテキストカラーを赤に、そうでない場合は黒に設定します。

```
var f = event.target; /* イベントが発生したフィールド */
f.target.textColor = event.value < 0 ? color.red : color.black;
```

color のメソッド

convert

5.0				
-----	--	--	--	--

color オブジェクトで定義されるカラースペースとカラー値を、指定のカラースペースに変換します。

- グレーのカラースペースに変換すると、カラーのテレビ信号を白黒テレビで表示するのと同じように、情報が失われることがあります。
- RGB から CMYK への変換では、黒の生成やカラー削除パラメータは考慮されません。

パラメータ

colorArray	カラー値の配列。 191 ページの「カラー配列」 を参照してください。
cColorspace	変換後のカラースペース。

戻り値

カラー配列。

例

次のコードでは、`["CMYK", 0, 1, 1, 0]` という配列が返されます。

```
color.convert(["RGB", 1, 0, 0], "CMYK");
```

equal

5.0			
-----	--	--	--

2つのカラー配列が同一かどうかを比較します。この判断のために、必要に応じて変換が行われます（例えば、`["RGB", 1, 1, 0]` と `["CMYK", 0, 0, 1, 0]` は同じです）。

パラメータ

`colorArray1` 比較する最初のカラー配列。

`colorArray2` 比較する 2 番目のカラー配列。

戻り値

配列が同じ色を表している場合は `true`、異なる場合は `false`。

例

```
var f = this.getField("foo");
if (color.equal(f.textColor, f.fillColor))
    app.alert("Foreground and background color are the same!");
```

colorConvertAction

このオブジェクトは、単一のカラー変換アクションを表します。

colorConvertAction オブジェクトは、Doc オブジェクトの [getColorConvertAction](#) メソッドから取得できます。[297 ページ](#)を参照してください。このオブジェクトを変更することで、カラー変換アクションを設定できます。カラー変換アクションは、主に「match」セクションと「action」セクションから構成されています。「match」セクションでは、このアクションの対象となるオブジェクトの種類を記述します。Doc オブジェクトの [colorConvertPage](#) メソッド ([276 ページ](#)を参照) にこのアクションの配列を渡せば、それに基づいて一致するものが検索され、アクションが実行されます。

colorConvertAction のプロパティ

action

8.0			P
-----	--	--	---

一致するものが見つかった場合に実行するアクションを記述します。

action の値にアクセスするには、colorConvertAction オブジェクトの constants.actions オブジェクトの次の定数を使用します。

constants.actions オブジェクト

名前	説明
Preserve	インキエイリアスの処理以外は何も行いません。
Convert	対象のカラースペースに変換します。
Decalibrate	キャリブレーション済みスペースをデバイスカラースペースに変換します。
DownConvert	NChannel を DeviceN にダウンコンバートします。

型

整数

アクセス

R / W

alias

8.0			P
-----	--	--	---

このアクションがインキアクションの場合、インキのエイリアスを記述します。

型

文字列

アクセス

R / W

colorantName

8.0			P
-----	--	--	---

このアクションがインキアクションの場合、色材名を記述します。

詳しくは、関連するプロパティである [isProcessColor](#) を参照してください。

型

文字列

アクセス

R / W

convertIntent

8.0			P
-----	--	--	---

オブジェクトのカラー変換に使用するレンダリングインテントを定義します。レンダリングインテントについて詳しくは、『PDF Reference』バージョン 1.7 の表 4.20 を参照してください。

convertIntent の値にアクセスするには、colorConvertAction オブジェクトの constants.renderingIntents オブジェクトを使用します。constants.renderingIntents オブジェクトのプロパティについては、[198 ページ](#)を参照してください。

型

整数

アクセス

R / W

convertProfile

8.0			P
-----	--	--	---

アクションが `Convert` の場合に、一致したオブジェクトの変換先となるカラープロファイルを記述します。使用可能なカラープロファイルのリストは、`app` オブジェクトの [printColorProfiles](#) プロパティから取得できます。

型

文字列

アクセス

R / W

embed

8.0			P
-----	--	--	---

このアクションが変換アクションであり、`embed` が `true` の場合は、変換後のオブジェクトにカラープロファイルが埋め込まれます。`embed` が `false` の場合は、オブジェクトはプロファイルで指定されたデバイスカラースペースに変換されて、保存されます。

型

ブーリアン

アクセス

R / W

isProcessColor

8.0			P
-----	--	--	---

このアクションがインキアクションであり、`isProcessColor` が `true` の場合、インキはプロセスカラーを表します。`colorantName` は、Cyan、Magenta、Yellow、Black、Red、Green、Blue のいずれかである必要があります。

型

ブーリアン

アクセス

R / W

matchAttributesAll

8.0			P
-----	--	--	---

matchAttributesAll プロパティの値は、一致の判断に使用するオブジェクト属性（フラグ）を表すビットマップです。すべてのフラグが一致する場合（完全一致の場合）に、一致したと判断されます。

フラグにアクセスするには、colorConvertAction オブジェクトの constants.objectFlags オブジェクトを使用します。各フラグの定義は次のとおりです。

constants.objectFlags オブジェクト

フラグ名	説明
ObjImage	オブジェクトが画像である。
ObjJPEG	オブジェクトが JPEG 画像である。
ObjJPEG2000	オブジェクトが JPEG2000 画像である。
ObjLossy	情報の損失があるスペースの画像である。
ObjNonLossy	情報の損失がないスペースの画像である。
ObjText	オブジェクトがテキストである。
ObjLineArt	オブジェクトがラインアート（塗りつぶしまたは線）である。
ObjShade	オブジェクトがスムーズシェードである。
ObjTransparent	オブジェクトが透明である。
ObjOverprinting	オブジェクトがオーバープリントされる。
ObjOverprintMode	OPM が 1 に設定されている。

注意： matchAttributesAll の値は、ビット演算子と constants.objectFlags のビットフラグを使用して設定できます。

型

整数

アクセス

R / W

例：

オーバープリントされる画像を対象として一致を行います。

```
var action = this.getColorConvertAction();
action.matchAttributesAll = action.constants.objectFlags.ObjImage |  
    action.constants.objectFlags.ObjOverprinting;
```

matchAttributesAny

8.0			P
-----	--	--	---

matchAttributesAny プロパティの値は、一致の判断に使用するオブジェクト属性を表すビットマップです。いずれかのフラグが一致する場合に、一致したと判断されます。

フラグにアクセスするには、colorConvertAction オブジェクトの constants.objectFlags オブジェクトを使用します。constants.objectFlags オブジェクトのプロパティについては、[197 ページ](#)を参照してください。

注意：matchAttributesAny の値は、ビット演算子と constants.objectFlags のビットフラグを使用して設定できます。

型

整数

アクセス

R / W

matchIntent

8.0			P
-----	--	--	---

オブジェクトで使用されているレンダリングインテントを対象として一致を行います。レンダリングインテントについて詳しくは、『PDF Reference』バージョン 1.7 の表 4.20 を参照してください。

matchIntent の値にアクセスするには、colorConvertAction オブジェクトの constants.renderingIntents オブジェクトを使用します。各値の定義は次のとおりです。

constants.renderingIntents オブジェクト

名前	説明
AbsoluteColorimetric	絶対的な色域を維持するレンダリングインテント。
RelativeColorimetric	相対的な色域を維持するレンダリングインテント。
Saturation	彩度レンダリングインテント。
Perceptual	知覚的レンダリングインテント。
Any	上記の任意のレンダリングインテントに一致します。
Document	現在のグラフィックステートで指定されているレンダリングインテントでレンダリングされているオブジェクトに一致します (convertIntent でのみ使用)。

型

整数

アクセス

R / W

matchSpaceTypeAll

8.0			P
-----	--	--	---

matchSpaceTypeAll プロパティの値は、一致の判断に使用するカラースペース属性（フラグ）を表すビットマップです。すべてのフラグが一致する場合（完全一致の場合）に、一致したと判断されます。

この 2 つのフィールドは、一致の判断に使用するカラースペース属性を表すビットマップです。最初のフィールドは、すべてのフラグが一致する場合（完全一致の場合）に、一致したと判断されます。2 つ目のフィールドは、いずれかのフラグが一致する場合に、一致したと判断されます。

フラグにアクセスするには、colorConvertAction オブジェクトの constants.spaceFlags オブジェクトを使用します。各フラグの定義は次のとおりです。

constants.spaceFlags オブジェクト

フラグ名	説明
DeviceSpace	カラースペースが、DeviceRGB、DeviceCMYK、DeviceGray のいずれかである。
CalibratedSpace	カラースペースが、ICCBased、CalRGB、CalGray、Lab のいずれかである。
AlternateSpace	このカラースペースが、DeviceN、Separation または ICCBased の代替スペースである。
BaseSpace	このカラースペースが、インデックススペースのベーススペースである。
IndexedSpace	このカラースペースが、インデックススペースである。
SeparationSpace	このカラースペースが、Separation カラースペースである。
DeviceNSpace	このカラースペースが、DeviceN カラースペースである。
NChannelSpace	このカラースペースが、NChannel 属性を持つ DeviceN カラースペースである。
RGBSpace	このスペースが、RGB である。このフラグは、DeviceSpace または CalibratedSpace と併用する場合にのみ使用します。
CMYKSpace	このスペースが、CMYK である。このフラグは、DeviceSpace または CalibratedSpace と併用する場合にのみ使用します。
GraySpace	このスペースが、グレースケールである。このフラグは、DeviceSpace または CalibratedSpace と併用する場合にのみ使用します。
LabSpace	このスペースが、CIELAB である。このフラグは、DeviceSpace または CalibratedSpace と併用する場合にのみ使用します。

注意： matchSpaceTypeAll の値は、ビット演算子と constants.spaceFlags のビットフラグを使用して設定できます。

型

整数

アクセス

R / W

matchSpaceTypeAny

8.0			P
-----	--	--	---

matchSpaceTypeAny プロパティの値は、一致の判断に使用するカラースペース属性（フラグ）を表すビットマップです。いずれかのフラグが一致する場合に、一致したと判断されます。

フラグにアクセスするには、colorConvertAction オブジェクトの constants.spaceFlags オブジェクトを使用します。constants.objectFlags オブジェクトのプロパティについては、[199 ページ](#)を参照してください。

注意：matchSpaceTypeAny の値は、ビット演算子と constants.spaceFlags のビットフラグを使用して設定できます。

型

整数

アクセス

R / W

preserveBlack

8.0			P
-----	--	--	---

preserveBlack が true の場合、一致したオブジェクトの変換は、黒を維持する変換を使用して実行されます。この変換では、テキストオブジェクトやラインアートオブジェクトが特別に処理されます。preserveBlack が true の場合、CMYK カラーの (0.0, 0.0, 0.0, 1.0)、RGB カラーの (0.0, 0.0, 0.0)、グレーカラーの (0.0) は特別な黒と見なされ、変換プロファイルを使用せずに、そのまま黒に変換されます。その結果、変換後の色は、プロファイルのターゲットカラースペースでの黒になります。例えば、変換プロファイルが RGB の場合、変換後の色は、カラープロファイルの指定に関係なく、(0.0, 0.0, 0.0) になります。

型

ブーリアン

アクセス

R / W

useBlackPointCompensation

8.0			P
-----	--	--	---

`useBlackPointCompensation` が `true` の場合、すべてのカラー変換で黒点の補正を使用します。このフラグの効果は、使用するカラーマネジメントメソッド（CMM）によって異なりますが、Adobe CMM を使用した場合は、ソースプロファイルのメディアの黒点に関係なく、黒がそのまま維持されるように色域変換のスケールが行われます。

型

ブーリアン

アクセス

R / W

Column

この汎用 JavaScript オブジェクトには、列内のすべての行のデータが含まれています。column オブジェクトは、statement オブジェクトの `getColumn` メソッドや `getColumnArray` メソッドによって返されます。[ColumnInfo](#) オブジェクトも参照してください。

Column のプロパティ

columnNum

列を識別する数値。

型

数値

アクセス

R

name

列の名前。

型

文字列

アクセス

R

type

列に含まれているデータの種類を表す SQL 型の 1 つ。

型

数値

アクセス

R

typeName

列のデータ型を示す名前。

型

文字列

アクセス

R

value

列のデータ値。データ型は取得されたデータの形式に準じます。

型

各種

アクセス

R / W

ColumnInfo

この汎用 JavaScript オブジェクトには、データ列の基本情報が含まれています。このオブジェクトは、`Connection` オブジェクトの `getColumnList` メソッドによって返されます。[Column](#) オブジェクトも参照してください。

ColumnInfo のプロパティ

name

列の識別名を表す文字列。この文字列は、`statement` オブジェクトの `getColumn` メソッドに渡す列の識別名として使用できます。

型

文字列

アクセス

R

description

その列に関する、データベース固有の情報を含む文字列。

型

文字列

アクセス

R

type

`ColumnInfo` オブジェクトが表す列のデータに適用される ADBC SQL 型を識別する数値。

型

数値

アクセス

R

typeName

該当する列のデータ型を識別する文字列。SQL 型（前述の `type` を参照）ではなく、データ型を表すデータベース固有の文字列です。このプロパティでは、ユーザ定義のデータ型に関して、有用な情報を得ることができます。

型

文字列

アクセス

R

Connection

5.0			X
-----	--	--	---

このオブジェクトは、データベースとのセッションをカプセル化します。Connection オブジェクトは、`ADBC.newConnection` によって返されます。[ADBC](#) オブジェクト、[Statement](#) オブジェクト、[Column](#) オブジェクト、[ColumnInfo](#) オブジェクト、[Row](#) オブジェクト、[TableInfo](#) オブジェクトも参照してください。

Connection のメソッド

close

6.0			X
-----	--	--	---

アクティブな接続を終了し、接続のために作成されたすべてのオブジェクトを無効にします。

getColumnList

5.0			X
-----	--	--	---

テーブル内の様々な列に関する情報を取得します。

パラメータ

cName 列情報を取得するテーブルの名前。

戻り値

`ColumnInfo` オブジェクトの配列。このメソッドが失敗することはありませんが、配列長が 0 になる場合があります。

例

`con` という Connection オブジェクトが既に存在しているものとして、すべての列名のリストを取得します。

```
var con = ADBC.newConnection("q32000data");
var columnInfo = con.getColumnList("sales");
console.println("Column Information");
for (var i = 0; i < columnInfo.length; i++) {
    console.println(columnInfo[i].name);
    console.println("Description: " + columnInfo[i].description);
}
```

getTableList

5.0			X
-----	--	--	---

データベース内の様々なテーブルに関する情報を取得します。

戻り値

TableInfo オブジェクトの配列を返します。このメソッドが失敗することはありませんが、配列長が 0 になる場合があります。

例

con という Connection オブジェクトが既に取得されているものとして ([getColumnList](#) および [newConnection](#) を参照)、テーブルのリストを取得します。

```
var tableInfo = con.getTableList();
console.println("A list of all tables in the database.");
for (var i = 0; i < tableInfo.length; i++) {
    console.println("Table name: " + tableInfo[i].name);
    console.println("Description: " + tableInfo[i].description);
}
```

newStatement

5.0			X
-----	--	--	---

データベース操作を実行するための Statement オブジェクトを作成します。

戻り値

成功した場合は Statement オブジェクト、失敗した場合は null。

例

Connection オブジェクトを取得して、Statement オブジェクトを取得します。

```
// Connection オブジェクトを取得
var con = ADBC.newConnection("q32000data");
// Statement オブジェクトを取得
var statement = con.createStatement();
var msg = (statement == null) ?
    "Failed to obtain newStatement!" : "newStatement Object obtained!";
console.println(msg);
```

console

console オブジェクトは、JavaScript コンソールにアクセスして JavaScript を実行したり、デバッグメッセージを表示したりするために使用する、静的なオブジェクトです。

このオブジェクトは、7.0 より前のバージョンの Adobe Reader では機能しません。バージョン 7.0 以降の Adobe Reader には、コンソールウィンドウが搭載されています。コンソールウィンドウは、主にエラーやメッセージを表示するために使用されます。この機能は、「JavaScript」環境設定の「エラーとメッセージをコンソールに表示」によって制御されます。このコンソールを対話的に使用することはできませんが、console オブジェクトのメソッドは、Acrobat Professional や Acrobat Standard と同様に機能します。

JavaScript デバッグウィンドウのデバッグ機能は、Windows 版と Macintosh 版の Adobe Reader で使用できます。Adobe Reader でデバッグを行うには、debugger.js という JavaScript ファイルをインストールして、Windows のレジストリを適切に編集する必要があります。技術的な内容について詳しくは、『Developing Acrobat Applications Using JavaScript』を参照してください。

[dbg](#) オブジェクトも参照してください。

console のメソッド

clear

3.01			
------	--	--	--

コンソールウィンドウのバッファからすべての出力をクリアします。

hide

4.0			
-----	--	--	--

コンソールウィンドウを閉じます。

println

3.01			
------	--	--	--

文字列の末尾に改行を付加してコンソールウィンドウに表示します。

パラメータ

cMessage 表示する文字列メッセージ。

例 1

この例では、フィールドの値をコンソールウィンドウに表示します。スクリプトは、「マウスボタンを放す」イベントで実行できます。

```
var f = this.getField("myText");
console.clear(); console.show();
console.println("Field value = " + f.value);
```

例 2

コンソールをデバッグツールとして使用できます。例えば、変数の値をコンソールに出力できます。次のスクリプトを文書レベルで実行します。

```
var debugIsOn = true;
function myFunction ( n, m )
{
    if (debugIsOn)
    {
        console.println("Entering function: myFunction");
        console.println(" Parameter 1: n = " + n);
        console.println(" Parameter 2: m = " + m);
    }
    ....
    ....
    if (debugIsOn) console.println(" Return value: rtn = " + rtn);
    return rtn;
}
```

Acrobat 6.0 以降、JavaScript デバッガを使用してデバッグできるようになりました。[dbg](#) オブジェクトを参照してください。

show

3.01			
------	--	--	--

コンソールウィンドウを表示します。

例

コンソールウィンドウをクリアして、表示します。

```
console.clear();
console.show();
```

Data

5.0			
-----	--	--	--

Data オブジェクトは、文書内に保持されている埋め込みファイルやデータストリームを表します。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

Data オブジェクトは、文書に関連するソースファイルやメタデータなどのデータを、文書に関連付けて埋め込むのに適しています。Data オブジェクトは、外部ファイルシステムから挿入、照会、抽出することができます。

Doc の次のプロパティやメソッドを参照してください。

[createDataObject](#)、[dataObjects](#)、[exportDataObject](#)、[getDataObject](#)、[importDataObject](#)、
[removeDataObject](#)、[openDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#)

注意：Data オブジェクトのメソッドは、Acrobat 5.0 で実装されました。ただし、追加の使用権限を付与して Adobe Reader でそれらのメソッドを使用可能にする機能は、Adobe Reader 6.0 で導入されました。

Data のプロパティ

creationDate

埋め込まれたファイルの作成日。

型

日付

アクセス

R

description

7.0.5			
-------	--	--	--

この data オブジェクトの説明。

型

文字列

アクセス

R / W

MIMETYPE

この data オブジェクトの MIME タイプ。

型

文字列

アクセス

R

modDate

埋め込まれたファイルの更新日。

型

日付

アクセス

R

name

この data オブジェクトの名前。

型

文字列

アクセス

R

例

この文書の添付ファイルの名前を表示します。

```
console.println("Dumping all data objects in the document.");
var d = this.dataObjects;
for (var i = 0; i < d.length; i++)
    console.println("DataObject[" + i + "]=" + d[i].name);
```

path

デバイスに依存しない、埋め込まれたファイルのパス。

型

文字列

アクセス

R

size

圧縮されていない data オブジェクトのサイズ（バイト単位）。

型

数値

アクセス

R

DataSourceInfo

この汎用 JavaScript オブジェクトには、データベースの基本情報が含まれています。このオブジェクトの配列は、ADBC.getDataSourceList メソッドによって返されます。

DataSourceInfo のプロパティ

name

データベースの識別名を表す文字列。この文字列を newConnection に渡して、DataSourceInfo オブジェクトが表すデータベースへの接続を確立することができます。

型

文字列

アクセス

R

description

そのデータベースに関する、データベース固有の情報を含む文字列。

型

文字列

アクセス

R

dbg

dbg オブジェクトは、コマンドラインコンソールから JavaScript デバッガを制御するときに使用できる静的なオブジェクトです。このオブジェクトのメソッドを使用すれば、JavaScript デバッガダイアログボックスのツールバーボタンと同じ操作が行えます。また、dbg オブジェクトを使用して、ブレークポイントを作成、削除、確認することもできます。

dbg オブジェクトと JavaScript デバッガは、Acrobat Professional でのみ使用できます。

注意： デバッグセッション中にビューアがロックされた場合は、Esc キーを押すことによって問題が解決される場合があります。

モーダルダイアログボックスが開いているときにデバッグを行うことはできません。この状況は、例えば、バッチシーケンスをデバッグするときに発生します。

app.setInterval や app.setTimeout でイベント処理を実行しているときにスクリプトをデバッグすると、警告ボックスが繰り返し表示される場合があります。この問題を解決するには、モーダルダイアログボックスを終了してから Esc キーを押してください。

(Acrobat 7.0) デバッガが開いており、デバッグセッションが進行中である間は、Acrobat アプリケーションは使用できません。

dbg のプロパティ

bps

6.0			P
-----	--	--	---

デバッガで設定されたブレークポイントに対応するブレークポイント汎用オブジェクトの配列。このオブジェクトのプロパティとメソッドは、次のとおりです。

プロパティ	型	アクセス	説明
fileName	文字列	R	デバッガ内のスクリプトを識別する文字列。
condition	文字列	R	ブレークポイントで停止するかどうかを決定するためにデバッガで評価する JavaScript 式。条件付きブレークポイントを作成するために使用します。このプロパティのデフォルト値は、「true」という文字列です。
lineNum	数値	R	ブレークポイントが設定されている、スクリプト内の行の番号。

メソッド	パラメータ	戻り値	説明
toString	なし	文字列	ブレークポイントを表す文字列。

型

配列

アクセス

R

例

現在アクティブなすべてのブレークポイントを示します。

```
var db = dbg.bps
for ( var i = 0; i < db.length; i++ )
{
    for ( var o in db[i] ) console.println(o + ":" + db[i][o]);
    console.println("-----");
}
```

別の使用例については、[sb](#) を参照してください。

dbg のメソッド

c

6.0			P
-----	--	--	---

c (continue) メソッドは、デバッガで停止されたプログラムの実行を再開します。再開された JavaScript プログラムは、ブレークポイントが設定された場所で再度停止するか、または最後まで実行されます。

cb

6.0	D		P
-----	---	--	---

cb (clear breakpoint) メソッドは、デバッガ内のブレークポイントをクリアします。dbg.cb メソッドによって文書が変更済みと認識されるのは、ブレークポイントを文書に保存するようにユーザ環境設定（編集／環境設定／JavaScript）で設定している場合のみです。

パラメータ

fileName ブレークポイントを削除するスクリプトの名前。

lineNum スクリプト内でクリアするブレークポイントの行番号。

q

6.0			P
-----	--	--	---

q (quit) メソッドは、現在の JavaScript のデバッグと実行を終了します。また、デバッガダイアログボックスも終了します。

sb

6.0	D		P
-----	---	--	---

`sb` (`set breakpoint`) メソッドは、デバッガ内に新しいブレークポイントを設定します。`dbg.sb` メソッドによって文書が変更済みと認識されるのは、ブレークポイントを文書に保存するようにユーザ環境設定（編集／環境設定／JavaScript）で設定している場合のみです。

パラメータ

<code>fileName</code>	ブレークポイントを設定するスクリプトの名前。
<code>lineNum</code>	ブレークポイントを作成するスクリプト内の行の番号。
<code>condition</code>	(オプション) デバッガがブレークポイントに到達するたびに評価する JavaScript 式。式の評価が <code>true</code> になった場合は、そのブレークポイントでデバッガが停止します。式の評価が <code>false</code> になった場合は、スクリプトの実行が継続され、ブレークポイントで停止しません。デフォルト値は <code>true</code> です。

例 1

スクリプトが実行され、エラーによって例外が発生したとします。エラーメッセージの情報を使用して、ブレークポイントをプログラムによって設定できます。

```
SyntaxError: missing ; before statement 213:Document-Level: myDLJS
// コンソールを使用してブレークポイントを設定
dbg.sb({
  fileName: "Document-Level: myDLJS",
  lineNum: 213,
  condition: "true"
});
```

例 2

「JavaScript」ユーザ環境設定の「ブレークポイントを PDF ファイルに保存」チェックボックスの機能をシミュレートします。

```
// PDF ファイルにブレークポイントを保存
this.addScript("myBreakpoints", "var myBPS = " + dbg.bps.toSource());

// ブレークポイントをリセット
for ( var i = 0; i < myBPS.length; i++ ) dbg.sb( myBPS[i] );
```

例 3

条件付きブレークを設定します。次のコードは、「マウスボタンを放す」アクションです。

```
for (var i=0; i<100; i++)
    myFunction(i);                                // 文書レベルで定義

// コンソールで、条件付きブレークを設定。ここでは、
// ループのインデックスが 30 を超えたらブレーク。
dbg.sb({
    fileName:"AcroForm:Button1:Annot1:MouseUp:Action1",
    lineNum:2,
    condition:"i > 30"
})
```

si

6.0			P
-----	--	--	---

si (step in) メソッドは、JavaScript プログラム内の次の命令にプログラムポインタを進めます。スクリプトで定義されている関数呼び出しの個所では、その関数内に進みます（ネイティブの JavaScript 関数にはステップインできません）。

sn

6.0			P
-----	--	--	---

sn (step instruction) メソッドは、JavaScript プログラム内の次のバイトコードにプログラムポインタを進めます（JavaScript の命令は、JavaScript インタプリタによって定義されたいくつかのバイトコードで構成されています）。

so

6.0			P
-----	--	--	---

so (step out) メソッドは、現在の関数が終了するまでプログラムを実行し、その関数の呼び出し元の直後の命令で実行を停止します。現在デバッグしているスコープが最上位のスコープである場合、プログラムは終了するまで継続して実行されるか、または次のブレークポイントで再度停止します。

sv

6.0			P
-----	--	--	---

sv (step over) メソッドは、JavaScript プログラム内の次の命令にプログラムポインタを進めます。関数呼び出しが検出された場合でも、デバッガは関数内に定義された命令にステップインしません。

Dialog

このオブジェクトのインスタンスは、ダイアログボックスハンドラにパラメータとして渡します ([108 ページの「ダイアログボックスハンドラ」を参照](#))。app.execDialog に渡すダイアログボックス記述子オブジェクトリテラルでは、initialize、validate、commit、destroy、ItemID というハンドラ（メソッド）を定義できます。Dialog オブジェクトを使用すれば、ダイアログボックスの現在の設定を取得したり、設定を変更したりできます。

Dialog のメソッド

enable

7.0			
-----	--	--	--

オブジェクトリテラルを渡して、様々なダイアログボックス要素を使用可能または使用不可にします。

通常、enable は、app.execDialog に渡すオブジェクトリテラルの initialize メソッド ([108 ページの「ダイアログボックスハンドラ」を参照](#)) で呼び出して、様々なダイアログボックス要素を使用可能にするかどうかを初期設定します。

パラメータ

オブジェクトリテラル 変更するダイアログボックスアイテムごとに、オブジェクトリテラルのエントリ（ラベルである ItemID と、使用可能かどうかを示すブーリアン値の組）が 1 つ必要です。

例

app.execDialog の例を参照してください。

end

7.0			
-----	--	--	--

現在実行中のダイアログボックスを終了します（「キャンセル」が押された場合と同じ処理を行います）。このメソッドは、オプションのパラメータとして、ダイアログボックス要素の ItemID（文字列）を取ります。これは、ダイアログボックスを閉じるために使用された要素として通知されます。この ItemID が、ダイアログボックスを作成した app.execDialog メソッドの戻り値になります。

パラメータ

文字列 (オプション) ダイアログボックス要素の ItemID。これは、ダイアログボックスを閉じるために使用された要素として通知されます。

例

app.execDialog の例を参照してください。

load

7.0			
-----	--	--	--

渡されたオブジェクトリテラルに基づいて、ダイアログボックス要素の値を設定します。ダイアログボックスアイテムは、一意の 4 文字の文字列である ItemID によって識別されます。

通常、`load` は、`app.execDialog` に渡すオブジェクトリテラルの `initialize` メソッド ([108 ページの「ダイアログボックスハンドラ」](#) を参照) で呼び出して、様々なダイアログボックス要素の値を初期設定します。

パラメータ

オブジェクトリテラル	変更するダイアログボックスアイテムごとに、オブジェクトリテラルのエントリ (ラベルである ItemID と、コンテンツであるダイアログボックス要素の設定の組) が 1 つ必要です。複数の値を持つダイアログボックス要素 (<code>list_box</code> や <code>popup</code> など) では、ラベルである表示エントリとコンテンツである数値の組が複数指定されたオブジェクトリテラルが値として使用されます。同様に、階層構造を持つダイアログボックス要素 (<code>hier_list_box</code> など) では、一連のネストしたオブジェクトリテラルが値として使用されます。数値が 0 より大きい場合はその項目が選択され、それ以外の場合は選択されません。
------------	---

例

`app.execDialog` の例を参照してください。

store

7.0			
-----	--	--	--

ダイアログボックス要素の値を取得して、1 つのオブジェクトリテラルとして返します。ダイアログボックスアイテムは、一意の 4 文字の文字列である ItemID によって識別されます。ダイアログボックスアイテムごとに、オブジェクトリテラルのエントリ (ラベルである ItemID と、コンテンツであるダイアログボックス要素の設定の組) が 1 つ存在します。複数の値を持つダイアログボックス要素 (`list_box` や `popup` など) では、ラベルである表示エントリとコンテンツである数値の組が複数指定されたオブジェクトリテラルが値として使用されます。数値が 0 より大きい場合はその項目が選択されていたことを表し、それ以外の場合は選択されていなかったことを表します。

通常、`store` は、`app.execDialog` に渡すオブジェクトリテラルの `commit` メソッド ([108 ページの「ダイアログボックスハンドラ」](#) を参照) で呼び出して、様々なダイアログボックス要素の値を抽出します。

戻り値

オブジェクトリテラル

DirConnection

6.0			
-----	--	--	--

このオブジェクトは、ディレクトリへの接続を表します。ディレクトリとは、公開鍵証明書などのユーザ情報が格納されているレポジトリのことです。ディレクトリ接続は、Directory オブジェクトの connect メソッドを使用して開かれます。特定の名前のディレクトリで、複数の接続が同時に開いている場合があります。特に明記されている場合を除き、すべての DirConnection オブジェクトで、ここに示されているすべてのプロパティとメソッドがサポートされている必要があります。

注意：このオブジェクトは、Directory オブジェクトからのみ取得でき、Directory オブジェクトのセキュリティ制限を受けます。したがって、DirConnection オブジェクトを使用できるのは、バッチイベント、コンソールイベント、アプリケーション初期化イベントのみです（Adobe Reader も含む）。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

DirConnection のプロパティ

canList

6.0			
-----	--	--	--

ディレクトリ接続がすべてのエントリをリストできるかどうかを示します。ディレクトリによっては、実際にこの操作を行うにはエントリ数が多すぎる場合があります。

型

ブーリアン

アクセス

R

例

AAB ディレクトリでは、ローカルの信頼済み証明書の一覧をリストすることができます。

```
var sh = security.getHandler( "Adobe.AAB" );
var dc = sh.directories[0].connect();
console.println( "CanList = " + dc.canList );
```

canDoCustomSearch

6.0			
-----	--	--	--

ディレクトリ固有の検索パラメータ属性を使用した検索がディレクトリ接続でサポートされているかどうかを示します。例えば、LDAP ディレクトリ固有の属性には、o (organization)、c (country)、cn (common name)、givenname、sn (surname)、uid、st、postalcode、mail、telephonenumber などがあります。

型

ブーリアン

アクセス

R

canDoCustomUISearch

6.0		⌚	
-----	--	---	--

検索パラメータを収集する独自のカスタムユーザインターフェイスを使用した検索がディレクトリ接続でサポートされているかどうかを示します。

型

ブーリアン

アクセス

R

canDoStandardSearch

6.0		⌚	
-----	--	---	--

標準の検索パラメータ属性を使用した検索がディレクトリ接続でサポートされているかどうかを示します。標準属性は、次のとおりです。

```
firstName  
lastName  
fullName  
email  
certificates
```

一部のディレクトリデータベースの実装では、これらの属性がサポートされていない場合がありますが、ディレクトリハンドラはこれらの属性をディレクトリで解釈可能な名前に自由に変換できます。

型

ブーリアン

アクセス

R

groups

6.0			
-----	--	--	--

言語に依存する、この接続で使用可能なグループ名の配列。

型

配列

アクセス

R

name

6.0			
-----	--	--	--

言語に依存しない、このオブジェクトが接続しているディレクトリ名。例えば、`Adobe.PPKMS.ADSI.dir0` のようになります。すべての `DirConnection` オブジェクトで、このプロパティがサポートされている必要があります。

型

文字列

アクセス

R

uiName

6.0			
-----	--	--	--

言語に依存する、このオブジェクトが接続しているディレクトリを表す文字列。この文字列は、ユーザインターフェイスでの使用に適しています。すべての `DirConnection` オブジェクトで、このプロパティがサポートされている必要があります。

型

文字列

アクセス

R

DirConnection のメソッド

search

6.0			
-----	--	---	--

ディレクトリを検索し、検索パラメータに一致する `UserEntity` オブジェクトの配列を返します。
`UserEntity` オブジェクトは、`setOutputFields` メソッドで要求されたすべての属性のプロパティを含む汎用オブジェクトです。検索の前に `setOutputFields` メソッドが呼び出されていない場合は、エントリを含まない `UserEntity` オブジェクトを返します。

パラメータ

<code>oParams</code>	(オプション) 検索属性名と対応する文字列で構成されたキーと値のペアの配列を含む汎用オブジェクト。 <code>oParams</code> が指定されておらず、このディレクトリの <code>canList</code> が <code>true</code> の場合は、ディレクトリ内のすべてのエントリを返します。 <code>oParams</code> が指定されておらず、 <code>canList</code> が <code>false</code> の場合は、例外が発生します。
<code>cGroupName</code>	(オプション) グループの名前 (Group オブジェクトではありません)。これを指定した場合、検索はこのグループに限定されます。
<code>bCustom</code>	(オプション) <code>false</code> (デフォルト) の場合、 <code>oParams</code> には標準検索属性が含まれます。 <code>true</code> の場合、 <code>oParams</code> にはディレクトリ固有の検索パラメータが含まれます。 <code>canDoCustomSearch</code> プロパティが <code>true</code> でない場合は、例外が発生します。
<code>bUI</code>	(オプション) <code>true</code> の場合、ハンドラは検索パラメータを収集するためのユーザインターフェイスを表示します。検索結果はこのメソッドから返されます。 <code>bUI</code> が <code>true</code> の場合、 <code>canDoCustomUISearch</code> も <code>true</code> である必要があります。 <code>false</code> の場合は例外が発生します。 <code>bUI</code> を指定する場合は、 <code>bCustom</code> も指定する必要があります (ただし、この値は無視されます)。

戻り値

`UserEntity` オブジェクトの配列。

例 1

ディレクトリ検索を行います。

```
var sh = security.getHandler( "Adobe.PPKMS" );
var dc= sh.directories[0].connect();
dc.setOutputFields( {oFields: ["certificates", "email"]} );
var retVal = dc.search({oParams:{lastName:"Smith"}});
if( retVal.length )
    console.println( retVal[0].email );
```

例 2

ローカルの Acrobat Address Book 内のエントリをすべてリストします。このスクリプトでは、ディレクトリを検索して、各ユーザとその認証情報を返します。

```
var sh = security.getHandler( "Adobe.AAB" );
var dc = sh.directories[0].connect();
if( dc.canList ) {
    var x = dc.search();
    for( j=0; j<x.length; ++j ) {
        console.println("Entry[" + j + "] = " + x[j].fullName + ":");
        for(i in x[j]) console.println("  " + i + " = " + x[j][i]);
    }
}
```

UserEntity オブジェクト

ディレクトリ内のユーザとユーザに関連付けられた証明書を示す汎用 JavaScript オブジェクト。すべてのディレクトリハンドラに対して特定の意味を持つ標準プロパティが含まれています。ディレクトリハンドラは、必要に応じて、これらのエントリをディレクトリハンドラ固有のエントリに変換します。このオブジェクトの配列は、DirConnection オブジェクトの search メソッドによって返されます。

このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
firstName	文字列	R / W	ユーザの名。
lastName	文字列	R / W	ユーザの姓。
fullName	文字列	R / W	ユーザの姓名。
certificates	Certificate オブジェクトの 配列	R / W	このユーザに属している証明書の配列。特定の用途に使用する証明書を検索する場合は、証明書の keyUsage プロパティを確認する必要があります。
defaultEncryptCert	Certificate オブジェクトの 配列	R / W	このユーザエンティティの文書を暗号化するときに優先して使用する証明書。ユーザエンティティオブジェクトを処理する関数では、暗号化証明書を選択するときに最初にこのプロパティが確認されます。このプロパティが設定されていない場合は、certificates プロパティで最初に一致した有効な証明書が使用されます。

setOutputFields

6.0		⌚	X
-----	--	---	---

search メソッドを実行したときに返される属性のリストを定義します。

注意：このメソッドは、Adobe.AAB ディレクトリハンドラではサポートされていません。カスタムオプションは、Adobe.PPKMS.ADSI ディレクトリハンドラではサポートされていません。

パラメータ

oFields	検索メソッドを呼び出したときにディレクトリから返す必要がある属性の名前を表す文字列の配列。この配列には、すべてのディレクトリハンドラで使用できる標準属性の名前か、特定のディレクトリに対して定義されているカスタム属性の名前を指定する必要があります。標準属性とは、UserEntity オブジェクトに対して定義されているプロパティ名のことです。ディレクトリハンドラは、必要に応じて、標準属性名を解釈可能な名前に変換できます。
bCustom	(オプション) oFields 内の名前が標準属性名であることを示すブーリアン値。true の場合、名前は特定のディレクトリハンドラに対して定義されているディレクトリ固有の属性を表します。デフォルトは false です。

戻り値

oFields で指定した属性のうち、このディレクトリでサポートされていない属性の名前を表す文字列の配列。oFields 配列が空の場合は、空の配列を返します。

例

この例では、dc.setOutputFields は ["x", "y"] という文字列の配列を返します。

```
var sh = security.getHandler("Adobe.PPKMS");
var dc = sh.directories[0].connect();
var w = dc.setOutputFields( [ "certificates", "email", "x", "y" ] );
console.println( w );
```

DirConnection.[search](#) メソッドの例も参照してください。

Directory

6.0			
-----	--	---	--

ディレクトリとは、公開鍵証明書などのユーザ情報が格納されているレポジトリのことです。ディレクトリには Directory オブジェクトを使用してアクセスできます。Directory オブジェクトは、SecurityHandler オブジェクトの `directories` プロパティまたは `newDirectory` メソッドを使用して取得できます。

Acrobat 6.0 以降では複数のディレクトリが提供されています。Adobe.AAB セキュリティハンドラには、Adobe.AAB.AAB という単一のディレクトリがあります。このディレクトリを使用すれば、ローカルの Acrobat アドレス帳（信頼済み証明書ストアとも呼ばれます）にアクセスできます。Windows では、Adobe.PPKMS セキュリティハンドラを使用して、ユーザが作成した任意の数のディレクトリに Microsoft Active Directory Script Interface (ADSI) を介してアクセスできます。これらのディレクトリは、`Adobe.PPKMS.ADSI.dir0`、`Adobe.PPKMS.ADSI.dir1` などの名前で順に作成されます。

注意：このオブジェクトは、SecurityHandler オブジェクトからのみ取得でき、SecurityHandler オブジェクトのセキュリティ制限を受けます。したがって、Directory オブジェクトを使用できるのは、バッチイベント、コンソールイベント、アプリケーション初期化イベントのみです (Adobe Reader も含む)。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。

Directory のプロパティ

info

6.0			
-----	--	---	--

このプロパティの値は、`DirectoryInformation` オブジェクトです。これは、`Directory` オブジェクトのプロパティを設定および取得するために使用する汎用オブジェクトです。

型

オブジェクト

アクセス

R / W

例

```
// 新しいディレクトリを作成してアクティビ化
var oDirInfo = { dirStdEntryID: "dir0",
    dirStdEntryName: "Employee LDAP Directory",
    dirStdEntryPrefDirHandlerID: "Adobe.PPKMS.ADSI",
    dirStdEntryDirType: "LDAP",
    server: "ldap0.example.com",
    port: 389 };
var sh = security.getHandler( "Adobe.PPKMS" );
var newDir = sh.newDirectory();
newDir.info = oDirInfo;
```

DirectoryInformation オブジェクト

directory information オブジェクトは、ディレクトリのプロパティを表す汎用オブジェクトで、次の標準プロパティがあります。

プロパティ	型	アクセス	必須	説明
dirStdEntryID	文字列	R / W	○	言語に依存しない、一意のディレクトリ名。英数字、アンダースコア、ピリオド、ハイフンを使用できます。新しい directory オブジェクトには、ID を指定しないことをお勧めします。この場合、新しい一意の名前が自動的に生成されます。
dirStdEntryName	文字列	R / W	○	ユーザにわかりやすいディレクトリ名。
dirStdEntryPrefDirHandlerID	文字列	R / W	×	このディレクトリで使用されるディレクトリハンドラの名前。セキュリティハンドラでは、複数のディレクトリタイプ（ローカルディレクトリや LDAP ディレクトリなど）に対応する複数のディレクトリハンドラをサポートできます。
dirStdEntryDirType	文字列	R / W	×	ディレクトリのタイプ。LDAP、ADSI、WINNT などがあります。
dirStdEntryVersion	文字列	R	×	データのバージョン。ディレクトリによって設定されない場合のデフォルト値は 0 です。Adobe.AAB ディレクトリハンドラと Adobe.PPKMS.ADSI ディレクトリハンドラの Acrobat 6.0 ディレクトリの値は、0x00010000 です。

Directory information オブジェクトには、ディレクトリハンドラ固有の追加プロパティが含まれていることがあります。Adobe.PPKMS.ADSI ディレクトリハンドラには、次の追加プロパティが含まれています。

プロパティ	型	アクセス	説明
server	文字列	R / W	データを保持するサーバ。例えば、addresses.employees.example.com などです。
port	数値	R / W	サーバのポート番号。標準の LDAP ポート番号は 389 です。

プロパティ	型	アクセス	説明
searchBase	文字列	R / W	検索範囲をディレクトリの特定の部分に限定します。 例えば、 o=XYZ Systems,c=US のように指定します。
maxNumEntries	数値	R / W	1回の検索で取得するエントリの最大数。
timeout	数値	R / W	検索に使用する時間の最大値。

例 1

新しいディレクトリを作成して、アクティブ化します。

```
var oDirInfo = { dirStdEntryID: "dir0",
    dirStdEntryName: "Employee LDAP Directory",
    dirStdEntryPrefDirHandlerID: "Adobe.PPKMS.ADSI",
    dirStdEntryDirType: "LDAP",
    server: "ldap0.example.com",
    port: 389
};
var sh = security.getHandler( "Adobe.PPKMS" );
var newDir = sh.newDirectory();
newDir.info = oDirInfo;
```

例 2

既存のディレクトリの情報を取得します。

```
var sh = security.getHandler("Adobe.PPKMS");
var dir0 = sh.directories[0];
// 効率化のために directory info オブジェクトを 1 回取得
var dir0Info = dir0.info;
console.println( "Directory " + dir0Info.dirStdEntryName );
console.println( "address " + dir0Info.server + ":" + dir0Info.port );
```

Directory のメソッド

connect

6.0			
-----	--	--	--

指定の名前を持つディレクトリへの接続を表す `DirConnection` オブジェクトを返します。1つのディレクトリに複数のアクティブな接続がある場合があります。

`DirConnection` オブジェクトおよび `SecurityHandler` オブジェクトの `directories` プロパティも参照してください。

パラメータ

oParams	(オプション) 接続を確立するために必要なパラメータを含む汎用オブジェクト。このオブジェクトのプロパティは、ディレクトリハンドラに依存します。userid と password を含めることができます。
bUI	(オプション) ブーリアン値。デフォルトは <code>false</code> です。接続の確立に UI が必要な場合に、ディレクトリハンドラで UI を表示できるかどうかを指定します。

戻り値

`DirConnection` オブジェクト。指定の名前のディレクトリがない場合は `null`。

例

使用可能なディレクトリを列举し、接続します。

```
var sh = security.getHandler( "Adobe.PPKMS" );
var dirList = sh.directories;
for ( var i=0; i< dirList.length; i++)
    for ( var o in dirList[i].info )
        console.println( o + " = " + dirList[i].info[o] );
var dirConnection = dirList[0].connect();
```

Doc

このオブジェクトは、ビューアで開いている PDF 文書と JavaScript インタプリタとの間を取り持つインターフェイスです。このオブジェクトには、PDF 文書にアクセスするためのメソッドやプロパティが用意されています。

JavaScript で Doc オブジェクトにアクセスするには、様々な方法があります。

- `this` オブジェクトは、通常、基盤となる文書の Doc オブジェクトを表します。
- `extractPages`、`app.activeDocs`、`app.openDoc` などのいくつかのプロパティやメソッドで、Doc オブジェクトが返されます。
- 多くの場合、`event` オブジェクトを使用して Doc オブジェクトにアクセスできます。このオブジェクトは、JavaScript を実行するイベントによって作成されます。
 - `mouse`、`focus`、`blur`、`calculate`、`validate`、`format` の各イベントでは、`event.target` によって、イベントを発生させた Field オブジェクトを取得できます。その Field オブジェクトの `doc` メソッドを使用すれば、Doc オブジェクトにアクセスできます。
 - それ以外のイベントでは、`event.target` によって Doc オブジェクトを取得できます。

例 1：this オブジェクトを介したアクセス

`this` オブジェクトを使用して、この文書のページ数を取得します。

```
var nPages = this.numPages;  
// 「this」文書のトリミング領域を取得  
var aCrop = this.getPageBox();
```

例 2：戻り値を使用したアクセス

ある文書からの戻り値を使用して、別の文書を開いたり、閉じたり、変更したり、保存したりします。

```
// 「this」文書からの相対パス  
var myDoc = app.openDoc("myNovel.pdf", this);  
myDoc.info.Title = "My Great Novel";  
myDoc.saveAs(myDoc.path);  
myDoc.closeDoc(true);
```

例 3：event オブジェクトを介したアクセス

`mouse`、`calculate`、`validate`、`format`、`focus`、`blur` の各イベントの場合

```
var myDoc = event.target.doc;
```

それ以外のイベントの場合（バッチャイベントやコンソールイベントなど）

```
var myDoc = event.target;
```

Doc のプロパティ

alternatePresentations

6.0			
-----	--	--	--

文書の `AlternatePresentation` オブジェクトを参照します。代替プレゼンテーションの表示に必要な機能が使用できない場合、このプロパティは `undefined` になります。

`AlternatePresentation` オブジェクトを使用すると、文書の代替プレゼンテーションにアクセスできます。PDF 言語の拡張機能では、文書に多数の名前付き代替プレゼンテーションを含めることが可能です。既知の `type` の代替プレゼンテーションが文書に含まれている場合、その文書の `alternatePresentations` プロパティは、それぞれの代替プレゼンテーションに対応するプロパティを持ちます。それらのプロパティは、代替プレゼンテーションと同じ名前を持ち、代替プレゼンテーションの `AlternatePresentation` オブジェクトを参照しています。文書内に認識可能な代替プレゼンテーションがない場合、このオブジェクトは空です（プロパティを持ちません）。

代替プレゼンテーションについて詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

注意：現在の実装との互換性を保つために、代替プレゼンテーション名は ASCII 文字列である必要があります。現在実装されている代替プレゼンテーションのタイプは「SlideShow」のみです。

代替プレゼンテーションの制御に使用できるプロパティやメソッドについて詳しくは、[AlternatePresentation](#) を参照してください。

型

オブジェクトまたは `undefined`

アクセス

R

例 1

`AlternatePresentation` オブジェクトが存在するかどうかをテストします。

```
if( typeof this.alternatePresentations != "undefined" )
{
    // AlternatePresentations が存在している
    // 文書内の代替プレゼンテーションの名前をすべてリスト
    for ( var ap in this.alternatePresentations ) console.println(ap);
}
```

例 2

「MySlideShow」という名前のスライドショーを取得して、スライドショーを開始します。

```
// oMySlideShow は AlternatePresentation オブジェクト
oMySlideShow = this.alternatePresentations["MySlideShow"];
oMySlideShow.start();
```

author

X	D		
---	---	--	--

注意：このプロパティの代わりに `info` プロパティを使用してください。

文書の作成者。

型

文字列

アクセス

R / W (Adobe Reader : Rのみ)

baseURL

5.0	D		
-----	---	--	--

文書のベース URL は、文書内の相対 Web リンクの解決に使用されます。[URL](#) も参照してください。

型

文字列

アクセス

R / W

例

ベース URL を設定し、そのベース URL からの相対位置で指定したページにアクセスするリンクを作成します。

```
console.println("Base URL was " + this.baseURL);
this.baseURL = "http://www.adobe.com/products/";
console.println("Base URL is " + this.baseURL);
// 最初のページにリンクを追加
var link = this.addLink(0, [200,200, 400, 300])
// Adobe Web サイトの Acrobat ページにアクセスするアクションを設定。
link.setAction("this.getURL('acrobat',false)")
```

bookmarkRoot

5.0			
-----	--	--	--

しおりツリーのルートしおり。このしおりはユーザには表示されません。これは、プログラムでツリーや子しおりにアクセスする際に使用されます。

型

オブジェクト

アクセス

R

例

例については、[Bookmark](#) を参照してください。

calculate

4.0			
-----	--	--	--

true (デフォルト値) の場合、この文書で計算を実行できます。false の場合、この文書で計算は実行できません。app.calculate プロパティの使用は推奨されません。代わりにこのプロパティを使用してください。

型

ブーリアン

アクセス

R / W

creationDate

X			
---	--	--	--

注意：このプロパティの代わりに info プロパティを使用してください。

文書の作成日。

型

日付

アクセス

R

creator

X			
---	--	--	--

注意：このプロパティの代わりに `info` プロパティを使用してください。

文書を作成したソフトウェア（「Adobe FrameMaker」や「Adobe PageMaker」など）。

型

文字列

アクセス

R

dataObjects

5.0			
-----	--	--	--

文書内の名前付き Data オブジェクトがすべて含まれた配列。

関連するプロパティやメソッドとしては、[openDataObject](#)、[getDataObject](#)、[createDataObject](#)、[importDataObject](#)、[removeDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#) があります。

型

配列

アクセス

R

例

文書内のすべての埋め込みファイルをリストします。

```
var d = this.dataObjects;
for (var i = 0; i < d.length; i++)
    console.println("Data Object[" + i + "]=" + d[i].name);
```

delay

4.0			
-----	--	--	--

このプロパティを使用すると、文書内のすべてのフィールドの再描画を一時的に停止できます。複数のフィールドに対する一連の変更が終了してから、フィールドの外観をまとめて再描画したい場合に使用するのが一般的です。`true` にすると、すべての変更が待ち状態になります。`delay` を `false` にリセットすると、ページ上のすべてのフィールドが再描画されます。

Field オブジェクトの [delay](#) プロパティも参照してください。

型

ブーリアン

アクセス

R / W

dirty

3.01	D		X
------	---	--	---

文書が変更されて保存が必要な状態であるかどうかを指定します。保存を必要としないような変更を文書に加えた場合（例えば、ステータスフィールドを変更した場合）は、`dirty` フラグをリセットすると有益です。

注意：一時的に作成した文書や新規作成した文書では、`dirty` を `false` に設定しても効果はありません。文書を閉じる前に、変更を保存するように求めるプロンプトが表示されます。[requiresFullSave](#) を参照してください。

型

ブーリアン

アクセス

R / W

例 1

フォームをリセットし、`dirty` を `false` に設定します。リセットを行うと、文書を閉じるときにユーザが保存ダイアログボックスに対応しなくて済むようになります。

```
var f = this.getField("MsgField");
f.value = "You have made too many mistakes, I'm resetting the form. "
+ "Start over, this time follow the directions!";
this.resetForm();
this.dirty = false;
```

例 2

フォームへの記入をユーザに求める文を、テキストフィールドに入力します。スクリプトでは、この入力によって文書の保存状態が変更されないように処置を行っています。

```
var f = this.getField("MsgField");
var b = this.dirty;
f.value = "Please fill in the fields below.";
this.dirty = b;
```

disclosed

5.05			
------	--	---	--

他の文書の JavaScript スクリプトからこの文書にアクセス可能かどうかを指定します。

`app.openDoc` メソッドと `app.activeDocs` メソッドは、Doc を返す前に、文書の `disclosed` プロパティを確認します。

注意： `disclosed` プロパティを設定できるのは、バッチイベント、コンソールイベント、Page / Open イベント、Doc / Open イベントのみです。JavaScript イベントについて詳しくは、[event オブジェクト](#)を参照してください。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

型

ブーリアン

アクセス

R / W

例 1

文書レベル（またはページを開くアクション）の最上位に次のコードを配置すれば、その文書を `disclosed` にできます。

```
this.disclosed = true;
```

例 2

「JavaScript を実行」バッチシーケンスで次のコードを使用すれば、選択されているすべての文書を `disclosed` にできます。

```
this.addScript("Disclosed", "this.disclosed = true;");
```

docID

6.0			
-----	--	--	--

16進エンコードされたバイナリ形式の2つの文字列の配列。最初の文字列は、ファイルが最初に作成された時のコンテンツに基づく不变の識別子で、ファイルが繰り返し更新されても変わりません。2番目の文字列は、ファイルが最後に更新されたときのコンテンツに基づく可変の識別子です。これらの識別子は、PDFファイルのtrailerディクショナリ内にあるオプションのIDエントリによって定義されます（『PDF Reference』バージョン1.7を参照）。

型

配列

アクセス

R

使用例については、[346ページの「例6\(Acrobat 7.0\)」](#)を参照してください。

documentFileName

6.0			
-----	--	--	--

Docオブジェクトによって参照されている文書のファイル名（拡張子を含む）。デバイスに依存しないパスは返されません。[path](#)プロパティおよび[URL](#)プロパティも参照してください。文書のファイルサイズは[filesize](#)プロパティから取得できます。

型

文字列

アクセス

R

例

文書のファイル名を取得します。

```
console.println('The file name of this document is '
+ this.documentFileName +'.');
```

このマニュアル（『JavaScript for Acrobat API Reference』）でこのスクリプトを実行すると、次の文が表示されます。

```
"The file name of this document is js_api_reference.pdf"
```

dynamicXFAForm

7.0			
-----	--	--	--

この文書がダイナミック XFA フォームである場合は `true` が返され、それ以外の場合は `false` が返されます。

ダイナミック XFA フォームとは、フィールド内の値に合わせてフィールドの一部が拡大または縮小するフォームです。

型

ブーリアン

アクセス

R

例

使用例については、[XFA オブジェクト](#)を参照してください。

external

4.0			
-----	--	--	--

現在の文書が、Acrobat アプリケーションで表示されているか、外部ウィンドウ（Web ブラウザなど）で表示されているかを示します。

型

ブーリアン

アクセス

R

例

この文書がブラウザで表示されているかどうかを調べます。

```
if ( this.external )
{
    // ブラウザで表示
}
else
{
    // Acrobat アプリケーションで表示
}
```

filesize

3.01			
------	--	--	--

文書のファイルサイズ（バイト単位）。

型

整数

アクセス

R

例（Acrobat 5.0）

文書の保存前と保存後のファイルサイズの差を計算します。

```
// 次のコードを「文書を保存する」セクションに追加
var filesizeBeforeSave = this.filesize
console.println("File size before saving is " + filesizeBeforeSave);

// 次のコードを「文書を保存した」セクションに追加
var filesizeAfterSave = this.filesize
console.println("File size after saving is " + filesizeAfterSave);
var difference = filesizeAfterSave - filesizeBeforeSave;
console.println("The difference is " + difference );
if ( difference < 0 )
    console.println("Reduced filesize!");
else
    console.println("Increased filesize!");
```

hidden

7.0			
-----	--	--	--

文書のウィンドウが非表示の場合は、このプロパティが `true` になります。文書がバッチモードで処理されている場合や、明示的に非表示で開かれた場合に、ウィンドウが非表示になることがあります。`openDataObject` メソッドや `app.openDoc` メソッドを使用すれば、ウィンドウを表示せずに文書を開くことができます。

型

ブーリアン

アクセス

R

例

文書を開いて非表示ステータスを示します。

```
oDoc = app.openDoc({  
    cPath: "/C/myDocs/myHidden.pdf",  
    bHidden: true  
});  
console.println("It is " + oDoc.hidden + " that this document hidden.");  
oDoc.closeDoc();
```

hostContainer

7.0.5			
-------	--	--	--

PDF 文書が別のコンテナ（Web ブラウザなど）に埋め込まれている場合は、その `HostContainer` オブジェクトのインスタンス。それ以外の場合は `undefined`。

注意：このプロパティは、Macintosh プラットフォームでは実装されていません。

型

オブジェクト

アクセス

R / W

icons

5.0			
-----	--	--	--

文書レベルの名前付きアイコンツリーに存在している名前付き `Icon` オブジェクトの配列。文書に名前付きアイコンがない場合、このプロパティの値は `null` になります。

[addIcon](#)、[getIcon](#)、[importIcon](#)、[removeIcon](#)、Field オブジェクトの [buttonGetIcon](#)、[buttonImportIcon](#)、[buttonSetIcon](#) の各プロパティ、`Icon` オブジェクトも参照してください。

型

配列または `null`

アクセス

R

例 1

現在の文書に含まれている名前付きアイコンの数を表示します。

```
if (this.icons == null)
    console.println("No named icons in this doc");
else
    console.println("There are " + this.icons.length
        + " named icons in this doc");
```

例 2

現在の文書に含まれている名前付きアイコンをすべてリストします。

```
for (var i = 0; i < this.icons.length; i++) {
    console.println("icon[" + i + "]=" + this.icons[i].name);
}
```

info

5.0			
-----	--	--	--

PDF ファイルの文書情報ディクショナリのプロパティを持つオブジェクト（『PDF Reference』バージョン 1.7 を参照）。標準エントリは、次のとおりです。

Title
Author
Subject
Keywords
Creator
Producer
CreationDate
ModDate
Trapped

Acrobat では、このオブジェクトの各プロパティは書き込み可能です。プロパティを設定すると、文書が変更されることになります。非標準のプロパティを設定して、文書情報フィールドを追加することもできます。

Adobe Reader でこのオブジェクトのプロパティに書き込みを行うと、例外が発生します。

注意：標準エントリでは、大文字と小文字は区別されません。例えば、`info.Keywords` と `info.keywords` は同じになります。

型

オブジェクト

アクセス

R / W (Adobe Reader : R のみ)

例 1

現在の文書のタイトルを取得します。

```
var docTitle = this.info.Title;
```

例 2

文書に関する情報を取得します。

```
this.info.Title = "JavaScript, The Definitive Guide";
this.info.ISBN = "1-56592-234-4";
this.info.PublishDate = new Date();
for (var i in this.info)
    console.println(i + ": " + this.info[i]);
```

このスクリプトの出力は、次のようになります。

```
CreationDate: Mon Jun 12 14:54:09 GMT-0500 (Central Daylight Time) 2000
Producer: Acrobat Distiller 4.05 for Windows
Title: JavaScript, The Definitive Guide
Creator: FrameMaker 5.5.6p145
ModDate: Wed Jun 21 17:07:22 GMT-0500 (Central Daylight Time) 2000
SavedBy: Adobe Acrobat 4.0 Jun 19 2000
PublishDate: Tue Aug 8 10:49:44 GMT-0500 (Central Daylight Time) 2000
ISBN: 1-56592-234-4
```

innerAppWindowRect

6.0			
-----	--	--	--

このプロパティは、Acrobat の内側のアプリケーションウィンドウを表す画面座標の配列（矩形）を返します。この矩形には、タイトルバー や サイズ変更境界線などのアイテムは含まれません。これらのアイテムは、外側のアプリケーションウィンドウの矩形に含まれます。

型

数値の配列

アクセス

R

例

Acrobat の内側のアプリケーションウィンドウを取得してコンソールに表示します。

```
var coords = this.innerAppWindowRect;
console.println(coords.toSource())
// 出力例: [115, 154, 1307, 990]
```

[innerDocWindowRect](#)、[outerAppWindowRect](#)、[outerDocWindowRect](#) も参照してください。

innerDocWindowRect

6.0			
-----	--	--	--

このプロパティは、Acrobat の内側の文書ウィンドウを表す画面座標の配列（矩形）を返します。この矩形には、タイトルバーやサイズ変更境界線などのアイテムは含まれません。これらのアイテムは、外側の文書ウィンドウの矩形に含まれます。

プラットフォームによって、文書の矩形とアプリケーションの矩形が異なることがあります。例えば、Windows の場合、文書ウィンドウは常にアプリケーションウィンドウの内部にありますが、Macintosh では、両者は同じものになります。

型

数値の配列

アクセス

R

[innerAppWindowRect](#)、[outerAppWindowRect](#)、[outerDocWindowRect](#)、[pageWindowRect](#) も参照してください。

isModal

7.0.5			
-------	--	--	--

文書が現在モーダル状態（例えば、`app.execDialog` によってモーダルダイアログボックスが表示されている状態）であるかどうかを示すブーリアン値。

型

オブジェクト

アクセス

R

keywords

X	D		
---	---	--	--

注意：このプロパティの代わりに `info` プロパティを使用してください。

文書を説明するキーワード（「forms」、「taxes」、「government」など）。

型

オブジェクト

アクセス

R / W (Adobe Reader : R のみ)

layout

5.0			
-----	--	--	--

現在の文書のページレイアウトを変更します。有効な値は、次のとおりです。

SinglePage
OneColumn
TwoColumnLeft
TwoColumnRight

Acrobat 6.0 以降では、次の 2 つのプロパティが追加されています。

TwoPageLeft
TwoPageRight

型

文字列

アクセス

R / W

例

文書のレイアウトを連続見開きページにし、最初のページが左側に来るようになります。

```
this.layout = "TwoColumnLeft";
```

media

6.0			
-----	--	--	--

文書のマルチメディア関連のプロパティやメソッドが含まれたオブジェクト。このプロパティやメソッドについては、`Doc.media` オブジェクトを参照してください。

型

`Doc.media` オブジェクト

アクセス

R / W

metadata

6.0	D		X
-----	---	--	---

PDF 文書に埋め込まれた XMP メタデータにアクセスできます。メタデータを含む文字列を XML として返します。埋め込み XMP メタデータについて詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

型

文字列

アクセス

R / W

例外

メタデータを XMP 形式でない文字列に設定すると、`RaiseError` が発生します。

例 1

XMP 形式でないメタデータを作成しようとした場合です。

```
this.metadata = "this is my metadata";
RaiseError: The given metadata was not in the XMP format
Global.metadata:1:Console undefined:Exec
====> The given metadata was not in the XMP format
```

例 2

文書のメタデータを使用して PDF レポートファイルを作成します。

```
var r = new Report();
r.writeText(this.metadata);
r.open("myMetadataReportFile");
```

例 3

(Acrobat 8.0) この例では、E4X を使用して文書のメタデータを変更する方法を示します。このスクリプトでは、`Copyright Status`、`Copyright Notice`、`Copyright Info URL` の各フィールドを設定します。このスクリプトは、コンソールから実行するか、バッチシーケンスとして実行できます。

```
var CopyrightStatus = "True";
var CopyrightNotice = "Copyright (C) 2006, Adobe Systems, Inc."
var CopyrightInfoURL = "http://www.adobe.com"
var meta = this.metadata;
var myXMPData = new XML(meta);
myx = new Namespace("adobe:ns:meta/");
myrdf = new Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#");
mypdf = new Namespace("http://ns.adobe.com/pdf/1.3/");
myxap = new Namespace("http://ns.adobe.com/xap/1.0/");
mydc = new Namespace("http://purl.org/dc/elements/1.1/");
```

```
myXapRights = new Namespace("http://ns.adobe.com/xap/1.0/rights/");
var p = myXMPData.myrdf::RDF.myrdf::Description;
/*
    この要素に値が既に指定されているかどうかを調べ、指定されていない場合は
    値を割り当て、指定されている場合は別の値を割り当てる。
*/
if (p.mydc::rights.myrdf::Alt.myrdf::li.toString() == "") {
    p[0] += <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <dc:rights>
            <rdf:Alt>
                <rdf:li xml:lang="x-default">
                    {CopyrightNotice}
                </rdf:li>
            </rdf:Alt>
        </dc:rights>
    </rdf:Description>
} else
    p.mydc::rights.myrdf::Alt.myrdf::li = CopyrightNotice;
/*
    要素の中には属性に変換されるものがある。したがって、最初に
    xapRights:Marked 属性が存在するかどうかを調べて、存在しない場合は
    要素として追加し、存在する場合は属性を更新する。
    Acrobat では、特定の要素が属性に変更される。xapRights:Marked と
    xapRights:WebStatement は属性に変更される例で、前述の dc:rights は
    属性に変更されない例。
*/
if (p.@myXapRights::Marked.toString() == "") {
    p[0] += <rdf:Description rdf:about=""
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xapRights="http://ns.adobe.com/xap/1.0/rights/">
        <xapRights:Marked>{CopyrightStatus}</xapRights:Marked>
        <xapRights:WebStatement>{CopyrightInfoURL}</xapRights:WebStatement>
    </rdf:Description>
} else {
    p.@myXapRights::Marked = CopyrightStatus;
    p.@myXapRights::WebStatement = CopyrightInfoURL;
}
// // myXMPData を文字列に変換し、
myNewXMPStr=myXMPData.toXMLString();
// 文書メタデータに割り当てる。
this.metadata = myNewXMPStr;
```

modDate



注意：このプロパティの代わりに info プロパティを使用してください。

文書が最後に変更された日付。

型

日付

アクセス

R

mouseX

7.0			
-----	--	--	--

現在のページのデフォルトユーザースペースにおけるマウスの位置の x 座標を取得します。

型

数値

アクセス

R

例

ビューア内を移動するマウスの座標を取得します。

```
function getMouseCoor() {
    console.println( "("+this.mouseX+","+ this.mouseY+" ) " );
}
var ckMouse = app.setInterval("getMouseCoor()", 100);
var timeout = app.setTimeOut(
    "app.clearInterval(ckMouse); app.clearTimeOut(timeout)", 2000);
```

mouseY

7.0			
-----	--	--	--

現在のページのデフォルトユーザースペースにおけるマウスの位置の y 座標を取得します。

型

数値

アクセス

R

noautocomplete

7.0			
-----	--	--	--

このプロパティを使用して、この文書における Acrobat フォームのオートコンプリート機能をオフにできます。

- `true` になると、ユーザがフィールドにデータを入力するときに入力候補が表示されなくなります。
- `false` になると、「フォーム」ユーザ環境設定の「オートコンプリート」の設定に従ってオートコンプリートが実行されます。

このプロパティを設定しても、「オートコンプリート」の環境設定が変更されることはありません。

このプロパティの初期値は `undefined` です。

型

ブーリアン

アクセス

R / W

例

次のスクリプトは、ページを開くアクションや、最上位の文書レベルの JavaScript として実行できます。これによって、オートコンプリート機能がオフになります。

```
this.noautocomplete = true;
```

nocache

7.0			
-----	--	--	--

このプロパティを使用して、フォームデータのキャッシュ機能をこの文書のみオフにできます。

- `true` の場合、インターネットブラウザのフォームデータは Acrobat に保持されません。
- `false` の場合は、「フォーム」ユーザ環境設定の「フォームデータを一時的にディスクへ保存」に従って処理されます。

注意： `nocache` プロパティの値を設定しても、「フォームデータを一時的にディスクへ保存」チェックボックスには影響しません。

このプロパティの初期値は `undefined` です。

型

ブーリアン

アクセス

R / W

例

次のスクリプトは、フォームデータのキャッシングをオフにして、機密データがローカルのハードドライブに残らないようにします。このスクリプトは、ページを開くアクションや、最上位の文書レベルの JavaScript として実行できます。

```
this.nocache = true;
```

numFields

4.0			
-----	--	--	--

文書に含まれているフィールドの総数。[getNthFieldName](#) も参照してください。

型

整数

アクセス

R

例 1

```
console.println("There are " + this.numFields + " in this document");
```

例 2

このスクリプトでは、`numFields` プロパティと `getNthFieldName` メソッドを使用して、文書内のすべてのフィールドをループします。ここでは、すべてのボタンフィールドの外観をベベルに変更します（文書のボタンに対してその他の変更を加えることもできます）。

```
for ( var i = 0; i < this.numFields; i++ ) {
    var fname = this.getNthFieldName(i);
    if ( fname.type = "button" ) f.borderStyle = border.b;
}
```

numPages

3.01			
------	--	--	--

文書内のページ数。

型

整数

アクセス

R

例 1

```
console.println("There are " + this.numPages + " in this document");
```

例 2

文書から最後のページを削除します。文書の最後のページの番号は、`this.numPages - 1` です（ページは 0 から数えます）。

```
this.deletePages({ nStart: this.numPages - 1 });
```

numTemplates

X			
---	--	--	--

注意：このプロパティの代わりに `templates` を使用してください。

文書内のテンプレート数。

型

整数

アクセス

R

path

3.01			
------	--	--	--

デバイスに依存しない、文書のパス。例えば、次のようにになります。

`/c/Program Files/Adobe/Acrobat 5.0/Help/AcroHelp.pdf`

型

文字列

アクセス

R

文書のファイル名は、`documentFileName` プロパティで取得できます。[URL](#) プロパティも参照してください。

outerAppWindowRect

6.0			
-----	--	--	--

このプロパティは、Acrobat の外側のアプリケーションウィンドウを表す画面座標の配列（矩形）を返します。この矩形には、タイトルバーやサイズ変更境界線などのアイテムが含まれます。これらのアイテムは、内側のアプリケーションウィンドウの矩形には含まれません。

型

数値の配列

アクセス

R

[innerAppWindowRect](#)、[outerDocWindowRect](#)、[outerDocWindowRect](#)、[pageWindowRect](#) も参照してください。

outerDocWindowRect

6.0			
-----	--	--	--

このプロパティは、Acrobat の外側の文書ウィンドウを表す画面座標の配列（矩形）を返します。この矩形には、タイトルバーやサイズ変更境界線などのアイテムが含まれます。これらのアイテムは、内側の文書ウィンドウの矩形には含まれません。

プラットフォームによって、アプリケーションの矩形と文書の矩形が異なることがあります。例えば、Windows の場合、文書ウィンドウは常にアプリケーションウィンドウの内部にありますが、Macintosh では、両者は同じものになります。

型

数値の配列

アクセス

R

[innerAppWindowRect](#)、[outerDocWindowRect](#)、[outerAppWindowRect](#)、[pageWindowRect](#) も参照してください。

pageNum

3.01			
------	--	--	--

文書の現在のページを取得または設定します。pageNum を特定のページに設定する場合、ページは 0 から数えることに注意してください。

型

整数

アクセス

R / W

例

文書の最初のページに移動します。

```
this.pageNum = 0;
```

文書の次のページに移動します。

```
this.pageNum++;
```

pageWindowRect

6.0			
-----	--	--	--

Acrobat のページビューウィンドウを表す画面座標の配列（矩形）。ページビューウィンドウとは、PDF コンテンツが表示される、内側の文書ウィンドウの内部にある領域のことです。

型

数値の配列

アクセス

R

[innerAppWindowRect](#)、[outerDocWindowRect](#)、[outerAppWindowRect](#)、[outerDocWindowRect](#) も参照してください。

permStatusReady

6.0			
-----	--	--	--

この文書の権限が解決済みかどうかを示すブーリアン値。

文書全体を対象とする証明用の署名に基づいて権限が決定される文書を、ネットワーク接続を介して途中までしかダウンロードしていない場合は、文書が使用できることを示す `false` になります。

型

ブーリアン

アクセス

R

producer

X			
---	--	--	--

注意：このプロパティの代わりに [info](#) プロパティを使用してください。

文書を PDF 変換したソフトウェア（「Acrobat Distiller®」、「PDFWriter」など）。

型

文字列

アクセス

R

requiresFullSave

7.0			
-----	--	--	--

文書が一時的または新規に作成されたため、完全な保存が必要である場合は `true` になります。その他の場合は `false` になります。

型

ブーリアン

アクセス

R

例

```
var oDoc = app.newDoc();
console.println("It is " + oDoc.requiresFullSave
+ " that this document requires a full save.");
```

securityHandler

5.0			
-----	--	--	--

文書の暗号化に使用されたセキュリティハンドラの名前。セキュリティハンドラがない場合（例えば、文書が暗号化されていない場合）は、`null` を返します。

型

文字列

アクセス

R

例

```
console.println(this.securityHandler != null ?  
    "This document is encrypted with " + this.securityHandler  
    + " security." : "This document is unencrypted.");
```

文書が標準セキュリティハンドラで暗号化されている場合、出力は次のようになります。

```
This document is encrypted with Standard security.
```

selectedAnnots

5.0			C
-----	--	--	---

現在選択されているすべてのマークアップ注釈に対応する `Annotation` オブジェクトの配列。

[getAnnot](#) および [getAnnots](#) も参照してください。

型

配列

アクセス

R

例

選択されている注釈のすべてのコメントをコンソールに表示します。

```
var aAnnots = this.selectedAnnots;  
for (var i=0; i < aAnnots.length; i++)  
    console.println(aAnnots[i].contents);
```

sounds

5.0			
-----	--	--	--

文書内の名前付き `Sound` オブジェクトがすべて含まれた配列。

[getSound](#)、[importSound](#)、[deleteSound](#)、[Sound](#) オブジェクトも参照してください。

型

配列

アクセス

R

例

```
var s = this.sounds;  
for (i = 0; i < s.length; i++)  
    console.println("Sound[" + i + "]=" + s[i].name);
```

spellDictionaryOrder

5.0			
-----	--	--	--

この文書における辞書の検索順序を示す配列。例えば、医学に関するフォームを作成してユーザ入力のスペルチェックを行う場合、ユーザの環境設定で指定されている辞書でチェックする前に、医学辞書でチェックすることができます。

Spelling プラグインは、まずこの配列で指定されている辞書で単語を検索してから、「スペルチェック」環境設定パネルでユーザが選択した辞書を検索します。ユーザが設定した検索順序は、spell.dictionaryOrder で取得できます。現在インストールされている辞書の配列は、spell.dictionaryNames で取得できます。

注意：このプロパティを設定するときに、配列の要素に無効な辞書名があると例外が発生します。

型

配列

アクセス

R / W

spellLanguageOrder

6.0			X
-----	--	--	---

この文書における言語の検索順序を示す配列。Spelling プラグインは、まずこの配列で指定されている言語で単語を検索してから、「スペルチェック」環境設定パネルでユーザが選択した言語を検索します。ユーザが設定した検索順序は、spell.languageOrder で取得できます。現在インストールされている言語の配列は、spell.languages プロパティで取得できます。

型

配列

アクセス

R / W

subject

X	D		
---	---	--	--

注意：このプロパティの代わりに `info` プロパティを使用してください。

文書のサブタイトル。このプロパティは、Adobe Reader では読み取り専用です。

型

文字列

アクセス

R / W

templates

5.0			
-----	--	--	--

文書内のすべての `Template` オブジェクトの配列。[createTemplate](#)、[getTemplate](#)、[removeTemplate](#) も参照してください。

型

配列

アクセス

R

例

文書中のすべてのテンプレートをリストします。

```
var t = this.templates
for ( var i=0; i < t.length; i++)
{
    var state = (t[i].hidden) ? "visible" : "hidden"
    console.println("Template: $" + t[i].name
                    + "$, current state: " + state);
}
```

title

X	D		X
---	---	--	---

注意：このプロパティの代わりに `info` プロパティを使用してください。

文書のタイトル。

型

文字列

アクセス

R / W (Adobe Reader : R のみ)

URL

5.0			
-----	--	--	--

文書の URL。文書がローカルにある場合、Windows と UNIX では `file:///` で始まる URL を、Macintosh では `file://localhost/` で始まる URL を返します。これは `baseURL` とは異なることがあります。

型

文字列

アクセス

R

`path` プロパティおよび `documentFileName` プロパティも参照してください。

viewState

7.0.5			
-------	--	--	--

文書の現在の表示状態を表すオブジェクト。この状態には、少なくとも、現在のページ番号、スクロール位置、倍率、フィールドフォーカスに関する情報が含まれます。

この値を設定するには、取得済みの値を利用する必要があります。これを使用して、文書の表示状態を復元することができます。

注意：このオブジェクトは、埋め込まれている PDF でのみ定義されます。

型

オブジェクト

アクセス

R / W

例

この例では、表示状態を取得してホストアプリケーションに送信します。ホストアプリケーションは、表示状態を保存し、後でビューアの表示状態を復元することができます。このコードは、PDF 文書内のボタンで実行することができます。配列の最初のエントリは、メッセージの種類をホストに伝えるためのシグナルです。

```
if(this.hostContainer)
{
    cViewState = this.viewState.toSource();
    aMsg = new Array( "viewState", cViewState );
    this.hostContainer.postMessage(aMsg);
}
```

ホストアプリケーションで、次のようなメッセージハンドラを定義します。

```
var cViewState="" ; // viewState を保存するための変数
function onMessageFunc( stringArray )
{
    var PDFObject = document.getElementById( PDFObjectID );
    if ( this != PDFObject.messageHandler )
        alert( "Incorrect this value in onMessage handler" );
    // 取得した配列の最初のエントリはシグナル
    var signal = stringArray[0];

    switch ( signal ) {
        case "Msg":
            var msgStr = "";
            for ( var i = 1; i < stringArray.length; i++ )
                msgStr += (stringArray[ i ] + "<br>");
            writeMsg( msgStr ); // 文書に書き込む関数
            break;

        case "viewState":
            // 表示状態を保存
            cViewState = stringArray[1];
            break;
    }
}
```

ボタンを使用して、埋め込まれた PDF に cViewState の値を戻すことができます。PDF の文書レベルの JavaScript で、次のコードを定義します。

```
if ( this.hostContainer )
{
    myHostContainer = this.hostContainer;
    myHostContainer.messageHandler = {
        onMessage: function(aMessage) {
            var f = this.doc.getField("msg4pdf");
            var strValue = "";
            var signal = aMessage[0];
            switch ( signal ) {
                case "Msg":
```

```
        for(var i = 1; i < aMessage.length; i++)
            strValue += aMessage[i] + "\r";
        f.value = strValue;
        break;
    case "viewState":
        var restoreViewState = eval( aMessage[1] );
        // viewType をリセット。this の doc プロパティを使用して
        // 正しい Doc を取得している点に注意。
        this.doc.viewState = restoreViewState;
        break;
    }
},
onError: function(error, aMessage) {
    console.println("error: "+ error.toString())
},
onDisclose: HostContainerDisclosurePolicy.SameOriginPolicy,
allowDeliverWhileDocIsModel: true
};
// this オブジェクトは、このメソッドを呼び出している messageHandler
// のインスタンスなので、messageHandler インスタンスの doc
// プロパティとして Doc を保存。
myHostContainer.messageHandler.doc = this;
}
```

xfa

6.0.2			
-------	--	--	--

このプロパティは、文書が XML フォームの場合にのみ定義されます。つまり、LiveCycle Designer で作成された文書でのみ定義されます。このプロパティが定義されている場合、`xfa` は静的な XFAObject です。これは、基盤となる `xfa` モデルのルートノードであり、これを使用して `xfa` のスクリプトオブジェクトモデル (SOM) にアクセスできます。

`xfa` の SOM について詳しくは、『Adobe XML Form Object Model Reference』というマニュアルを参照してください。『Converting Acrobat JavaScript for Use in LiveCycle Designer Forms』というマニュアルでは、Acrobat と LiveCycle Designer のスクリプトオブジェクトモデルの違いが説明されています。

注意：このプロパティをフォルダレベルのスクリプトから参照する場合は、目的の文書の Doc オブジェクトを渡して、`xfa` が適切なコンテキストで参照されるようにしてください。例 2 を参照してください。

型

XFAObject

アクセス

R

例 1

この文書が XML フォームであり、「EmployeeName」というテキストフィールドがあるとします。この例では、`xfa` オブジェクトを使用してこのフィールドの値にアクセスし、値を変更します。

```
var eN = this.xfa.form.form1.EmployeeName;  
console.println("EmployeeName: " + eN.rawValue);
```

コンソールへの出力は、次のようにになります。

```
EmployeeName: A. C. Robat
```

次に `EmployeeName` の値を変更します。

```
eN.rawValue = "Robat, A. C."  
console.println("EmployeeName: " + eN.rawValue);
```

コンソールへの出力は、次のようにになります。

```
EmployeeName: Robat, A. C.
```

フィールドの値が変更されました。

例 2

フォルダレベルのスクリプトファイルに、`xfa` プロパティを使用する関数を定義します。これを呼び出すには `Doc` オブジェクトを渡します。

```
function isXFA(doc) {  
    var wasWasNot = (typeof doc.xfa == "undefined") ? "not" : "";  
    console.println("This document was " + wasWasNot + "created by Designer.");  
}
```

文書の内部からであっても、コンソールからであっても、この関数は `isXFA(this)` によって呼び出せます。

XFAForeground

8.0			
-----	--	--	--

この文書が XFA Foreground タイプのフォームである場合は `true` が返され、それ以外の場合は `false` が返されます。

バージョン 8.0 以降の LiveCycle Designer では、PDF ファイルをアートワークとして取り込むことができます。PDF の描画内容は背景として使用され、LiveCycle Designer を使用してこの背景の上にフォームフィールドを配置することができます。`XFAForeground` プロパティは、PDF がこの方法で作成されたかどうかを表します。値が `true` である場合は、LiveCycle Designer でアートワークとして PDF が取り込まれ、LiveCycle Designer によって保存されたことを表します。

型

ブーリアン

アクセス

R

例

このスクリプトは、現在の文書が XFA Foreground タイプのフォームであるかどうか（LiveCycle Designer で PDF を取り込んで保存する方法で作成された文書かどうか）を判別します。

```
if ( this.XFAForeground )
    console.println("This is an XFA Foreground form.");
```

zoom

3.01			
------	--	--	--

現在のページのズームレベル。指定可能な範囲は 8.33 ~ 6400 %で、パーセント値として指定します。例えば、ズーム値 100 は 100 %です。

型

数値

アクセス

R / W

例

現在のズームレベルの 2 倍に設定します。

```
this.zoom *= 2;
```

ズームを 200 %に設定します。

```
this.zoom = 200;
```

zoomType

3.01			
------	--	--	--

文書の現在のズームタイプ。有効なズームタイプを、次の表に示します。

zoomtype オブジェクトには有効なズームタイプがすべて定義されているので、このオブジェクトを使用してすべてのズームタイプにアクセスできます。

ズームタイプ	キーワード	バージョン
NoVary	zoomtype.none	
FitPage	zoomtype.fitP	
FitWidth	zoomtype.fitW	
FitHeight	zoomtype.fitH	
FitVisibleWidth	zoomtype.fitV	

ズームタイプ	キーワード	バージョン
Preferred	zoomtype.pref	
ReflowWidth	zoomtype.refW	6.0

型

文字列

アクセス

R / W

例

文書のズームタイプを「幅に合わせる」に設定します。

```
this.zoomType = zoomtype.fitW;
```

Doc のメソッド

addAnnot

5.0	④		⑤
-----	---	--	---

指定のプロパティを持つ Annotation オブジェクトを作成します。プロパティを指定しなかった場合は、指定したタイプの注釈のデフォルト値が使用されます。

注意：(Acrobat 8.0) `author` プロパティを指定しなかった場合の `addAnnot` の動作が変更されました。セキュリティによる制限があるコンテキストで `addAnnot` を実行した場合、`author` プロパティのデフォルト値は `undefined` という文字列になります。セキュリティによる制限がないコンテキストで `addAnnot` を実行した場合、`author` プロパティのデフォルト値は、現在のユーザのログイン名になります。

パラメータ

オブジェクトリテラル	作成する Annotation オブジェクトのプロパティ (<code>type</code> 、 <code>rect</code> 、 <code>page</code> など) を指定する汎用オブジェクト。
------------	---

戻り値

新しい Annotation オブジェクト。

例 1

この単純な例では、長方形注釈を作成します。

```
var sqannot = this.addAnnot({type: "Square", page: 0});
```

長方形注釈 `sqannot` が、最初のページ（ページは 0 から数えます）に作成されます。

例 2

各種のプロパティを指定したテキスト注釈を追加します。

```
var annot = this.addAnnot({  
    page: 0,  
    type: "Text",  
    author: "A. C. Robat",  
    point: [300, 400],  
    strokeColor: color.yellow,  
    contents: "Need a little help with this paragraph.",  
    noteIcon: "Help"  
});
```

例 3

各種のプロパティを指定した長方形注釈を追加します。

```
var annot = this.addAnnot({  
    page: 0,  
    type: "Square",  
    rect: [0, 0, 100, 100],  
    name: "OnMarketShare",  
    author: "A. C. Robat",  
    contents: "This section needs revision."  
});
```

例 4

三つ葉模様の鉛筆注釈を追加します。

```
var inch = 72, x0 = 2*inch, y0 = 4*inch;  
var scaledInch = .5*inch;  
var nNodes = 60;  
var theta = 2*Math.PI/nNodes;  
var points = new Array();  
for (var i = 0; i <= nNodes; i++) {  
    Theta = i*theta;  
    points[i] = [x0 + 2*Math.cos(3*Theta)*Math.cos(Theta)*scaledInch,  
                y0 + 2*Math.cos(3*Theta)*Math.sin(Theta)*scaledInch];  
}  
var annot = this.addAnnot({  
    type: "Ink",  
    page: 0,  
    name: "myRose",  
    author: "A. C. Robat",  
    contents: "Three leaf rose",  
    gestures: [points],  
    strokeColor: color.red,  
    width: 1  
});
```

addField



新しいフォームフィールドを作成し、そのフォームフィールドを Field オブジェクトとして返します。

注意：(Acrobat 6.0) Acrobat 6.0 以降では、文書にフォーム使用権限が追加されていれば、Adobe Reader でこのメソッドを使用することができます。6.0よりも前のバージョンでは、Adobe Reader でこのメソッドを使用することはできません。

パラメータ

cName	作成する新しいフィールドの名前。この名前をドットで区切って階層を示すことができます（例えば、name.last は親ノード name と子ノード last を示します）。
cFieldType	作成するフォームフィールドのタイプ。有効な値は、次のとおりです。 text button combobox listbox checkbox radiobutton signature
nPageNum	フィールドを追加するページのインデックス（0 から数えます）。
oCoords	フォームフィールドのサイズと位置を指定する、回転ユーザースペースにおける 4 つの数値の配列。これらの 4 つの数値は、境界を表す矩形の 左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。Field オブジェクトの rect プロパティも参照してください。
	注意： 情報パネルを使用して矩形の座標を調べた場合は、情報パネルの座標を回転ユーザースペースの座標に変換する必要があります。変換するには、画面のページの高さから、情報パネルの y 座標を減算します。

戻り値

新しく作成された Field オブジェクト。

例

次のコードをバッチシーケンスなどで使用して、バッチ処理の対象となる文書の全ページにナビゲーションアイコンを作成することができます。

```
var inch = 72;
for (var p = 0; p < this.numPages; p++) {
    // 矩形 (0.5 インチ, 0.5 インチ) を配置
    var aRect = this.getPageBox( {nPage: p} );
    aRect [0] += .5*inch;                      // ページの左上隅から
    aRect [2] = aRect [0]+.5*inch;              // 幅 0.5 インチ
    aRect [1] -= .5*inch;
    aRect [3] = aRect [1] - 24;                 // 高さ 24 ポイント

    // ZapfDingbats の右矢印を使用してボタンフィールドを作成
```

```
var f = this.addField("NextPage", "button", p, aRect )
f.setAction("MouseUp", "this.pageNum++");
f.delay = true;
f.borderStyle = border.s;
f.highlight = "push";
f.fontSize = 0; // サイズは自動で
f.textColor = color.blue;
f.fillColor = color.ltGray;
f.textFont = font.Zapfd;
f.buttonSetCaption("¥341") // 右矢印
f.delay = false;
}
```

別の使用例については、Field オブジェクトの [setAction](#) メソッドを参照してください。

addIcon



新しい名前付き Icon オブジェクトを文書レベルのアイコンツリーに追加し、指定した名前で保存します。

[icons](#)、[getIcon](#)、[importIcon](#)、[removeIcon](#)、Field オブジェクトの [buttonGetIcon](#)、[buttonImportIcon](#)、[buttonSetIcon](#) の各メソッドも参照してください。

パラメータ

cName	新しいオブジェクトの名前。
icon	追加する Icon オブジェクト。

例

この例では、文書のフォームボタンフィールドにアイコンが設定されていることを前提として、そのアイコンに名前を割り当てます。例えば、`getIcon` にこの名前を指定して icon オブジェクトを取得し、別のボタンで使用することができます。

```
var f = this.getField("myButton");
this.addIcon("myButtonIcon", f.buttonGetIcon());
```

addLink



文書にリンクを追加する権限がユーザにある場合は、指定したページに、指定した座標で新しいリンクを追加します。[getLinks](#)、[removeLinks](#)、[Link](#) オブジェクトも参照してください。

パラメータ

nPage	新しいリンクを追加するページ。
oCoords	リンクのサイズと位置を指定する、4 つの数値の配列。これらの数値は、回転ユーザースペースにおける境界を表す矩形の 左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。

戻り値

新しく作成された Link オブジェクト。

例 1

現在の文書の各ページの左下隅と右下隅に、簡単なナビゲーションリンクを作成します。左下隅のリンクは前のページ、右下隅のリンクは次のページへ進みます。

```
var linkWidth = 36, linkHeight = 18;
for ( var i=0; i < this.numPages; i++)
{
    var cropBox = this.getPageBox("Crop", i);
    var linkRect1 = [0,linkHeight,linkWidth,0];
    var offsetLink = cropBox[2] - cropBox[0] - linkWidth;
    var linkRect2 = [offsetLink,linkHeight,linkWidth + offsetLink,0]
    var lhLink = this.addLink(i, linkRect1);
    var rhLink = this.addLink(i, linkRect2);
    var nextPage = (i + 1) % this.numPages;
    var prevPage = (i - 1) % this.numPages;
    var prevPage = (prevPage>=0) ? prevPage : -prevPage;
    lhLink.setAction( "this.pageNum = " + prevPage);
    lhLink.borderColor = color.red;
    lhLink.borderWidth = 1;
    rhLink.setAction( "this.pageNum = " + nextPage);
    rhLink.borderColor = color.red;
    rhLink.borderWidth = 1;
}
```

リンクのプロパティやアクションの設定について詳しくは、[Link オブジェクト](#)を参照してください。

例 2

文書全体にわたって「Acrobat」という単語を検索し、この単語にリンクを作成します。

```
for (var p = 0; p < this.numPages; p++)
{
    var numWords = this.getPageNumWords(p);
    for (var i=0; i<numWords; i++)
    {
        var ckWord = this.getPageNthWord(p, i, true);
        if ( ckWord == "Acrobat")
        {
            var q = this.getPageNthWordQuads(p, i);
            // デフォルトユーザースペースの四角形を、回転したユーザースペースの座標に変換して
            // リンクの作成に使用
            m = (new Matrix2D).fromRotated(this,p);
            mInv = m.invert();
            r = mInv.transform(q)
            r=r.toString()
            r = r.split(",");
            l = addLink(p, [r[4], r[5], r[2], r[3]]);
            l.borderColor = color.red
        }
    }
}
```

```
    l.borderWidth = 1
    l.setAction("this.getURL('http://www.adobe.com/');");
}
}
}
```

addRecipientListCryptFilter

6.0			
-----	--	--	--

文書に暗号フィルタを追加します。暗号フィルタは、Data オブジェクトの暗号化に使用します。

[importDataObject](#)、[createDataObject](#)、[setDataObjectContents](#) の各メソッドのパラメータである cCryptFilter も参照してください。

注意：このメソッドを実行できるのは、バッヂイベント、アプリケーション初期化イベント、コンソールイベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

パラメータ

cCryptFilter 言語に依存しない、暗号フィルタの名前。これと同じ名前を、Doc の importDataObject、createDataObject、setDataObjectContents の各メソッドの cCryptFilter パラメータで使用します。

注意：Acrobat 7.0 では、cCryptFilter の値は DefEmbeddedFile である必要があります。他のバージョンの Acrobat では、cCryptFilter の値には任意の文字列を指定できます。

oGroup データ暗号化の対象となる受信者をリストした Group オブジェクトの配列。

例

このスクリプトでは、現在の文書を暗号化して、PDF 文書に埋め込みます。

```
var Note = "Select the list of people that you want to send this"
+ " document to. Each person must have both an email address"
+ " and a certificate that you can use when creating the"
+ " envelope.";
var oOptions = { bAllowPermGroups: false, cNote: Note,
    bRequireEmail: true };
var oGroups = security.chooseRecipientsDialog( oOptions );
var env = app.openDoc( "/c/temp/ePaperMailEnvelope.pdf" );
env.addRecipientListCryptFilter( "MyFilter", oGroups );
env.importDataObject( "secureMail0", this.path, "MyFilter" );
var envPath = "/c/temp/outMail.pdf";
env.saveAs( envPath );
```

注意：このスクリプトはコンソールで実行できますが、フォルダレベルの JavaScript の一部として実行することで、PDF 文書を安全に送信する機能の開発に活かすことができます。

addRequirement

7.0.5	D	S	X
-------	---	---	---

特定の要件が満たされている場合にのみ、その文書の機能が Acrobat で適切に動作するようにします。

要件が含まれている文書を開くと、文書が自由に操作できるようになる前に、Acrobat によって要件が確認されます。要件が満たされていない場合は、アプリケーションによって文書の機能が制限されることがあります。

注意：このメソッドを呼び出せるのは、コンソールイベントまたはバッヂイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

パラメータ

cType	文書の要件のタイプ。タイプは Requirements 列挙子オブジェクトで指定します。
oReq	(オプション) Requirement オブジェクト。

Requirements 列挙子オブジェクト

このオブジェクトには、Acrobat で適切に機能するための要件として文書に設定できるすべての要件タイプが含まれています。

プロパティ	説明
requirements.EnableJavaScripts	ユーザ環境設定の「Acrobat JavaScript を使用」を無効にしている場合は、文書にデータ検証スクリプトを含めていても、そのスクリプトは実行されません。このプロパティを使用すれば、Acrobat で PDF 文書のスクリプトを強制的に実行することができます。その際には、その文書における JavaScript の実行を許可するか、読み取り専用モードで文書を開くかを選択するように求めるプロンプトがユーザに表示されます。

Requirement オブジェクト

この汎用オブジェクトには、要件の特性を指定するプロパティが含まれています。

プロパティ	説明
aRH	(オプション) ReqHandler オブジェクトの配列。

ReqHandler オブジェクト

この汎用オブジェクトでは、認識できない要件が Acrobat で検出された場合に使用する要件ハンドラの情報を指定します。認識できない要件は、該当するタイプをサポートする配列内の最初のハンドラによって確認されます。認識できない要件を処理できる要件ハンドラが見つからない場合は、ビューアによって汎用メッセージが表示されます。

プロパティ	説明
cType	要件ハンドラのタイプを指定する文字列（有効な名前については、 ReqHandlers 列挙子オブジェクト を参照）。
cScriptName	（オプション）文書内に存在する文書レベル JavaScript の名前を指定する文字列。 cType の値が reqHandlers.js の場合に指定することができます。 ここで指定したスクリプトは、要件が満たされた場合には実行されません。

ReqHandlers 列挙子オブジェクト

このオブジェクトには、文書に含めることができる要件ハンドラのタイプが列挙されています。

プロパティ	説明
reqHandlers.JS	このハンドラに登録された文書レベルスクリプトを使用して、認識できない要件を処理します。
reqHandlers.NoOp	このハンドラを使用すると、認識できない要件を旧バージョンのビューアで無視できるようになります。

例

JavaScript が文書で有効であるという要件を追加します。

```
addRequirement(this.requirements.EnableJavaScripts,  
    { [ {cType: reqHandlers.JS, cScriptName: "requirement"} ] } );
```

addScript

6.0			
-----	--	--	--

文書に文書レベルのスクリプトを設定します。[setAction](#)、[setPageAction](#)、Bookmark オブジェクトの [setAction](#) メソッド、Field オブジェクトの [setAction](#) メソッドも参照してください。

注意： cName で指定したスクリプト名が既に使用されている場合、このメソッドを実行すると既存のスクリプトが新たな内容に上書きされます。

パラメータ

cName	スクリプトの名前。同じ名前のスクリプトが既に存在している場合は、古いスクリプトが新しいスクリプトに置換されます。
cScript	文書が開かれたときに実行する JavaScript 式。

例

文書が開かれるたびにビープ音を鳴らします。

```
this.addScript("My Code", "app.beep(0);");
```

他の例については、[disclosed](#) プロパティの[例 2](#) を参照してください。

addThumbnails

5.0			
-----	--	--	--

指定したページのサムネールを作成します。[removeThumbnails](#) メソッドも参照してください。

パラメータ

nStart	(オプション) ページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
nEnd	(オプション) ページ範囲の終了を定義するインデックス (0 から数えます)。詳しくは、nStart を参照してください。

addWatermarkFromFile

7.0			
-----	--	--	--

文書の指定のページに透かしを追加します。透かしはオプショナルコンテンツグループ (OCG) に配置します。[OCG](#) オブジェクトも参照してください。

注意： このメソッドを実行できるのは、バッヂイベントまたはコンソールイベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。

パラメータ

cDIPath	デバイスに依存しない、透かしとして使用するソースファイルのパス。この場所にあるファイルが PDF ファイルでない場合は、Acrobat によって PDF ファイルへの変換が試みられます。
nSourcePage	(オプション) 透かしとして使用する、ソースファイルのページのインデックス (0 から数えます)。デフォルトは 0 です。
nStart	(オプション) 透かしを追加するページ範囲の最初のページのインデックス (0 から数えます)。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
nEnd	(オプション) 透かしを追加するページ範囲の最後のページ。詳しくは、nStart を参照してください。
bOnTop	(オプション) 透かしの z オーダーを示すブーリアン値。true (デフォルト) の場合は、他のすべてのページコンテンツの上に透かしが追加されます。false の場合は、他のすべてのページコンテンツの下に透かしが追加されます。このパラメータは、bFixedPrint が true の場合には無視されます。
bOnScreen	(オプション) 文書を画面に表示する際に、透かしを表示するかどうかを示すブーリアン値。デフォルトは true です。

bOnPrint	(オプション) 文書を印刷する際に、透かしを表示するかどうかを示すブーリアン値。デフォルトは <code>true</code> です。
nHorizAlign	(オプション) 透かしの水平方向の整列方法を表す数値。有効な値については、 <code>app.constants.align</code> を参照してください。デフォルトは <code>app.constants.align.center</code> です。
nVertAlign	(オプション) 透かしの垂直方向の整列方法を表す数値。有効な値については、 <code>app.constants.align</code> を参照してください。デフォルトは <code>app.constants.align.center</code> です。
nHorizValue	(オプション) ページ上で透かしの水平位置をシフトさせるために使用する数値。 <code>bPercentage</code> が <code>true</code> の場合、この数値はページの幅のパーセンテージを表します。 <code>bPercentage</code> が <code>false</code> の場合、この数値はオフセットするポイント数を表します。デフォルトは <code>0</code> です。
nVertValue	(オプション) ページ上で透かしの垂直位置をシフトさせるために使用する数値。 <code>bPercentage</code> が <code>true</code> の場合、この数値はページの高さのパーセンテージを表します。 <code>bPercentage</code> が <code>false</code> の場合、この数値はオフセットするポイント数を表します。デフォルトは <code>0</code> です。
bPercentage	(オプション) <code>nHorizValue</code> や <code>nVertValue</code> がページサイズのパーセンテージを表すのか、ポイント数を表すのかを示すブーリアン値。デフォルトは <code>false</code> です。
nScale	(オプション) 透かしに使用するスケール。 <code>1.0</code> が <code>100 %</code> です。値が <code>-1</code> の場合は、透かし形状を維持したまま、大きさをページにフィットさせます。デフォルトは <code>1.0</code> です。
bFixedPrint	(オプション) 透かしを FixedPrint Watermark 注釈として追加することを示すブーリアン値。これによって、印刷するページのサイズに関係なく、透かしが固定のサイズおよび位置に印刷されます。 <code>true</code> の場合は、 <code>boTop</code> は無視されます。デフォルトは <code>false</code> です。
nRotation	(オプション) 透かしを反時計回りに回転させる角度。デフォルトは <code>0</code> です。
nOpacity	(オプション) 透かしに使用する不透明度。 <code>0</code> が透明で、 <code>1.0</code> が不透明です。デフォルトは <code>1.0</code> です。

例 1

`watermark.pdf` の最初のページを透かしとして使用し、それを現在の文書のすべてのページの中央に追加します。

```
this.addWatermarkFromFile("/C/temp/watermark.pdf");
```

例 2

`watermark.pdf` の 5 番目のページを透かしとして使用し、それを現在の文書の最初の 10 ページに追加します。透かしは反時計回りに 45° 回転し、ページの左上隅から 1 インチ下および 2 インチ右に配置します。

```
this.addWatermarkFromFile({
  cDIPath: "/C/temp/watermark.pdf",
  nSourcePage: 4, nEnd: 9,
  nHorizAlign: app.constants.align.left,
```

```
nVertAlign: app.constants.align.top,  
nHorizValue: 144, nVertValue: -72,  
nRotation: 45  
});
```

addWatermarkFromText

7.0		
-----	--	--

文書の指定のページに指定のテキストを透かしとして追加し、透かしをオプショナルコンテンツグループ(OCG)に配置します。

[OCG](#) オブジェクトを参照してください。

パラメータ

cText	透かしとして使用するテキスト。複数行のテキストを使用できます。「¥r」という文字列で改行を指定できます。
n TextAlign	(オプション) 透かし内の cText のテキスト整列。有効な値については、app.constants.align を参照してください。cText が 1 行のみの場合はこのパラメータを指定しても影響はありません。
cFont	(オプション) 透かしに使用するフォント。有効なフォントは、font オブジェクトのプロパティとして定義されています（その一覧については、Field オブジェクトの textFont プロパティを参照）。任意のフォントを使用するには、フォントの PostScript 名を表す文字列を渡します。デフォルトは font.Helv です。
nFontSize	(オプション) 透かしに使用するフォントのポイントサイズ。デフォルトは 24 です。
aColor	(オプション) 透かしに使用する色。 191 ページの「カラー配列」 を参照してください。デフォルトは color.black です。
nStart	(オプション) 透かしを追加するページ範囲の最初のページのインデックス（0 から数えます）。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 透かしを追加するページ範囲の最後のページ。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
bOnTop	(オプション) 透かしの z オーダーを示すブーリアン値。値が true の場合は、他のすべてのページコンテンツの上に透かしが追加されます。値が false の場合は、他のすべてのページコンテンツの下に透かしが追加されます。このパラメータは、bFixedPrint が true の場合には無視されます。デフォルトは true です。
bOnScreen	(オプション) 文書を画面に表示する際に、透かしを表示するかどうかを示すブーリアン値。
bOnPrint	(オプション) 文書を印刷する際に、透かしを表示するかどうかを示すブーリアン値。

nHorizAlign	(オプション) 透かしの水平方向の整列方法を表す数値。有効な値については、 app.constants.align を参照してください。デフォルトは app.constants.align.center です。
nVertAlign	(オプション) 透かしの垂直方向の整列方法を表す数値。有効な値については、 app.constants.align を参照してください。デフォルトは app.constants.align.center です。
nHorizValue	(オプション) ページ上で透かしの水平位置をシフトさせるために使用する数値。 bPercentage が true の場合、この数値はページの幅のパーセンテージを表します。 bPercentage が false の場合、この数値はオフセットするポイント数を表します。 デフォルトは 0 です。
nVertValue	(オプション) ページ上で透かしの垂直位置をシフトさせるために使用する数値。 bPercentage が true の場合、この数値はページの高さのパーセンテージを表します。 bPercentage が false の場合、この数値はオフセットするポイント数を表します。 デフォルトは 0 です。
bPercentage	(オプション) nHorizValue や nVertValue がページサイズのパーセンテージを表す のか、ポイント数を表すのかを示すブーリアン値。デフォルトは false です。
nScale	(オプション) 透かしに使用するスケール。1.0 が 100 %です。値が -1 の場合は、透 かし形状を維持したまま、大きさをページにフィットさせます。デフォルトは 1.0 で す。
bFixedPrint	(オプション) 透かしを FixedPrint Watermark 注釈として追加することを示すブーリア ン値。これによって、印刷するページのサイズに関係なく、透かしが固定のサイズおよ び位置に印刷されます。true の場合は、bOnTop は無視されます。デフォルトは false です。
nRotation	(オプション) 透かしを反時計回りに回転させる角度。デフォルトは 0 です。
nOpacity	(オプション) 透かしに使用する不透明度。0 が透明で、1.0 が不透明です。デフォル トは 1.0 です。

例 1

「Confidential」を透かしとして、現在の文書のすべてのページの中央に追加します。

```
this.addWatermarkFromText("Confidential", 0, font.Helv, 24, color.red);
```

例 2

現在の文書の各ページに複数の行からなる透かしを追加し、ページの右上隅から 1 インチ下および 1 インチ左に配置します。

```
this.addWatermarkFromText({  
    cText: "Confidential Document by A. C. Robot",  
    nTextAlign: app.constants.align.right,  
    nHorizAlign: app.constants.align.right,  
    nVertAlign: app.constants.align.top,  
    nHorizValue: -72, nVertValue: -72  
});
```

addWeblinks

5.0	D		X
-----	----------	--	---

指定のページをスキャンして `http:` スキームを使用したテキストのインスタンスを検出し、このインスタンスを URL アクションを伴うリンクに変換します。

[removeWeblinks](#) メソッドも参照してください。

パラメータ

nStart	(オプション) ページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) ページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しない場合は、文書のすべてのページが対象範囲になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。

戻り値

文書に追加された Web リンクの数。

例

文書全体を検索し、Web アドレスと判断されるすべてのコンテンツを Web リンクに変換します。また、作成したリンクの数を表示します。

```
var numWeblinks = this.addWeblinks();
console.println("There were " + numWeblinks +
    " instances of text that looked like a web address,"
    +" and converted as such.");
```

bringToFront

5.0			
-----	--	--	--

開かれている文書を手前に表示します。

例

この例では、開かれている文書の中から「Annual Report」というタイトルの文書を検索し、この文書を手前に表示します。

```
var d = app.activeDocs;
for (var i = 0; i < d.length; i++)
    if (d[i].info.Title == "Annual Report") d[i].bringToFront();
```

calculateNow

3.01			
------	--	--	--

現在の文書にあるすべての計算フィールドで強制的に計算を実行します。

大量の計算が使用されているフォームでは、フィールドへの入力を行うと、そのフィールドが計算フィールドでなくても、かなりの遅延が発生する場合があります。その解決策としては、ある時点で計算をオフにしておき、後でオンに戻す方法があります（例を参照）。

例

計算をオフにする

```
this.calculate = false;  
.....
```

計算をオンにする

```
this.calculate = true;
```

this.calculate を true にしても、ユーザがデータを確定しない限り、自動的に計算は行われません。強制的に計算を実行するには、次のコードを使用します。

```
this.calculateNow();
```

closeDoc

5.0			S
-----	--	--	----------

文書を閉じます。

Adobe Reader 5.1 以降では、このメソッドは常に許可されます。

- 文書が変更されており、文書の保存権限がない場合は、警告なしに文書が閉じられ、変更内容は失われます。
- 文書の保存権限がある場合は、変更されたファイルを保存するかどうかをユーザが選択できます。

このメソッドを実行して文書を閉じることにより、他に実行中の JavaScript プログラムに影響を与える可能性がありますので、このメソッドは慎重に使用してください。また、ページイベントや文書イベントでこのメソッドを使用すると、アプリケーションの動作が不安定になることがあります。

7.0 より前のバージョンの Acrobat では、文書で this.closeDoc を実行して、それ自身を閉じると、その後のスクリプトはすべて実行されません。Acrobat 7.0 では、スクリプトの実行が継続され、通常の方法で終了します。ただし、閉じた文書の Doc を参照した場合は、例外が発生します。

パラメータ

bNoSave	(オプション) 文書を保存せずに閉じるかどうかを示すブーリアン値。
	• <code>false</code> (デフォルト) の場合、文書が変更されていれば文書を保存するようプロンプトが表示されます。
	• <code>true</code> の場合は、文書が変更されていてもプロンプトは表示されず、文書は保存されずに閉じられます。ユーザの確認がないままデータが失われる可能性があるので、慎重に使用してください。

例 1

コンソールから、開いている文書をすべて閉じます。

```
var d = app.activeDocs;
for( var i in d ) d[i].closeDoc();
```

次のコードは、開いている文書で、「マウスボタンを放す」アクションとして実行できます。このコードは、開いている disclosed 状態の文書をすべて閉じます。このコードの実行が途中で終了しないように、アクティブな文書は最後に閉じています。

```
var d = app.activeDocs;
for( var i in d )
    if( d[i] != this ) d[i].closeDoc();
if ( this.disclosed ) this.closeDoc();
```

例 2

3つのテストファイルを作成し、ディレクトリに保存します。`saveAs` にはセキュリティ制限があるので、このコードはコンソールで実行する必要があります。

```
var myDoc = app.newDoc();
for (var i=0; i < 3; i++) {
    myDoc.info.Title = "Test File " + i;
    myDoc.saveAs("/c/temp/test"+i+".pdf");
}
myDoc.closeDoc(true);
```

`closeDoc` の別の例については、[saveAs](#) を参照してください。

colorConvertPage

8.0	D		P
-----	---	--	---

文書のページのカラーを変換します。

パラメータ

pageNum	変換する文書のページ番号を定義するインデックス（0 から数えます）。
actions	このカラー変換の colorConvertAction オブジェクトの配列。 colorConvertAction オブジェクトのプロパティについては、 194 ページ 以降を参照してください。 ページ上の各オブジェクトで、アクション配列内のアクションが順番に照合され、オブジェクトの属性やカラースペースと一致するかどうかが確認されます。一致した場合にはそのアクションが実行されます。このアクションリストは、多くの電子メールクライアントに搭載されているフィルタリストに似ています。各オブジェクトでアクションの選択条件が順番に比較され、一致するアクションが見つかると、オブジェクトに対してそのアクションが実行されます。インキ定義にエイリアスが設定されている場合を除いて、アクションが連鎖することはありません。
inkActions	このカラー変換のインキアクションを表す colorConvertAction オブジェクトの配列。このインキリストによって、それぞれの色分解に対するアクションを定義します。これは、Separation であっても DeviceN であっても対応できます。これを使用して、インキエイリアスの定義などを行うことができます。 DeviceN に、エイリアスを設定するインキと変換を行うインキが含まれている場合は、OPP テクノロジーを使用してカラー変換が行われます。その結果、変換が行われた部分はプロセスカラーになり、エイリアスが設定された部分はスポットカラーになります。 インキアクションの場合、match フィールドは無視されます。 変換内容を表すアクションリストでは、基盤となる Separation または DeviceN を定義する必要があります。基盤となるスペースのアクションリストで Preserve アクションまたは Decalibrate アクションが設定されている場合は、インキアクションリストのエイリアスが適用されます。

戻り値

ブーリアン値。ページが変更された場合は `true` が返され、それ以外の場合は `false` が返されます。

例

例については、[297 ページの getColorConvertAction](#) を参照してください。

createDataObject

5.0			
-----	--	--	--

Data オブジェクトを作成します。

Data オブジェクトは、必要に応じて作成することができます。これは、外部ファイル以外のソース（ADBC データベースから読み込んだデータなど）から JavaScript でデータを作成する場合に便利です。

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObject](#)、[openDataObject](#)、[importDataObject](#)、[removeDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#)、[Data](#) オブジェクトがあります。

パラメータ

cName	data オブジェクトに関連付ける名前。
cValue	埋め込むデータを表す文字列。
cMimeType	(オプション) データの MIME タイプ。デフォルトは「text/plain」です。
cCryptFilter	(オプション、Acrobat 6.0) 言語に依存しない、この data オブジェクトを暗号化する暗号フィルタの名前。暗号フィルタは、Doc の addRecipientListCryptFilter メソッドを使用して、文書の暗号フィルタのリストにあらかじめ追加しておく必要があります。追加されていない場合は、例外が発生します。data オブジェクトをファイル内で暗号化したくない場合は、定義済みの Identity 暗号フィルタを使用します。使用しない場合は、Doc の encryptForRecipients メソッドによって暗号化されます。

例

```
this.createDataObject("MyData.txt", "This is some data.");
```

[addRecipientListCryptFilter](#) の例も参照してください。

createTemplate



注意： Adobe Reader 5.1 以降では、拡張フォーム機能権限が付与されなければこのメソッドを実行できません。Adobe Reader 7.0 以降では、このメソッドは許可されず、NotAllowedError 例外が発生します。

指定のページから可視テンプレートを作成します。[templates](#) プロパティ、[getTemplate](#) メソッド、[removeTemplate](#) メソッド、[Template](#) オブジェクトも参照してください。

注意： このメソッドを実行できるのは、バッヂイベントまたはコンソールイベントのみです（[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照）。JavaScript イベントについて詳しくは、`event` オブジェクトを参照してください。

パラメータ

cName	ページに関連付ける名前。
nPage	(オプション) テンプレート化するページのインデックス（0 から数えます）。デフォルトは 0（文書の最初のページ）です。

戻り値

新しく作成された `Template` オブジェクト。

例

ページ 2 以降のすべてのページを非表示のテンプレートに変換します。テンプレートを非表示にしているので、変換を行うたびに `this.numPages` (表示されているページ数) は変化します。次のループでは、ページ 2 のみをテンプレートにして非表示にしている点に注意してください。これを行うと、その次のページが新しいページ 2 になります。

```
numNewTemplates = this.numPages - 2;  
for ( var i = 0; i < numNewTemplates; i++ )  
{  
    var t = this.createTemplate({cName:"myTemplate"+i, nPage:2});  
    t.hidden = true;  
}
```

deletePages

5.0			
-----	--	--	--

文書からページを削除します。対象のページを指定しない場合は、最初のページ（ページ 0）が削除されます。[insertPages](#)、[extractPages](#)、[replacePages](#) も参照してください。

注意：文書のすべてのページを削除することはできません。少なくとも 1 ページは残しておく必要があります。

(Acrobat 6.0) バージョン 6.0 以降では、文書にフォーム使用権限が追加されいれば、テンプレートから生成されたページを Adobe Reader でこのメソッドを使用して削除できます。

パラメータ

<code>nStart</code>	(オプション) 削除するページ範囲の最初のページのインデックス (0 から数えます)。 デフォルトは 0 (文書の最初のページ) です。
<code>nEnd</code>	(オプション) 削除するページ範囲の最後のページ。 <code>nEnd</code> を指定しなかった場合は、 <code>nStart</code> で指定したページのみが削除されます。

例

ページ 1 から 3 (0 から数えます) を削除します。

```
this.deletePages({nStart: 1, nEnd: 3});
```

deleteSound

5.0			
-----	--	--	--

指定の名前の `sound` オブジェクトを文書から削除します。

[sounds](#)、[getSound](#)、[importSound](#)、[Sound](#) オブジェクトも参照してください。

パラメータ

<code>cName</code>	削除する Sound オブジェクトの名前。
--------------------	-----------------------

例

```
this.deleteSound("Moo");
```

embedDocAsDataObject

7.0			D
-----	--	--	---

指定した文書を Data オブジェクトとして文書に埋め込みます。

注意： Adobe Reader 7.0 以降では、文書にファイル添付権限が付与されていれば、このメソッドを実行できます。ただし、変更が加えられている文書を埋め込むには、その文書に文書の保存権限が付与されている必要があります。

パラメータ

cName	data オブジェクトに関連付ける名前。
oDoc	data オブジェクトとして埋め込む文書。
cCryptFilter	(オプション) 言語に依存しない、暗号フィルタの名前。この data オブジェクトを暗号化する際に利用します。暗号フィルタは、addRecipientListCryptFilter メソッドを使用して、文書の暗号フィルタのリストにあらかじめ追加しておく必要があります。追加されていない場合は、例外が発生します。data オブジェクトをファイル内で暗号化たくない場合は、定義済みの Identity 暗号フィルタを使用します。使用しない場合は、encryptForRecipients メソッドによって暗号化されます。
bUI	(オプション) true の場合、oDoc の保存が必要であるにも関わらず保存権限が付与されないと、警告が表示されます。デフォルト値は false です。

例

「myFilter」という暗号フィルタが含まれている envelope ファイルがあらかじめ作成されており、現在の文書に含まれています。

```
var authorEmail = "johndoe@example.com";
var envelopeDoc = this.openDataObject( "envelope" );
envelopeDoc.embedDocAsDataObject( "attachment", this, "myFilter" );
envelopeDoc.title.Author = authorEmail;
envelopeDoc.mailDoc({
  cTo: "support@example.com",
  cSubject: "Application from " + authorEmail
});
```

embedOutputIntent

8.0	D		P
-----	---	--	---

カラープロファイルを PDF/X の出力インテントとして埋め込みます（『PDF Reference』バージョン 1.7 を参照）。

パラメータ

outputIntentColorSpace	出力インテントに使用するプロファイルを表す文字列。使用可能なカラープロファイルのリストは、app オブジェクトの printColorProfiles プロパティ (93 ページ を参照) から取得できます。
------------------------	---

例

カラープロファイルを埋め込みます。

```
this.embedOutputIntent ("U.S. Sheetfed Coated v2")
```

encryptForRecipients

6.0			
-----	--	--	--

指定のリスト内の受信者に対して、各受信者の公開鍵証明書を使用して、文書を暗号化します。暗号化は文書が保存されるまで実行されません。受信者をグループとしてまとめ、各グループに固有の権限設定を指定できます。このメソッドでは、失敗すると例外が発生します。

注意：このメソッドを使用できるのは、バッチイベント、コンソールイベント、アプリケーション初期化イベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

[createDataObject](#) メソッド、[security.chooseRecipientsDialog](#) メソッド、[Data](#) オブジェクトも参照してください。

パラメータ

oGroups	文書暗号化の対象となる受信者をリストした汎用 Group オブジェクトの配列。
bMetaData	(オプション) <code>true</code> (デフォルト) の場合、文書のメタデータが暗号化されます。この値を <code>false</code> に設定すると、Acrobat 6.0 以降でのみ表示できる文書が生成されます。
bUI	(オプション) <code>true</code> の場合、ハンドラによって表示されるユーザインターフェイスを使用して、文書暗号化の対象となる受信者を選択できます。デフォルト値は <code>false</code> です。

戻り値

成功した場合は `true` を返し、失敗した場合は例外が発生します。

Group オブジェクト

文書またはデータ暗号化の対象となる受信者のリストに一連の権限を付与するための汎用 JavaScript オブジェクト。このオブジェクトは `encryptForRecipients` の入力パラメータになったり、`security.chooseRecipientsDialog` の戻り値になったりします。このオブジェクトのプロパティは、次のとおりです。

プロパティ	説明
permissions	グループの権限を含む <code>Permissions</code> オブジェクト。
userEntities	権限の適用対象となるユーザを表す <code>UserEntity</code> オブジェクトの配列。

Permissions オブジェクト

Group オブジェクトで使用する、一連の権限を含む汎用 JavaScript オブジェクト。このオブジェクトのプロパティは、次のとおりです。ブーリアンプロパティのデフォルト値は、すべて `false` です。

プロパティ	型	アクセス	説明
allowAll	ブーリアン	R / W	制限のないアクセスを許可するかどうかを示します。 <code>true</code> の場合、他のすべてのプロパティよりも優先されます。
allowAccessibility	ブーリアン	R / W	視覚障害のあるユーザ用のコンテンツアクセスを許可するかどうかを示します。 <code>true</code> の場合、テキスト音読アプリケーションなどに使用する内容の抽出が許可されます。
allowContentExtraction	ブーリアン	R / W	内容のコピーや抽出を許可するかどうかを示します。
allowChanges	文字列	R / W	文書に対して許可される変更を示します。有効な値は、次のとおりです。 none documentAssembly fillAndSign editNotesFillAndSign all
allowPrinting	文字列	R / W	文書に対して許可される印刷セキュリティレベルを示します。有効な値は、次のとおりです。 none lowQuality highQuality

例

文書内のすべての文字列とストリームを暗号化します。これにより、Acrobat 5.0 以降で開くことができるファイルが生成されます。

```
var sh = security.getHandler( "Adobe.PPKMS" );
var dir = sh.directories[0];
var dc = dir.connect();

dc.setOutputFields({oFields:["certificates"]});
var importantUsers = dc.search({oParams:{lastName:"Smith"}});
var otherUsers = dc.search({oParams: {lastName:"Jones" }});

this.encryptForRecipients({
  oGroups :
  [
    {userEntities:importantUsers,permissions:{allowAll:true }},
    {userEntities: otherUsers, permissions:{allowPrinting:"highQuality"}}
  ],
  bMetaData : true
});
```

encryptUsingPolicy

7.0			
-----	--	--	--

指定したポリシーオブジェクトとハンドラを使用して文書を暗号化します。このメソッドはユーザの操作が必要な場合があり、新規のセキュリティポリシーが作成される場合があります。

注意：このメソッドを実行できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

パラメータ

<code>oPolicy</code>	文書を暗号化する際に使用するポリシーオブジェクト。 <code>SecurityPolicy</code> オブジェクトは、 <code>chooseSecurityPolicy</code> または <code>getSecurityPolicies</code> から返されるオブジェクトです。 このパラメータは、 <code>policyId</code> プロパティが定義されている汎用オブジェクトです。定義済みのポリシー ID を渡すと、関連するポリシーが取得されて使用されます。渡したポリシー ID が未知の場合は、エラーが返されます。 注意： この特別なポリシー ID を使用したときに <code>oGroups</code> が <code>null</code> の場合は、エラーが返されます。
<code>oGroups</code>	(オプション) ポリシーを適用するときにハンドラが使用する <code>Group</code> オブジェクトの配列。使用するポリシーと関連するハンドラによって、動作は異なります。 <code>Group</code> オブジェクトには許可の情報が埋め込まれていることがあります。その情報が使用されるかどうかは、ポリシーや関連するセキュリティハンドラによります。デフォルト値は <code>null</code> です。
<code>oHandler</code>	(オプション) 暗号化に使用する <code>SecurityHandler</code> オブジェクト。このハンドラが <code>oPolicy</code> オブジェクトで指定したハンドラ名と一致しない場合は、メソッドの実行に失敗します。このパラメータを指定しなかった場合は、ハンドラに関連付けられているデフォルトの <code>SecurityHandler</code> オブジェクトが使用されます。 APS セキュリティハンドラを使用する場合は、あらかじめ新しい <code>SecurityHandler</code> を作成して、 <code>login</code> を行うことで、Acrobat に設定されていないサーバに対して認証を行い、その <code>SecurityHandler</code> を <code>oHandler</code> に渡すことができます。これによって、Acrobat に接続設定されていないサーバで定義されているポリシーを使用することができます。 PPKLite セキュリティハンドラを使用する場合は、あらかじめ新しい <code>SecurityHandler</code> を作成して、 <code>login</code> を行うことで、Acrobat に組み込まれていないデジタル ID ファイルを開き、その <code>SecurityHandler</code> を <code>oHandler</code> に渡すことができます。これによって、Acrobat には設定されていないデジタル ID ファイルに含まれている証明書を使用することができます。
<code>bUI</code>	(オプション) <code>true</code> の場合は、(認証などのために) ユーザインターフェイスが表示されることがあります。 <code>false</code> の場合は、ユーザインターフェイスは表示されません。ユーザの操作が必要であるにも関わらず、許可されないとときは、エラーが返されます。デフォルト値は <code>false</code> です。

戻り値

戻り値は、次のプロパティを持つ `SecurityPolicyResults` オブジェクトです。

プロパティ	型	説明
errorCode	整数	<p>ポリシーを実装するハンドラから返されるエラーコード。これには 3 つのエラーがあります。</p> <p>0 = Success</p> <p><code>errorText</code> は定義されません。 <code>unknownRecipients</code> は定義される場合があります。 <code>policyApplied</code> は定義されます。</p> <p>1 = Failure</p> <p><code>errorText</code> は定義されます。 <code>unknownRecipients</code> は定義される場合があります。 <code>policyApplied</code> は定義されません。</p> <p>2 = Abort (ユーザによるプロセスの中止)</p> <p><code>errorText</code> は定義されません。 <code>unknownRecipients</code> は定義されません。 <code>policyApplied</code> は定義されません。</p>
errorText	文字列	ローカライズされたエラーの説明（このプロパティの値は定義されない場合があります）。エラーについて詳しくは、 <code>errorCode</code> を参照してください。
policyApplied	オブジェクト	適用された <code>SecurityPolicy</code> オブジェクト（このプロパティの値は定義されない場合があります）。「 <code>adobe_secure_for_recipients</code> 」ポリシーを渡した場合は、呼び出しによって新規のポリシーが作成され、対応するポリシーオブジェクトがここで返されます。エラーについて詳しくは、 <code>errorCode</code> を参照してください。
unknownRecipients	Recipients オブジェクト	暗号化の対象として指定した受信者のうち、ポリシーを適用する際に使用できなかった受信者（このプロパティの値は定義されない場合があります）。エラーについて詳しくは、 <code>errorCode</code> を参照してください。

例 1

選択したポリシーを使用して、新規に作成した文書を暗号化します。

```
var doc = app.newDoc();
var policy = security.chooseSecurityPolicy();
var results = doc.encryptUsingPolicy( { oPolicy: policy } );
if ( results.errorCode == 0)
    console.println("The policy applied was: " + results.policyApplied.name);
```

例 2

テンプレートポリシーを使用して、新規に作成した文書を暗号化します（この例を実行する前に、LiveCycle Policy Server の発行設定を行なう必要があります）。

```
var doc = app.newDoc();
var groups = [ { userEntities: [{email:"jdoe@example.com"}, {email:"bsmith@example.com"}] }
];
```

```
var policy = { policyId: "adobe_secure_for_recipients" };
var results = doc.encryptUsingPolicy({
    oPolicy: policy,
    oGroups: groups,
    bUI: true
});
if ( results.errorCode == 0 )
    console.println("The policy applied was: "
        + results.policyApplied.name);
```

exportAsPDF

4.0			
-----	--	--	--

フォームフィールドのデータを FDF ファイルとしてローカルハードドライブに書き出します。

注意： cPath パラメータを指定した場合、このメソッドを実行できるのは、バッチイベントとコンソールイベントのみになります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

bAllFields	(オプション) true の場合は、値がないフィールドも含め、すべてのフィールドが書き出されます。false (デフォルト) の場合は、現在値のないフィールドが除外されます。
bNoPassword	(オプション) true (デフォルト) の場合、パスワードフラグが設定されたテキストフィールドは FDF ファイルに書き出されません。
aFields	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">これを指定した場合は、指定のフィールドのみが書き出されます (bNoPassword で除外したフィールドは除かれます)。aFields が空の配列である場合、フィールドは書き出されません。ただし、この場合でも、bAnnotations パラメータの値によっては FDF ファイルにデータが含まれることがあります。このパラメータを省略するか、または値が null の場合は、フォーム内のすべてのフィールドが書き出されます (bNoPassword で除外したフィールドは除かれます)。 フィールドのサブツリー全体を書き出すには、親のフィールド名を指定します (次の例を参照)。
bFlags	(オプション) true の場合、書き出される FDF ファイルにフィールドフラグが含められます。デフォルトは false です。
cPath	(オプション) デバイスに依存しない、ファイルのパスを表す文字列。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。
	<p>注意： パラメータ cPath は、セーフパスであり (31 ページの「セーフパス」 を参照)、拡張子が .fdf であることが必要です。これらのセキュリティ条件が満たされていない場合は、NotAllowedError という例外 (Error オブジェクトを参照) が発生してメソッドが失敗します。</p>
bAnnotations	(オプション、Acrobat 6.0) true の場合、書き出される FDF ファイルに注釈が含められます。デフォルトは false です。

例 1

フォーム全体（空のフィールドを含む）をフラグとともに書き出します。

```
this.exportAsFDF(true, true, null, true);
```

例 2

name サブツリーをフラグなしで書き出します。

```
this.exportAsFDF(false, true, "name");
```

この例は、サブツリー全体を簡単に書き出す方法を示しています。aFields パラメータの一部として「name」を渡すと、「name.title」、「name.first」、「name.middle」、「name.last」などのフィールドが書き出されます。

exportAsFDFStr

8.0			
-----	--	--	--

doc.exportAsFDF メソッドを呼び出した場合と同じです。ただし、結果はファイルに保存されずに文字列として返されます。

パラメータ

bAllFields	(オプション) true の場合は、値がないフィールドも含め、すべてのフィールドが書き出されます。false（デフォルト）の場合は、現在値のないフィールドが除外されます。
bNoPassword	(オプション) true（デフォルト）の場合、パスワードフラグが設定されたテキストフィールドは FDF ファイルに書き出されません。
aFields	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">これを指定した場合は、指定のフィールドのみが書き出されます（bNoPassword で除外したフィールドは除かれます）。aFields が空の配列である場合、フィールドは書き出されません。ただし、この場合でも、bAnnotations パラメータの値によっては FDF ファイルにデータが含まれることがあります。このパラメータを省略するか、または値が null の場合は、フォーム内のすべてのフィールドが書き出されます（bNoPassword で除外したフィールドは除かれます）。 フィールドのサブツリー全体を書き出すには、親のフィールド名を指定します（次の例を参照）。
bFlags	(オプション) true の場合、書き出される FDF ファイルにフィールドフラグが含められます。デフォルトは false です。
bAnnotations	false（デフォルト）であることが必要です。注釈はサポートされていません。
cHRef	指定すると、返される FDF 文字列のソースまたはターゲットファイルとして（FDF ディクショナリの F キーの値として）この値が挿入されます。

戻り値

`doc.exportAsFDF` メソッドで生成されるファイルの内容が、文字列として返されます。`cHRef` パラメータを指定した場合は、そのパラメータの値が **FDF** ディクショナリの **F** キーの値として挿入されます。指定しなかった場合、**F** キーの値は、`doc.exportAsFDF` で生成される値になります。

例

`FirstName`、`LastName`、`Address` の各フィールドのフォームデータを、FDF 形式の文字列として取得します。

```
var cFDF = this.exportAsFDFStr({
    aFields: ["FirstName", "LastName", "Address"],
    cHRef: "http://www.example.com/formcatalog/ThisFormName.pdf"
});
```

exportAsText

6.0			
-----	--	--	--

フォームフィールドのデータをタブ区切りテキストファイルとしてローカルハードディスクに書き出します。作成されるテキストファイルは、Microsoft Excel でのテキストファイル形式に従います。`exportAsText` は、引用符や複数行にわたるテキストフィールドを適切に処理できます。

このメソッドは 2 行のテキストファイルを書き出します。最初の行は、`aFields` で指定したフィールド名のタブ区切りリストです。次の行は、フィールド値のタブ区切りリストです。

注意：`cPath` パラメータを指定した場合、このメソッドを実行できるのは、バッチイベントとコンソールイベントのみになります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

<code>bNoPassword</code>	(オプション) <code>true</code> (デフォルト) の場合、パスワードフラグが設定されたテキストフィールドはテキストファイルに書き出されません。
<code>aFields</code>	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">これを指定した場合は、指定のフィールドのみが書き出されます (<code>bNoPassword</code> で除外したフィールドは除かれます)。<code>aFields</code> が空の配列である場合、フィールドは書き出されません。このパラメータを省略するか、または値が <code>null</code> の場合は、フォーム内のすべてのフィールドが書き出されます (<code>bNoPassword</code> で除外したフィールドは除かれます)。
<code>cPath</code>	(オプション) デバイスに依存しない、ファイルのパスを表す文字列。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。

注意： パラメータ `cPath` は、セーフパスであり ([31 ページの「セーフパス」](#) を参照)、拡張子が `.txt` であることが必要です。これらのセキュリティ条件が満たされていない場合は、`NotAllowedError` という例外 ([Error](#) オブジェクトを参照) が発生してメソッドが失敗します。

例

すべてのフィールドをタブ区切りのファイルに書き出すには、コンソールで次のスクリプトを実行します。

```
this.exportAsText();
```

複数のデータ行からなるタブ区切りのファイルを作成するには、[298 ページの「例」](#) を参照してください。

exportAsXFDF



フォームフィールドのデータを XFDF ファイルとしてローカルハードドライブに書き出します。

XFDF は、Acrobat フォームデータを XML で表現したものです。『XML Form Data Format Specification』を参照してください ([28 ページの「関連ドキュメント」](#) を参照)。

インポートを行うメソッド `importAnXFDF` も用意されています。

注意： `cPath` パラメータを指定した場合、このメソッドを実行できるのは、バッチイベントとコンソールイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

<code>bAllFields</code>	(オプション) <code>true</code> の場合は、値がないフィールドも含め、すべてのフィールドが書き出されます。 <code>false</code> (デフォルト) の場合は、現在値のないフィールドが除外されます。
<code>bNoPassword</code>	(オプション) <code>true</code> (デフォルト) の場合、パスワードフラグが設定されたテキストフィールドは XFDF に書き出されません。
<code>aFields</code>	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">これを指定した場合は、指定のフィールドのみが書き出されます (<code>bNoPassword</code> で除外したフィールドは除かれます)。<code>aFields</code> が空の配列である場合、フィールドは書き出されません。ただし、この場合でも、<code>bAnnotations</code> パラメータの値によっては XFDF ファイルにデータが含まれることがあります。このパラメータを省略するか、または値が <code>null</code> の場合は、フォーム内のすべてのフィールドが書き出されます (<code>bNoPassword</code> で除外したフィールドは除かれます)。 フィールドのサブツリー全体を書き出すには、親のフィールド名を指定します。
<code>cPath</code>	(オプション) デバイスに依存しない、ファイルのパスを表す文字列。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。
<code>bAnnotations</code>	(オプション、Acrobat 6.0) <code>true</code> の場合、書き出される XFDF ファイルに注釈が含められます。デフォルトは <code>false</code> です。

exportAsXFDFStr

8.0			
-----	--	--	--

doc.exportAsXFDF メソッドを呼び出した場合と同じです。ただし、結果はファイルに保存されずに文字列として返されます。

パラメータ

bAllFields	(オプション) true の場合は、値がないフィールドも含め、すべてのフィールドが書き出されます。false (デフォルト) の場合は、現在値のないフィールドが除外されます。
bNoPassword	(オプション) true (デフォルト) の場合、パスワードフラグが設定されたテキストフィールドは XFDF ファイルに書き出されません。
aFields	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">これを指定した場合は、指定のフィールドのみが書き出されます (bNoPassword で除外したフィールドは除かれます)。aFields が空の配列である場合、フィールドは書き出されません。ただし、この場合でも、bAnnotations パラメータの値によっては XFDF ファイルにデータが含まれることがあります。このパラメータを省略するか、または値が null の場合は、フォーム内のすべてのフィールドが書き出されます (bNoPassword で除外したフィールドは除かれます)。 フィールドのサブツリー全体を書き出すには、親のフィールド名を指定します。
bAnnotations	false (デフォルト) であることが必要です。注釈はサポートされていません。
cHRef	指定すると、返される XFDF 文字列のソースまたはターゲットファイルとして (xfdf 要素の f 子要素の href 属性の値として) この値が挿入されます。

戻り値

doc.exportAsXFDF メソッドで生成されるファイルの内容が、文字列として返されます。cHRef パラメータを指定した場合は、そのパラメータの値が xfdf 要素の f 子要素の href 属性の値として挿入されます。指定しなかった場合、f 要素キーの href 属性の値は、doc.exportAsXFDF で生成される値になります。

例

FirstName、LastName、Address の各フォームフィールドの値を、XFDF 形式の文字列として取得します。

```
var cXFDF = this.exportAsXFDFStr({
    aFields: ["FirstName", "LastName", "Address"],
    cHRef: "http://www.example.com/formcatalog/ThisFormName.pdf"
});
```

exportDataObject

5.0			
-----	--	---	--

指定の data オブジェクトを外部ファイルに書き出します。

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[openDataObject](#)、[createDataObject](#)、[removeDataObject](#)、[importDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#)、[Data](#) オブジェクトがあります。

注意：Acrobat 6.0 以降では、パラメータ cDIPath が null でない場合は、NotAllowedError という例外 ([Error](#) オブジェクトを参照) が発生してメソッドが失敗します。

このメソッドに cDIPath を渡さなかった場合は、ファイル選択ダイアログボックスが開いて、埋め込まれている data オブジェクトの保存パスを選択できるようになります。

パラメータ

cName	抽出する data オブジェクトの名前。
cDIPath	(オプション) デバイスに依存しない、data オブジェクトの抽出先となるパス。このパスには、絶対パスまたは現在の文書に対する相対パスのどちらでも指定できます。指定しなかった場合は、保存場所を指定するように求めるプロンプトが表示されます。 注意： (Acrobat 6.0) このパラメータの使用はサポートされていません。このパラメータを使用しないでください。前述のセキュリティに関する注意を参照してください。
bAllowAuth	(オプション、Acrobat 6.0) true の場合、ダイアログボックスを使用してユーザ認証を行います。encryptForRecipients メソッドを使用して data オブジェクトが暗号化されている場合は、認証が必要になることがあります。認証ダイアログボックスは bAllowAuth が true の場合に使用できます。デフォルト値は false です。
nLaunch	(オプション、Acrobat 6.0) nLaunch は、保存後にファイルを起動する（開く）かどうかを制御します。ファイルが PDF ファイルでない場合は、起動時に外部アプリケーションが開くことがあります。nLaunch の値は、次のとおりです。 <ul style="list-style-type: none">値が 0 の場合、保存後にファイルは起動しません。値が 1 の場合、保存後にファイルが起動します。ファイルが PDF ファイルでない場合は、起動時にセキュリティ警告が表示されます。保存パスを指定するプロンプトが表示されます。値が 2 の場合、保存後にファイルが起動します。ファイルが PDF ファイルでない場合は、起動時にセキュリティ警告が表示されます。一時パスが使用され、保存パスを指定するプロンプトは表示されません。作成された一時ファイルは、アプリケーションのシャットダウン時に Acrobat によって削除されます。 デフォルト値は 0 です。

例 1

抽出先のパスをユーザに尋ねます。

```
this.exportDataObject ("MyData");
```

例 2 (Acrobat 6.0)

PDF 文書を抽出してビューアで開きます。

```
this.exportDataObject({ cName: "MyPDF.pdf", nLaunch: 2 });
```

例 3

`importDataObject` メソッドを使用して添付ファイルが取り込まれている場合は、このメソッドの `cName` パラメータによって `Data.name` プロパティの値が割り当てられています。しかし、UI を使用してファイルが添付されている場合、`name` は自動的に割り当てられます。添付ファイルには、「Untitled Object」、「Untitled Object 2」、「Untitled Object 3」などのように、番号の付いた名前が割り当てられます。

UI を使用して添付されたファイルを書き出すには、その添付ファイルの `name` を調べる必要があります。次のコードでは、UI を使用して最後に添付されたファイルを書き出します（ファイルが存在する場合）。

```
var d = this.dataObjects;
if ( d == null ) console.println("No file attachments");
else {
    for ( var i = d.length - 1; i>=0; i-- )
        if ( d[i].name.indexOf("Untitled Object") != -1 ) break;
    if ( i != -1 ) this.exportDataObject(d[i].name);
    else console.println("No attachment was embedded by UI");
}
```

exportXFAData

6.0			
-----	--	--	--

文書から XFA データを書き出し（存在する場合）、XDP ファイルとして保存します。

注意： Adobe Reader から XFA データを書き出す場合は、フォームの書き出し権限が文書に付与されていることが必要です。

`cPath` パラメータを指定した場合、このメソッドを実行できるのは、バッヂイベント、コンソールイベント、メニューイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

`cPath` (オプション) デバイスに依存しない、ファイルのパス。このパスには、文書に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。

パスは次の条件を満たす必要があります。

- セーフパスであることが必要です ([31 ページの「セーフパス」](#) を参照)。
- `bXDP` が `true` の場合は、ファイル名の拡張子が `.xdp` であることが必要です。
- `bXDP` が `false` の場合は、ファイル名の拡張子が `.xml` であることが必要です。

これらの条件が満たされていない場合は、`NotAllowedError` という例外 ([Error](#) オブジェクトを参照) が発生します。

bXDP	(オプション) <code>true</code> (デフォルト) の場合、データは XDP 形式で書き出されます。 <code>false</code> の場合は、標準 XML データ形式で書き出されます。
aPackets	(オプション) XDP の書き出しに含める情報を指定する文字列の配列。このパラメータが有効なのは、 <code>bXDP</code> が <code>true</code> の場合のみです。 有効な文字列は、次のとおりです。 <pre>template datasets stylesheet xfdf sourceSet pdf config *</pre> <p><code>pdf</code> を指定した場合は、PDF ファイルが埋め込まれます。それ以外の場合は、PDF ファイルへのリンクのみが XDP ファイルに含められます。</p> <p><code>xfdf</code> を指定した場合は、注釈が XDP ファイルに含められます（注釈の情報は XFDF 形式であるため）。</p> <p>* を指定した場合は、すべての情報が XDP ファイルに含められます。ただし、<code>pdf</code> の情報は、デフォルトで参照として含められます。PDF ファイルを XDP ファイルに埋め込むには、明示的に <code>pdf</code> を指定します。</p> <p>注意：（保存権限が必要）Adobe Reader から XDP 形式で書き出す場合、文書の保存権限が文書に付与されている必要があるのは、<code>pdf</code> を明示的にリストした場合のみです。</p> <p>このパラメータのデフォルトは、<code>["datasets", "xfdf"]</code> です。</p>

例

XFA データを書き出します。次の例では、すべての情報が含まれられます。ただし、PDF 文書は参照のみで埋め込まれません。

```
this.exportXFADe({
  cPath: "/c/temp/myData.xdp",
  bXDP: true,
  aPackets: ["*"]
})
```

次の例では、すべての情報が含まれられ、PDF 文書が XDP ファイルに埋め込まれます。

```
this.exportXFADe({
  cPath: "/c/temp/myData.xdp",
  bXDP: true,
  aPackets: ["*", "pdf"]
})
```

extractPages

5.0	D	S	X
-----	---	---	---

現在の文書からページを抽出し、抽出したページで新しい文書を作成します。ページ範囲を指定しない場合は、文書の全ページが抽出されます。

[deletePages](#)、[insertPages](#)、[replacePages](#) も参照してください。

注意： cPath パラメータを指定した場合、このメソッドを実行できるのは、バッチイベント、コンソールイベント、外部呼び出し（OLE など）のみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

nStart	(オプション) ソース文書から抽出するページ範囲の開始を定義するインデックス（0 から数えます）。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) ソース文書から抽出するページ範囲の終了を定義するインデックス（0 から数えます）。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
cPath	(オプション) デバイスに依存しない、新しい文書の保存先となるパス。このパス名には、現在の文書の位置に対する相対パスも指定できます。 注意： パラメータ cPath は、セーフパスであり (31 ページの「セーフパス」 を参照)、拡張子が .pdf であることが必要です。これらのセキュリティ条件が満たされていない場合は、NotAllowedError という例外 (Error オブジェクトを参照) が発生してメソッドが失敗します。

戻り値

cPath を指定しなかった場合は、新しい文書の Doc が返されます。指定した場合は、null オブジェクトが返されます。

例

次のバッチシーケンスでは、ファイルごとに各ページを抽出し、それを一意の名前でフォルダに保存することができます。これは、クライアントの請求書がページ別に作成されている場合などに使用できます。このような場合、請求書の送付や印刷をするためには、PDF 文書をページごとに切り分けられると便利です。

```
/* ページをフォルダに抽出 */
// ファイルのベース名を取得するために正規表現を使用
var re = /\$.pdf$/i;
// ファイル名は、Acrobat が処理しているファイルのベース名
var filename = this.documentFileName.replace(re, "");
try {for (var i = 0; i < this.numPages; i++)
    this.extractPages({
        nStart: i,
        cPath: "/F/temp/" + filename + "_" + i + ".pdf"
    });
} catch (e) { console.println("Aborted: " + e) }
```

flattenPages

5.0	D		X
-----	---	--	---

ページ範囲にあるすべての注釈をページコンテンツに変換します。ページ範囲を指定しない場合は、文書のすべての注釈が変換されます。

注意：このメソッドの使用には十分注意してください。指定のページ範囲に存在するフォームフィールド、コメント、リンクを含むすべての注釈がページコンテンツに同化します。外観は残されますが、注釈ではなくなります。

パラメータ

nStart	(オプション) 現在の文書のページ範囲の開始を定義するインデックス（0 から数えます）。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 現在の文書のページ範囲の終了を定義するインデックス（0 から数えます）。
nNonPrint	(オプション、Acrobat 6.0) このパラメータは、印刷されない注釈の処理方法を設定します。有効な値は、次のとおりです。 0 — (デフォルト) 印刷されない注釈もページコンテンツに同化します。 1 — 印刷されない注釈は元の状態のまま残します。 2 — 印刷されない注釈は文書から削除します。

例

文書の全ページの注釈をページコンテンツに同化します。

```
this.flattenPages();
```

getAnnot

5.0			
-----	--	--	--

指定の文書ページに含まれている Annotation オブジェクトを返します。

パラメータ

nPage	Annotation オブジェクトが含まれているページ。
cName	Annotation オブジェクトの名前。

戻り値

Annotation オブジェクト。指定の注釈が存在しない場合は null。

例

特定の注釈を取得します。

```
var ann = this.getAnnot(0, "OnMarketShare");
if (ann == null)
    console.println("Not Found!")
else
    console.println("Found it! type: " + ann.type);
```

getAnnot3D

7.0			
-----	--	--	--

指定のページにある指定の名前の Annot3D オブジェクトを取得します。

パラメータ

nPage	Annot3D オブジェクトが含まれているページの番号（0 から数えます）。
cName	Annot3D オブジェクトの名前。

戻り値

Annot3D オブジェクト。指定の注釈が存在しない場合は `undefined`。

getAnnots

5.0			
-----	--	--	--

指定の条件を満たす Annotation オブジェクトの配列を取得します。[getAnnot](#) および [syncAnnotScan](#) も参照してください。

パラメータ

nPage	(オプション) ページ番号（0 から数えます）。これを指定した場合は、指定のページにある注釈のみが取得されます。指定しなかった場合は、検索条件を満たす注釈が全ページから取得されます。
nSortBy	(オプション) 配列のソート方法。有効な値は、次のとおりです。 ANSB_None — (デフォルト) ソートしません。このパラメータを指定しない場合と同じです。 ANSB_Page — ソートのプライマリキーとしてページ番号を使用します。 ANSB_Author — ソートのプライマリキーとして作成者を使用します。 ANSB_ModDate — ソートのプライマリキーとして更新日を使用します。 ANSB_Type — ソートのプライマリキーとして注釈タイプを使用します。

bReverse	(オプション) <code>true</code> の場合、配列のソート順が <code>nSortBy</code> の降順になります。
nFilterBy	(オプション) 特定の条件を満たす注釈のみを取得します。有効な値は、次のとおりです。 <code>ANFB_ShouldNone</code> — (デフォルト) すべての注釈を取得します。このパラメータを指定しない場合と同じです。 <code>ANFB_ShouldPrint</code> — 印刷可能な注釈のみを対象にします。 <code>ANFB_ShouldView</code> — 表示可能な注釈のみを対象にします。 <code>ANFB_ShouldEdit</code> — 編集可能な注釈のみを対象にします。 <code>ANFB_ShouldAppearInPanel</code> — 「注釈」タブに表示される注釈のみを対象にします。 <code>ANFB_ShouldSummarize</code> — 注釈の一覧に表示される注釈のみを対象にします。 <code>ANFB_ShouldExport</code> — 書き出し可能な注釈のみを対象にします。

戻り値

`Annotation` オブジェクトの配列。見つからない場合は `null`。

例

最初のページのすべての注釈を取得し、その情報をコンソールに書き込みます。

```
this.syncAnnotScan();
var annots = this.getAnnots({
    nPage: 0,
    nSortBy: ANSB_Author,
    bReverse: true
});
console.show();
console.println("Number of Annotations: " + annots.length);
var msg = "%s in a %s annot said: %s";
for (var i = 0; i < annots.length; i++)
    console.println(util.printf(msg, annots[i].author, annots[i].type,
        annots[i].contents));
```

getAnnots3D

7.0			
-----	--	--	--

指定のページの `Annot3D` オブジェクトの配列を返します。

パラメータ

nPge	<code>Annot3D</code> オブジェクトが含まれているページの番号 (0 から数えます)。
------	--

戻り値

Annot3D オブジェクトの配列。見つからない場合は undefined。

getColorConvertAction

8.0			P
-----	--	--	---

デフォルトのカラー変換設定を表す [colorConvertAction](#) オブジェクトを取得します。

[colorConvertPage](#) を参照してください。このメソッドは、colorConvertAction オブジェクトの配列をパラメータとして 2つ取ります。

戻り値

colorConvertAction オブジェクト。

例

colorConvertAction オブジェクトを取得し、すべての色を RGB に変換するように設定します（代替スペースは変換しないので、「スペースタイプ」の一致では代替スペースを除いています）。

```
// カラー変換アクションを取得
var toRGB = this.getColorConvertAction();

// RGB への変換アクションを設定
toRGB.matchAttributesAny = -1;
toRGB.matchSpaceTypeAny = ~toRGB.constants.spaceFlagsAlternateSpace;
toRGB.matchIntent = toRGB.constants.renderingIntentsAny;
toRGB.convertProfile = "Apple RGB";
toRGB.convertIntent = toRGB.constants.renderingIntentsDocument;
toRGB.embed = true;
toRGB.preserveBlack = false;
toRGB.useBlackPointCompensation = true;
toRGB.action = toRGB.constants.actionsConvert;

// 文書の最初のページを変換
var result = this.colorConvertPage(0, [toRGB], []);
```

getDataObject

5.0			
-----	--	--	--

特定の Data オブジェクトを取得します。[dataObjects](#)、[createDataObject](#)、[exportDataObject](#)、[importDataObject](#)、[removeDataObject](#) も参照してください。

パラメータ

cName 取得する data オブジェクトの名前。

戻り値

指定した名前に対応する Data オブジェクト。

例

特定の添付ファイルを取得して、各種の情報をコンソールに書き込みます。

```
var MyData = this.getDataObject("MyData");
console.show(); console.clear();
for (var i in MyData) console.println("MyData." + i + "=" + MyData[i]);
```

getDataObjectContents

7.0			
-----	--	--	--

Data オブジェクトに関連付けられている添付ファイルのコンテンツにアクセスできます。

パラメータ

cName	取得する Data オブジェクトに関連付けられている名前。
bAllowAuth	(オプション) デフォルト値は <code>false</code> です。 <code>true</code> の場合は、ダイアログボックスを使用してユーザ認証を行います。 <code>encryptForRecipients</code> を使用して data オブジェクトが暗号化されている場合は、認証が必要なことがあります。認証ダイアログボックスは <code>bAllowAuth</code> が <code>true</code> の場合に使用できます。

戻り値

ReadStream オブジェクト

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObject](#)、[openDataObject](#)、[createDataObject](#)、[importDataObject](#)、[setDataObjectContents](#)、[removeDataObject](#)、[Data](#) オブジェクトがあります。

例

このコードは、1つの PDF ファイルを複数のメンバで回覧して使用する場合に使用します。電子メールリストに登録されているメンバの間で PDF ファイルを回覧します。各受信者は予算の数値を入力して、リストの次の人へ文書を転送します。文書が送信される前に、埋め込まれているタブ区切りの文書 `budget.xls` (この文書の添付ファイル) に予算の数値が追加されます。最後の受信者はスプレッドシートアプリケーションで添付ファイル `budget.xls` を開いて、予算の数値を見ることができます。

```
// 現在の受信者の名前と部署を取得
var firstName = this.getField("Name.First").value;
var lastName = this.getField("Name.Last").value;
var deptName = this.getField("Dept.Name").value;
// 予算の数値を取得
var deptBudget = this.getField("Dept.Budget").value;
if (app.viewerVersion >= 7) {
    // 埋め込まれているファイルのファイルストリームオブジェクトを取得
    var oFile = this.getDataObjectContents("budget.xls");
```

```
// 文字列に変換
var myBudget = util.stringFromStream(oFile, "utf-8");
// タブで情報を区切って、現在のデータを末尾に追加
var myBudget = myBudget + "\r\n" + firstName
  + "\t" + lastName + "\t" + deptName + "\t" + deptBudget;
// ファイルストリームに変換
var oFile = util.streamFromString(myBudget, "uft-8");
// budget.xls に上書き
this.setDataObjectContents("budget.xls", oFile);
} else {
    app.alert("Acrobat 7.0 or later is required."
        + " Your budget data will not be included. "
        + "Will e-mail on to the next correspondent, sorry. "
        + "Send in your budget request using traditional methods.");
}
```

この後のコードは示されていませんが、文書を保存してメーリングリストの次の人に送信するコードになります。

この例では、[getDataObjectContents](#)、[setDataObjectContents](#)、[util.stringFromStream](#)、[util.streamFromString](#) を使用しています。

getField

3.01			
------	--	--	--

PDF 文書内の Field オブジェクトを JavaScript 変数にマッピングします。

Acrobat 6.0 以降、このメソッドは、個々のウィジェットの Field オブジェクトを返せるようになりました。詳しくは、[Field](#) オブジェクトを参照してください。

パラメータ

cName	目的のフィールドの名前。
-------	--------------

戻り値

PDF 文書内のフォームフィールドを表す Field オブジェクト。

例 1

テキストフィールドを複数行にして高さを 3 倍にします。

```
var f = this.getField("myText");
var aRect = f.rect;                                // 境界を表す矩形を取得
f.multiline = true;                               // 複数行にする
var height = aRect[1]-aRect[3];                   // 高さを計算
aRect[3] -= 2* height;                            // テキストフィールドの高さを 3 倍にする
f.rect = aRect;                                   // 設定
```

例 2 (Acrobat 6.0)

個々のウィジェット（この場合はラジオボタン）に JavaScript アクションを指定します。

```
var f = this.getField("myRadio.0");
f.setAction("MouseUp",
    "app.alert('Thanks for selecting the first choice.');");
```

例 3

フィールドのプロパティをすべてリストします。この方法を使用すれば、プログラムでフィールドとそのプロパティを複製することができます。

```
f = this.getField("myField");
for ( var i in f ) {
    try {
        if ( typeof f[i] != "function" )          // フィールドメソッドはリストしない
            console.println( i + ":" + f[i] )
    } catch(e) {}                         // このフィールドタイプに適用されないプロパティの場合に
}                                         // 例外が発生する。
```

getIcon

5.0				
-----	--	--	--	--

特定の icon オブジェクトを取得します。[icons](#) プロパティ、[addIcon](#)、[importIcon](#)、[removeIcon](#) の各メソッド、Field オブジェクトの [buttonGetIcon](#)、[buttonImportIcon](#)、[buttonSetIcon](#) の各メソッドも参照してください。

パラメータ

cName 取得する icon オブジェクトの名前。

戻り値

文書内の指定の名前が関連付けられている Icon オブジェクト。その名前のアイコンが存在しない場合は null。

例

次の例は、コンボボックスのカスタムキーストロークスクリプトです。このコンボボックスには、文書に埋め込まれている各アイコンの名前を表示名を持つ項目が含まれています。コンボボックスで項目を選択すると、その項目名に対応するアイコンが「myPictures」フィールドのボタンに表示されます。

```
if (!event.willCommit) {
    var b = this.getField("myPictures");
    var i = this.getIcon(event.change);
    b.buttonSetIcon(i);
}
```

Field オブジェクトの [buttonSetIcon](#) メソッドなどを参照してください。

getLegalWarnings

6.0			
-----	--	--	--

この文書に関する警告コメント（文書内で見つかった各警告をエントリとして持つオブジェクト）を返します。

注意：8.0 より前のバージョンの Acrobat では、`Document.getLegalWarnings` を実行すると、文書が変更された状態になります。

Adobe Reader では、ファイルを解析して警告リストを取得するこのプロセスは使用できません。

パラメータ

bExecute	<code>true</code> の場合は、ファイルを調べて、検出されたすべての警告を返します。Acrobat 8 では、PDF/SigQ 準拠チェックを実行してファイルを調べます。 <code>false</code> （デフォルト値）の場合は、ファイルに埋め込まれている警告と、証明者の証明（存在する場合）を返します。 Acrobat 6 と 7 では、証明を行うときに、警告コメントをファイルに埋め込むことができます（Field オブジェクトの <code>signatureSign</code> メソッドの <code>cLegalAttest</code> を使用）。 Acrobat 8 でも、証明者が証明を埋め込むことはできますが、警告自体を埋め込むことはできません。この証明を取得するには、 <code>bExecute=false</code> にしてこのメソッドを呼び出します。
----------	---

戻り値

警告コメントのプロパティ名と値を含む `DocLegalWarning` オブジェクト。各エントリの値は、その警告が文書内に出現する回数です。`bExecute` が `false` の場合は、『PDF Reference』バージョン 1.7 にリストされているプロパティ名が使用されます。`bExecute` が `true` の場合は、以下にリストされている PDF/SigQ Level A の違反がプロパティ名として使用されます。『PDF Reference』バージョン 1.7 に示されている警告リストと以下のリストは、重複する部分もありますが、大半は異なることに注意してください。

DocLegalWarning オブジェクト

次のプロパティは、PDF/SigQ1-A の違反を示します。

プロパティ	説明
<code>AlternateImages</code>	画像の XObject に代替バージョンが含まれていてはいけません。
<code>Annotations</code>	この文書には、注釈が含まれています。注釈の表示は、外部変数によって変化することがあります。
<code>CatalogHasAA</code>	エンドユーザ向けではない隠しアクション、またはエンドユーザに知られていない隠しアクションが文書に含まれています。アクションには、JavaScript アクション（文書のオープンや保存など）、マルチメディアの再生、メニュー項目の実行などがあります。
<code>CatalogHasOpenAction</code>	文書を開いたときに起動する隠しアクションが文書に含まれています。これらのアクションは、エンドユーザ向けでないか、エンドユーザに知られていない可能性があります。アクションには、JavaScript アクション（文書のオープンや保存など）、マルチメディアの再生、メニュー項目の実行などがあります。

プロパティ	説明
DevDepGS_FL	文書の拡張グラフィックスステートで FL キーが使用されています。このキーはオブジェクトの描画時に必要な許容平滑度を示す数値です。Acrobat とその他のアプリケーションとで内容表示が異なることがあります。
DevDepGS_TR	この PDF では色を置き換える可能性のあるトランスマスク関数が使用されています。極端な場合、黒が白に置き換えられる可能性もあります。
DocHasCryptFilter	文書の一部または全部が暗号化されており、暗号化に使用された方法が Acrobat の標準インストールで使用できません。例えば、Adobe LiveCycle Policy Server による暗号化などが使用されています。暗号フィルタを使用して暗号化されたストリームが文書に含まれています。
DocHasNonSigField	この文書には、署名のないフォームフィールドが含まれています。フィールドの表示は、外部変数によって変化することがあります。
DocHasPresentation	プレゼンテーションは使用できません。プレゼンテーションにアニメーションなどの要素が含まれていると、文書の外観や動作が変わることがあります。
DocHasPSXObject	可視要素が外部変数に基づいて変わる可能性があります。例えば、ロゴが時刻やズームレベルに応じて変わる可能性があります。PostScript の XObject は使用できません。
DocHasXFA	XFA ベース（ダイナミックフォーム）の文書は使用できません。このフォームでは、文書の外観や動作が変更される可能性があります。
DynamicSigAP	この文書には、署名された署名フィールドが含まれています。フィールドの表示は、外部変数によって変化することがあります。
ExternalOPIdicts	PDF ファイル内にない画像への文書リンクが代替として使用されています。例えば、代替の高解像度画像が印刷用に指定されている可能性があります。画像やフォームの XObject に OPI 代替バージョンが含まれていってはいけません。
ExternalRefXObject	文書のリンク先が、PDF ファイル内にない画像です。外部 XObject は使用できません。
ExternalStreams	文書に外部ストリームが含まれています。作成者により、外部ソースからデータを取得する可能性があるストリームとして指定されています。
GoTo3DViewActions	この文書には、ユーザの知らないところで 3D ビューを操作して文書の表示を変更する可能性がある「3D ビューへ移動」アクションが含まれています。
GoToEHasF	インターネット、ファイルシステム、またはネットワーク上の外部 PDF へのリンクが文書にあります。リンク先の内容の性質は管理できません。埋め込みの Go To アクションで外部階層を参照しないでください。
GoToRemoteAction	文書には、外部のコンテンツにリンクしている可能性がある Go To アクションが含まれています。
InvalidEOF	この PDF ファイルには、PDF ファイルの終わりを示すマーカーの後に余分なバイトが含まれています。

プロパティ	説明
InvalidFileHeader	この PDF ファイルには、PDF ファイルヘッダの前に余分なバイトが含まれています。
JavaScriptActions	ユーザの知らないところで起動する可能性がある JavaScript アクションが文書に含まれています。
LaunchActions	添付ファイルを起動するアクションが文書に含まれています。
MalformedContentStm	不正な形式の描画命令：構文エラーです。ページの内容がページコンテンツ定義の文法に違反しています。例えば、矩形の描画命令で構文が間違っている可能性があります。
MovieActions	ユーザの知らないところで起動する可能性があるムービー起動アクションが文書に含まれています。
NonEmbeddedFonts	文書に埋め込まれていないフォントがあります。該当フォントがないシステムでこの文書を開くと、別のフォントに置き換えて表示されます。フォント関連の警告は常にオンにしておくことをお勧めします。
OptionalContent	文書の内容が表示／非表示の切り替えが可能なレイヤーに分かれています。
PageHasAA	エンドユーザ向けではない隠しアクション、またはエンドユーザに知られていない隠しアクションがページに含まれています。アクションには、JavaScript アクション（文書のオープンや保存など）、マルチメディアの再生、メニュー項目の実行などがあります。
RenditionActions	ユーザの知らないところでムービーを起動する可能性があるレンディションアクションが文書に含まれています。
SetOCStateActions	この文書には、ユーザの知らないところでレイヤーの表示を変更して文書の表示を変更する可能性がある SetOCState アクションが含まれています。
SigFieldHasAA	マウスのポイントを移動するなどのユーザ操作によって起動される可能性のあるアクションが署名フィールドに含まれています。アクションには、JavaScript アクション（文書のオープンや保存など）、マルチメディアの再生、メニュー項目の実行などがあります。
SigFieldHasAction	クリックによって起動される可能性のあるアクションが署名フィールドに含まれています。アクションには、JavaScript アクション（文書のオープンや保存など）、マルチメディアの再生、メニュー項目の実行などがあります。
SoundActions	サウンドを起動するアクションが文書に含まれています。
TrueTypeFonts	この文書では TrueType フォントが使用されています。TrueType フォントや TrueType ベースの OpenType フォントは使用できません。これらは状況によって文書の外観を変える可能性があります。PDF/SigQ にこの制限はありません。また、環境設定の security¥DigSig¥bTrueTypeFontPDFSigQWarn が 1 に設定されていない場合はレポートされません。

プロパティ	説明
UnknownNamedAction	ユーザの知らないところでメニュー項目を起動する可能性がある名前付きアクションが文書に含まれています。
UnknownPDFContent	認識できない PDF コンテンツ：Acrobat の現在のバージョンではサポートしていない PDF コンテンツまたはカスタムコンテンツが文書に含まれています。この文書は新しいバージョンの Acrobat (PDF 1.8 以降) で作成されている可能性があります。
UnknownPDFContentStmOp	認識できない描画演算子：Acrobat の現在のバージョンではサポートしていない PDF コンテンツまたはカスタムコンテンツが文書に含まれています。この文書は新しいバージョンの Acrobat で作成されている可能性があります。
URIActions	文書には、外部のコンテンツにリンクしている Launch URI アクションが含まれています。
XObjHasInterpolate	文書の作成者により画像の補間が有効にされています。画像の補間は許可されません。

例

文書を処理し、PDF の警告コメントを取得します。

```
var w = this.getLegalWarnings( true );
console.println( "Actual Legal PDF Warnings:" );
for(i in w) console.println( i + " = " + w[i] );

var w1 = this.getLegalWarnings( false );
console.println( "Declared Legal PDF Warnings:" );
for(i in w1) console.println( i + " = " + w1[i] );

// 証明用の署名について、MDP 設定で
// 注釈が許可されているか

var f = this.getField( "AuthorSig" );
var s = f.signatureInfo();
if( s.mdp == "defaultAndComments" )
    console.println( "Annotations are allowed" );

// 作成者からのコメント

console.println( "Legal PDF Attestation:" );
console.println( w1.Attestation );
```

getLinks

6.0			
-----	--	--	--

ページの指定した座標内に含まれている Link オブジェクトの配列を取得します。[addLink](#) および [removeLinks](#) も参照してください。

パラメータ

nPage	Link オブジェクトが含まれているページ。最初のページは 0 です。
oCoords	矩形を表す 4 つの数値の配列。これらの数値は、回転ユーザースペースにおける矩形の左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。

戻り値

Link オブジェクトの配列。

例

文書内のリンク数をカウントし、コンソールに表示します。

```
var numLinks=0;
for ( var p = 0; p < this.numPages; p++)
{
    var b = this.getPageBox("Crop", p);
    var l = this.getLinks(p, b);
    console.println("Number of Links on page " + p + " is " + l.length);
    numLinks += l.length;
}
console.println("Number of Links in Document is " + numLinks);
```

getNthFieldName

4.0			
-----	--	--	--

文書内の n 番目のフィールドの名前を取得します。[numFields](#) も参照してください。

パラメータ

nIndex	名前を取得するフィールドのインデックス。
--------	----------------------

戻り値

文書内のフィールドの名前。

例

文書内のすべてのフィールドを列挙します。

```
for (var i = 0; i < this.numFields; i++)
    console.println("Field[" + i + "] = " + this.getNthFieldName(i));
```

getNthTemplate

X			
---	--	--	--

注意：このメソッドの代わりに、`templates` プロパティ、`getTemplate` メソッド、`Template` オブジェクトを使用してください。

文書内の `n` 番目のテンプレートの名前を取得します。

パラメータ

<code>nIndex</code>	取得するテンプレートのインデックス。
---------------------	--------------------

戻り値

指定のテンプレートの名前。

getOCGs

6.0			
-----	--	--	--

指定のページにある OCG オブジェクトの配列を取得します。

関連するメソッドとしては、[getOCGOrder](#)、[setOCGOrder](#)、[OCG](#) オブジェクトがあります。

パラメータ

<code>nPage</code>	(オプション) ページ番号 (0 から数えます)。指定しなかった場合は、文書内のすべての OCG を返します。
	引数を渡さなかった場合は、すべての OCG が名前のアルファベット順で返されます。 <code>nPage</code> を渡した場合は、そのページの OCG が作成順に返されます。

戻り値

OCG オブジェクトの配列。OCG が存在しない場合は `null`。

例

指定した文書とページのすべての OCG をオンにします。

```
function TurnOnOCGsForPage(doc, nPage)
{
    var ogcArray = doc.getOCGs(nPage);
    for (var i=0; i < ogcArray.length; i++)
        ogcArray[i].state = true;
}
```

getOCGOrder

7.0			
-----	--	--	--

文書の OCGOrder 配列を返します。この配列は UI にレイヤーを表示する方法を表します。

関連するメソッドとしては、[getOCGs](#)、[setOCGOrder](#)、[OCG](#) オブジェクトがあります。

戻り値

OCG オブジェクト、文字列、同様のサブ配列を含む配列。OCG が存在しない場合は null。

配列の順序については、[setOCGOrder](#) を参照してください。

getPageBox

5.0			
-----	--	--	--

回転ユーザースペースにおける、ページ上の指定の矩形を取得します。[setPageBoxes](#) も参照してください。

パラメータ

cBox (オプション) ボックスの種類。有効な値は、次のとおりです。

Art
Bleed
BBox
Crop (デフォルト)
Media
Trim

これらのボックスの定義については、『PDF Reference』バージョン 1.7 を参照してください。

nPage (オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。

戻り値

回転ユーザースペースにおける、ページ上の指定の矩形。

例

Media ボックスの大きさを取得します。

```
var aRect = this.getPageBox("Media");
var width = aRect[2] - aRect[0];
var height = aRect[1] - aRect[3];
console.println("Page 1 has a width of " + width + " and a height of "
+ height);
```

getPageLabel

5.0			
-----	--	--	--

指定のページのページラベル情報を取得します。

パラメータ

nPage (オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。

戻り値

指定のページのページラベル情報。

例

例については、[setPageLabels](#) を参照してください。

getPageNthWord

5.0		⌚	
-----	--	---	--

ページに含まれる n 番目の単語を取得します。

[getPageNumWords](#) および [selectPageNthWord](#) も参照してください。

注意：文書のセキュリティで内容を抽出できないように設定されている場合は、例外が発生します。

パラメータ

nPage (オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。

nWord (オプション) 単語のインデックス (0 から数えます)。デフォルトは 0 (ページの最初の単語) です。

bStrip (オプション) 区切り文字と空白文字を取り除いてから単語を返すかどうかを示します。デフォルトは true です。

戻り値

ページに含まれる n 番目の単語。

例

例については、spell.checkWord の[例 2](#) を参照してください。

getPageNthWordQuads

5.0			
-----	--	--	--

ページ上の n 番目の単語の quads のリストを取得します。Annotation オブジェクトの quads プロパティは、テキストマークアップ、下線、取り消し線、ハイライト、波線などの注釈を作成するために使用できます。[getPageNthWord](#)、[getPageNumWords](#)、[selectPageNthWord](#) も参照してください。

注意：文書のセキュリティで内容を抽出できないように設定されている場合は、例外が発生します。

パラメータ

nPage	(オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。
nWord	(オプション) 単語のインデックス (0 から数えます)。デフォルトは 0 (ページの最初の単語) です。

戻り値

ページ上の n 番目の単語の quads のリスト。

例

文書の 2 ページ目にある 5 番目の単語に下線を付けます。

```
var annot = this.addAnnot({
    page: 1,
    type: "Underline",
    quads: this.getPageNthWordQuads(1, 4),
    author: "A. C. Robat",
    contents: "Fifth word on second page"
});
```

他の例については、spell.[checkWord](#) を参照してください。

getPageNumWords

5.0			
-----	--	--	--

ページに含まれる単語の数を取得します。

[getPageNthWord](#)、[getPageNthWordQuads](#)、[selectPageNthWord](#) も参照してください。

パラメータ

nPage	(オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。
-------	--

戻り値

ページに含まれる単語の数。

例

文書内の単語数をカウントします。

```
var cnt=0;  
for (var p = 0; p < this.numPages; p++)  
    cnt += getPageNumWords(p);  
console.println("There are " + cnt + " words in this doc.");
```

他の例については、spell.checkWord の[例 2](#) を参照してください。

getPageRotation

5.0			
-----	--	--	--

指定のページの回転角度を取得します。[setPageRotations](#) も参照してください。

パラメータ

nPage (オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。

戻り値

回転角度 (0、90、180 または 270)。

getPageTransition

5.0			
-----	--	--	--

指定のページの効果を取得します。[setPageTransitions](#) も参照してください。

パラメータ

nPage (オプション) ページのインデックス (0 から数えます)。デフォルトは 0 (文書の最初のページ) です。

戻り値

3 つの値の配列 [nDuration, cTransition, nTransDuration]。

- nDuration は、ビューアが自動的にページを切り替えるまでの最大表示時間です。-1 という値は、自動的にページが切り替わらないことを示します。
- cTransition は、画面が切り替わる際にページに適用される効果の名前です。効果の有効な値については、app.fs.[transitions](#) プロパティを参照してください。
- cTransDuration は、効果の継続時間 (秒単位) です。

getPrintParams

6.0			
-----	--	--	--

デフォルトの印刷設定を表す `PrintParams` オブジェクトを取得します。[print](#) メソッドを参照してください。このメソッドは、`PrintParams` オブジェクトをパラメータとして取ります。

戻り値

`PrintParams` オブジェクト。

例

デフォルトのプリンタの `PrintParams` オブジェクトを取得します。

```
var pp = this.getPrintParams();
pp.colorOverride = pp.colorOverrides.mono; // いくつかのプロパティを設定
this.print(pp); // 印刷
```

getSound

5.0			
-----	--	--	--

指定した名前に対応する `sound` オブジェクトを取得します。[sounds](#)、[importSound](#)、[deleteSound](#)、[Sound](#) オブジェクトも参照してください。

パラメータ

cName	取得するオブジェクトの名前。
-------	----------------

戻り値

指定した名前に対応する `sound` オブジェクト。

例

サウンドを再生します。

```
var s = this.getSound("Moo");
console.println("Playing the " + s.name + " sound.");
s.play();
```

getTemplate

5.0			
-----	--	--	--

文書から指定の名前のテンプレートを取得します。[templates](#)、[createTemplate](#)、[removeTemplate](#)、[Template](#) オブジェクトも参照してください。

パラメータ

cName	取得するテンプレートの名前。
-------	----------------

戻り値

Template オブジェクト。指定の名前のテンプレートが文書に存在しない場合は null。

例

特定のテンプレートを取得し、それが非表示であるか表示されているかを調べます。

```
var t = this.getTemplate("myTemplate");
if ( t != null ) console.println( "myTemplate exists and is "
    + eval( '( t.hidden ) ? "hidden" : "visible"' ) + ".");
else console.println( "myTemplate is not present!" );
```

getURL

4.0	D	S	X
-----	---	---	---

GET を使用して指定の URL にインターネット経由でアクセスします。現在の文書をブラウザ内で表示しているか、Acrobat Web Capture が使用できない場合は、Weblink プラグインを使用して目的の URL にアクセスします。Acrobat 内で実行された場合は、[baseUrl](#) (Web Capture で取得された文書の場合は最初のページ、つまり、ページ 0 の URL) か、ファイルシステムのいずれかから現在の文書の URL を取得します。

注意： bAppend パラメータが true に設定されている場合、Adobe Reader でこのメソッドは使用できません。

このメソッドは、「Web ページを開く」アクションとほぼ同じです。

関連するメソッドとしては、app.[launchURL](#) があります。

パラメータ

cURL	完全修飾 URL または相対 URL。URL の最後にクエリ文字列を指定することもできます。
bAppend	(オプション) true (デフォルト) の場合は、取得されたページが現在の文書に追加されます。文書が Web ブラウザで表示されているか、Acrobat Web Capture プラグインが使用できないか、URL が「file:///」形式の場合、このフラグは false であると見なされます。

注意： Acrobat 6.0 以降で bAppend を true にした場合、getURL メソッドを実行できるのは、コンソールイベントまたはバッチャイベンツのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

例

```
this.getURL("http://www.adobe.com/", false);
```

gotoNamedDest

3.01			
------	--	--	--

PDF 文書内の指定の名前の移動先に移動します。名前付きの移動先とその作成方法について詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

パラメータ

cName	文書内の移動先の名前。
-------	-------------

例

文書を開いて、指定の名前の移動先に移動します。この例では、`openDoc` で開いている文書が `disclosed` であると仮定しています。

```
// 新しい文書を開く
var myNovelDoc = app.openDoc("/c/fiction/myNovel.pdf");
// この新しい文書内の移動先に移動
myNovelDoc.gotoNamedDest("chapter5");
// 古い文書を閉じる
this.closeDoc();
```

同じタスクをより効率的に行う方法については、[131 ページの `openDoc` の例 6 \(Acrobat 8.0\)](#) を参照してください。

importAnFDF

4.0	D		F
-----	---	--	---

指定の FDF ファイルを取り込みます。[importAnXFDF](#) および [importTextData](#) も参照してください。

パラメータ

cPath	(オプション) デバイスに依存しない、FDF ファイルのパス。このパスの形式は、 <code>submitForm</code> メソッドや、フォーム／フォームデータを管理／データを書き出しを使用して書き出された FDF ファイル内の F エントリの値と同等です。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。
-------	---

例

「ページを開く」イベントのアクションである次のコードでは、`ProcResponse` という関数が定義されているかどうかを確認し、未定義の場合は、FDF ファイルにある JavaScript を文書レベルに取り込みます。

```
if(typeof ProcResponse == "undefined") this.importAnFDF("myDLJS.fdf");
```

ここでは、相対パスを使用しています。このコードは、PostScript ファイルから作成した PDF ファイルに、文書レベルのスクリプトを自動的に取り込むのに便利です。

importAnXFDF

5.0	D		F
-----	---	--	---

XML フォームデータが含まれている指定の XFDF ファイルを取り込みます。

XFDF は、Acrobat フォームデータを XML で表現したものです。『XML Form Data Format (XFDF) Specification』を参照してください ([28 ページの「関連ドキュメント」](#) を参照)。

[exportAsXFDF](#)、[importAnFDF](#)、[importTextData](#) も参照してください。

パラメータ

cPath	(オプション) デバイスに依存しない、XFDF ファイルのパス。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。
-------	---

importDataObject

5.0	D	S	
-----	---	---	--

外部ファイルを文書に取り込み、指定の名前を data オブジェクトに関連付けます。data オブジェクトは後で抽出または操作できます。

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObject](#)、[openDataObject](#)、[createDataObject](#)、[exportDataObject](#)、[importDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#)、[Data](#) オブジェクトがあります。

注意： cDIPath パラメータを指定した場合、このメソッドを実行できるのは、バッチイベント、コンソールイベント、外部呼び出し (OLE など) のみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

`importDataObject` を使用して添付ファイルが取り込まれている場合は、cName パラメータによって `Data.name` の値が割り当てられています。しかし、UI を使用してファイルが添付されている場合、`name` は自動的に割り当てられます。添付ファイルには、「Untitled Object」、「Untitled Object 2」、「Untitled Object 3」などのように、番号の付いた名前が割り当てられます。

パラメータ

cName	data オブジェクトに関連付ける名前。
cDIPath	(オプション) デバイスに依存しない、ユーザのハードドライブにあるデータファイルのパス。このパスには、絶対パスまたは現在の文書に対する相対パスのどちらでも指定できます。指定しなかった場合は、データファイルを選択するように求めるプロンプトが表示されます。
cCryptFilter	(オプション、Acrobat 6.0) 言語に依存しない、この data オブジェクトを暗号化する暗号フィルタの名前。暗号フィルタは、Doc の <code>addRecipientListCryptFilter</code> メソッドを使用して、文書の暗号フィルタのリストにあらかじめ追加しておく必要があります。追加されていない場合は、例外が発生します。Doc の <code>encryptForRecipients</code> メソッドで暗号化したファイル内で、この data オブジェクトを暗号化たくない場合は、定義済みの Identity 暗号フィルタを使用します。

戻り値

成功した場合は `true`。失敗した場合は例外が発生します。

例

現在の文書に 2 つのファイルを添付し、Data オブジェクトのすべての情報をコンソールに書き込みます。

```
function DumpDataObjectInfo(dataobj)
{
    for (var i in dataobj)
        console.println(dataobj.name + "[" + i + "]=" + dataobj[i]);
}
// 埋め込むデータファイルをユーザに指定させる。
this.importDataObject("MyData");
DumpDataObjectInfo(this.getDataObject("MyData"));
// この文書の親ディレクトリにある Foo.xml を埋め込む。
this.importDataObject("MyData2", "../Foo.xml");
DumpDataObjectInfo(this.getDataObject("MyData2"));
```

importIcon

5.0			
-----	--	--	--

文書にアイコンを取り込んで、指定の名前を関連付けます。

`icons`、`addIcon`、`getIcon`、`removeIcon`、Field オブジェクトの `buttonGetIcon`、`buttonImportIcon`、`buttonSetIcon` の各メソッド、`Icon` オブジェクトも参照してください。

Acrobat バージョン 6.0 以降では、最初に `cDIPath` を PDF ファイルとして開こうと試みます。失敗した場合は、`cDIPath` を既知のグラフィック形式 (BMP、GIF、JPEG、PCX、PNG、TIFF) から PDF に変換して、変換されたファイルをボタンアイコンとして取り込もうと試みます。

注意： `cDIPath` を指定した場合、このメソッドを実行できるのは、バッチイベントとコンソールイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。Acrobat JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cName	アイコンに関連付ける名前。
cDIPath	(オプション) デバイスに依存しない、ユーザのハードドライブにある PDF ファイルのパス。このパスには、絶対パスまたは現在の文書に対する相対パスのどちらでも指定できます。 <code>cDIPath</code> を指定できるのは、バッチ環境またはコンソールのみです。指定しなかった場合、 <code>nPage</code> パラメータは無視され、PDF ファイルを指定してページを選択するプロンプトが表示されます。
nPage	(オプション) アイコンとして取り込む PDF ファイルのページのインデックス (0 から数えます)。デフォルトは 0 です。

戻り値

成功したかどうかを示す次の整数コード。

- 0 — エラーなし
- 1 — ユーザがダイアログボックスをキャンセル
- 1 — 選択したファイルを開けない
- 2 — 選択したページが無効

例

このメソッドは、後で利用する名前付きのアイコンを、文書にまとめて埋め込むのに便利です。例えば、ユーザがリストボックスで特定の項目を選択したときに、その項目に応じた絵をリストボックスの横に表示したいことがあります。このメソッドを使用しない場合は、いくつものフィールドの表示と非表示を切り替えて対処することになりますが、そのコードはかなり複雑になります。代わりに、次のようなスクリプトを使用すれば、簡単に対処できます。

```
var f = this.getField("StateListBox");
var b = this.getField("StateButton");
b.buttonSetIcon(this.getIcon(f.value));
```

この例では、同じ処理を行うために 1 つのフィールドしか使用していません。

名前付きのアイコンを文書に追加するための簡単なユーザインターフェイスを作成することもできます。例えば、アイコンの名前を入力するための `IconName` というフィールドと、アイコンを文書に追加するための `IconAdd` というフィールドがあるとします。`IconAdd` の「マウスボタンを放す」アクションのスクリプトは、次のようにになります。

```
var t = this.getField("IconName");
this.importIcon(t.value);
```

このようなスクリプトをバッチ処理で使用すると、フォルダ内で選択したすべてのアイコンファイルを文書に埋め込むことができます。

importSound

5.0			
-----	--	--	--

サウンドを文書に取り込み、指定の名前をそのサウンドに関連付けます。

注意： `cDIPath` を指定した場合、このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

<code>cName</code>	sound オブジェクトに関連付ける名前。
<code>cDIPath</code>	(オプション) デバイスに依存しない、ユーザのハードドライブにあるサウンドファイルのパス。このパスには、絶対パスまたは現在の文書に対する相対パスのどちらでも指定できます。指定しなかった場合は、サウンドファイルを選択するように求めるプロンプトが表示されます。

例

2つのサウンドを取り込んで、再生します。

```
this.importSound("Moo");
this.getSound("Moo").play();
this.importSound("Moof", "./moof.wav");
this.getSound("Moof").play();
```

[sounds](#)、[getSound](#)、[deleteSound](#)、[Sound](#) オブジェクトも参照してください。

importTextData

5.0			
-----	--	--	--

テキストファイルからデータを1行取り込みます。各行はタブで区切られている必要があります。最初の行はデータの列名で、PDFファイルに存在するテキストフィールドのフィールド名と同じです。データの行番号は0から数えるので、データの最初の行は行番号が0になります（列名の行は含まれません）。データの行を取り込むと、各列のデータはそれぞれ対応するフィールドの値になります。

書き出しを行うメソッドである [exportAsText](#) も参照してください。

注意： (Acrobat 8.0) cPath を指定した場合、このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cPath	(オプション) デバイスに依存しない、テキストファイルの相対パス。指定しなかった場合は、テキストデータファイルを選択するように求めるプロンプトが表示されます。
nRow	(オプション) 取り込むデータの行番号（0から数えます）。ただし、ヘッダ行はカウントしません。指定しなかった場合は、取り込む行を選択するように求めるプロンプトが表示されます。

戻り値

整数の戻りコード。

戻りコード	説明
-3	警告：データが見つかりません
-2	警告：ユーザが行の選択をキャンセルしました
-1	警告：ユーザがファイルの選択をキャンセルしました
0	エラーはありません
1	エラー：ファイルが開けません
2	エラー：データがロードできません
3	エラー：無効な行です

例 1

この例では、「First」、「Middle」、「Last」という名前のテキストフィールドがあるとします。また、データファイルの 1 行目に、タブで区切られた「First」、「Middle」、「Last」という 3 つの文字列があり、その後にタブで区切られた名前データが 4 行存在しているとします。

```
First           Middle          Last
A.             C.              Robat
T.             A.              Croba
A.             T.              Acrob
B.             A.              Tacro
// 「myData.txt」からデータの最初の 1 行を取り込む。
this.importTextData("/c/data/myData.txt", 0)
```

例 (続き)

次のコードは、ボタンの「マウスボタンを放す」アクションです。ボタンをクリックするたびに、テキストファイル内の次の行を取得し、「First」、「Middle」、「Last」の 3 つのフィールドにその行の名前データを埋め込みます。

```
if (typeof cnt == "undefined") cnt = 0;
this.importTextData("/c/data/textdata.txt", cnt++ % 4)
```

[ADBC](#) オブジェクトとそのプロパティやメソッドを使用しても、同じ処理が行えます。その場合、データファイルはスプレッドシートやデータベースでもかまいません。

importXFAData

6.0			
-----	--	--	--

指定の XFA ファイルを取り込みます。[importAnXFDF](#) および [importTextData](#) も参照してください。

注意：このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

パラメータ

cPath	(オプション) デバイスに依存しない、XFA ファイルのパス。このパスには、現在の文書の位置に対する相対パスも指定できます。このパラメータを省略すると、ファイルを選択するダイアログボックスが表示されます。
-------	--

insertPages

5.0			
-----	--	--	--

ソース文書のページを現在の文書に挿入します。ページ範囲を指定しない場合は、ソース文書の全ページが取得されます。

[deletePages](#) および [replacePages](#) も参照してください。

注意：このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

nPage	(オプション) ページのインデックス (0 から数えます)。このページの後にソース文書のページが挿入されます。文書の最初のページの前にページを挿入するには、-1 に設定します。
cPath	デバイスに依存しない、挿入するページが含まれている PDF ファイルのパス。このパスには、現在の文書の位置に対する相対パスも指定できます。
nStart	(オプション) 挿入するソース文書のページ範囲の開始を定義するインデックス (0 から数えます)。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 挿入するソース文書のページ範囲の終了を定義するインデックス (0 から数えます)。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。

例

現在の文書にカバーページを挿入します。

```
this.insertPages ({
    nPage: -1,
    cPath: "/c/temp/myCoverPage.pdf",
    nStart: 0
});
```

mailDoc

4.0			S
-----	--	--	---

現在の PDF 文書を保存し、この文書をすべての受信者に添付ファイルとして送信します。ユーザ操作を要求するかどうかを選択できます。

[mailForm](#)、[app.mailGetAddrs](#)、[app.mailMsg](#)、FDF オブジェクトの [mail](#) メソッド、Report オブジェクトの [mail](#) メソッドも参照してください。

注意：(Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、bUI パラメータは無視され、デフォルトの `true` の動作になります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

(保存権限) Adobe Reader 5.1 以降の場合、このメソッドは使用できますが、文書が変更されている場合は文書の保存権限が必要です。

Windows でこのメソッドを使用するには、クライアントコンピュータのデフォルトのメールプログラムで MAPI が有効になっている必要があります。

パラメータ

bUI	(オプション) <code>true</code> (デフォルト) の場合は、メールーの新規メッセージウィンドウがユーザに表示され、他のパラメータ値が初期値として使用されます。 <code>false</code> の場合、 <code>cTo</code> パラメータが必須で、その他のパラメータはすべてオプションです。
	注意: (Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、 <code>bUI</code> パラメータは無視され、デフォルトの <code>true</code> の動作になります。 32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」 を参照してください。
cTo	(オプション) セミコロンで区切られた、メッセージの受信者のリスト。
cCc	(オプション) セミコロンで区切られた、メッセージの CC 受信者のリスト。
cBcc	(オプション) セミコロンで区切られた、メッセージの BCC 受信者のリスト。
cSubject	(オプション) メッセージの件名。長さの制限は 64 KB です。
cMsg	(オプション) メッセージの内容。長さの制限は 64 KB です。

例

メールーのメッセージウィンドウを表示します。

```
this.mailDoc(true);
```

PDF ファイルが添付された電子メールを `apstory@example.com` と `dpsmith@example.com` に送信します。Acrobat 7.0 以降では、`bUI` パラメータを `false` に設定して、実際に UI が表示されないようにするために、セキュリティによる制限がないコンテキストでコードを実行する必要があります。

```
this.mailDoc({
  bUI: false,
  cTo: "apstory@example.com",
  cCC: "dpsmith@example.com",
  cSubject: "The Latest News",
  cMsg: "A.P., attached is my latest news story in PDF."
});
```

mailForm

4.0			F
-----	--	--	----------

フォームデータを書き出して、その FDF ファイルをすべての受信者に添付ファイルとして送信します。ユーザ操作を要求するかどうかを選択できます。このメソッドでは、署名済みの署名フィールドはサポートされません。

`mailDoc`、`app.mailGetAddrs`、`app.mailMsg`、FDF オブジェクトの `mail` メソッド、Report オブジェクトの `mail` メソッドも参照してください。

注意: Windows でこのメソッドを使用するには、クライアントコンピュータのデフォルトのメールプログラムで MAPI が有効になっている必要があります。

パラメータ

bUI	true の場合は、メールの新規メッセージウィンドウがユーザに表示され、他のパラメータ値が初期値として使用されます。false の場合、cTo パラメータが必須で、その他のパラメータはすべてオプションです。
	注意： (Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、bUI パラメータは無視され、デフォルトの true の動作になります。 32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」 を参照してください。
cTo	(bUI が true の場合に必要) セミコロンで区切られた、メッセージの受信者のリスト。
cCc	(オプション) セミコロンで区切られた、メッセージの CC 受信者のリスト。
cBcc	(オプション) セミコロンで区切られた、メッセージの BCC 受信者のリスト。
cSubject	(オプション) メッセージの件名。長さの制限は 64 KB です。
cMsg	(オプション) メッセージの内容。長さの制限は 64 KB です。

例

メールのメッセージウィンドウを表示します。

```
this.mailForm(true);
```

FDF ファイルが添付されたメールを fun1@example.com と fun2@example.com に送信します。

```
this.mailForm(false, "fun1@example.com; fun2@example.com", "", "",  
"This is the subject", "This is the body of the mail.");
```

movePage

5.0	D		X
-----	---	--	---

文書内でページを移動します。

パラメータ

nPage	(オプション) 移動するページのインデックス (0 から数えます)。デフォルトは 0 です。
nAfter	(オプション) ページのインデックス (0 から数えます)。このページの後に指定のページが移動されます。文書の最初のページの前にページを移動するには、-1 に設定します。デフォルトは、文書の最後のページです。

例

文書のページ順を逆にします。

```
for (i = this.numPages - 1; i >= 0; i--) this.movePage(i);
```

newPage

6.0	D	S	X
-----	---	---	---

アクティブな文書に新しいページを追加します。

注意：このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

パラメータ

nPage	(オプション) ページのインデックス (1 から数えます)。このページの後に新しいページが追加されます。デフォルトは、文書の最後のページです。最初のページの前にページを追加するには、0 に設定します。無効なページ範囲を指定すると、有効なページ範囲になるよう切り捨てられます。
nWidth	(オプション) ページの幅 (ポイント単位)。デフォルト値は 612 です。
nHeight	(オプション) ページの高さ (ポイント単位)。デフォルト値は 792 です。

例

文書のページサイズに合わせて、新しいページを追加します。

```
var Rect = this.getPageBox("Crop");
this.newPage(0, Rect[2], Rect[1]);
```

openDataObject

7.0			
-----	--	--	--

このメソッドを呼び出した PDF 文書に埋め込まれている data オブジェクト (添付ファイル) の Doc を返します。

次の場合には、Doc を返さずに例外が発生します。

- このメソッドを呼び出した文書に、埋め込みの data オブジェクトが含まれていない場合。
- data オブジェクトが PDF 文書でない場合。
- JavaScript で添付ファイルを開く権限がない場合。

不要になった文書は、(closeDoc を使用して) 閉じてください。

パラメータ

cName	data オブジェクトの名前。
-------	-----------------

data オブジェクトの名前は、Data オブジェクトのプロパティです。オブジェクトの名前はオブジェクトが埋め込まれるときに、Acrobat UI によって自動的に命名されるか、JavaScript の createDataObject メソッドまたは importDataObject メソッドによってプログラムで命名されます。

戻り値

Doc を返します。または例外が発生します。

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObjectContents](#)、[setDataObjectContents](#)、[createDataObject](#)、[importDataObject](#)、[Data](#) オブジェクトがあります。

例

PDF の添付ファイルを開いて、そのフォームデータを抽出します。

```
var oDoc = this.openDataObject("myAttachment");
try {
    var myField = this.getField("myTextField");
    // PDF 添付ファイルの「yourTextField」の値を取得
    var yourField = oDoc.getField("yourTextField");
    // この値を「myTextField」に表示
    myField.value = yourField.value;
    oDoc.closeDoc();
} catch(e) { app.alert("Operation failed"); }
```

`submitForm` メソッドの [346 ページの「例 5 \(Acrobat 7.0\)」](#) も参照してください。

print

3.01			
------	--	--	--

文書のすべてのページまたは特定のページを出力します。

Acrobat 6.0 以降では、`PrintParams` オブジェクトに含まれている設定を使用して文書を出力できるようになりました（他のパラメータを使用する必要はありません）。永続的な出力設定は変更されません。

注意：(Acrobat 6.0) ファイル出力する場合、パスはセーフパスである必要があります ([31 ページの「セーフパス」](#) を参照)。`print` メソッドによって既存のファイルは上書きされません。

(Acrobat 7.0) ユーザの操作を要求せずに、自動的な出力を行えるのは、バッヂイベント、コンソールイベント、メニューイベントのみです。自動的な出力を行うには、`bUI` を `false` に設定するか、次のように `interactive` プロパティを `silent` に設定します。

```
var pp = this.getPrintParams();
pp.interactive = pp.constants.interactionLevel.silent;
```

バッヂイベント、コンソールイベント、メニューイベント以外では、`bUI` や `interactive` の値は無視され、出力ダイアログボックスが常に表示されます。

[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。

注意：Windows プラットフォームでは、ファイル名に `.ps` または `.prn` という拡張子を付ける必要があります（大文字と小文字は区別されません）。また、`print` メソッドで、ルートディレクトリ、`Windows` ディレクトリ、`Windows` システムディレクトリにファイルを直接作成することはできません。

これらのセキュリティ制限のいずれかが満たされていない場合は、`InvalidArgsError` という例外 ([Error](#) オブジェクトを参照) が発生し、`print` は失敗します。

パラメータ

bUI	(オプション) <code>true</code> (デフォルト) の場合は、出力情報の取得とアクションの確認のために、ユーザに UI が表示されます。
nStart	(オプション) ページ範囲の開始を定義するインデックス (0 から数えます)。 <code>nStart</code> と <code>nEnd</code> を指定しなかった場合は、文書のすべてのページが出力されます。 <code>nStart</code> のみを指定した場合は、 <code>nstart</code> で指定した単一のページが対象範囲になります。 <code>nStart</code> と <code>nEnd</code> を使用した場合は、 <code>bUI</code> を <code>false</code> にする必要があります。
nEnd	(オプション) ページ範囲の終了を定義するインデックス (0 から数えます)。 <code>nStart</code> と <code>nEnd</code> を指定しなかった場合は、文書のすべてのページが出力されます。 <code>nEnd</code> のみを指定した場合は、0 から <code>nEnd</code> までが対象範囲になります。 <code>nStart</code> と <code>nEnd</code> を使用した場合は、 <code>bUI</code> を <code>false</code> にする必要があります。
bSilent	(オプション) <code>true</code> の場合は、文書の出力中にキャンセルのダイアログボックスが表示されません。デフォルトは <code>false</code> です。
bShrinkToFit	(オプション、Acrobat 5.0) <code>true</code> の場合は、(必要に応じて) ページが出力範囲に収まるように縮小されます。 <code>false</code> の場合は、縮小されません。デフォルトは <code>false</code> です。
bPrintAsImage	(オプション、Acrobat 5.0) <code>true</code> の場合は、ページが画像として出力されます。デフォルトは <code>false</code> です。
bReverse	(オプション、Acrobat 5.0) <code>true</code> の場合は、 <code>nEnd</code> から始まって、 <code>nStart</code> まで出力されます。デフォルトは <code>false</code> です。
bAnnotations	(オプション、Acrobat 5.0) <code>true</code> (デフォルト) の場合は、注釈が出力されます。
printParams	(オプション、Acrobat 6.0) 出力に使用する設定を含む <code>PrintParams</code> オブジェクト。このパラメータを渡すと、他のパラメータは無視されます。

例 1

現在のページを出力します。

```
this.print(false, this.pageNum, this.pageNum);
// ファイルを silent で出力
this.print({bUI: false, bSilent: true, bShrinkToFit: true});
```

例 2 (Acrobat 6.0)

現在の文書を既知のプリンタに出力します。

```
var pp = this.getPrintParams();
pp.interactive = pp.constants.interactionLevel.automatic;
pp.printerName = "hp officejet d series";
this.print(pp);
```

注意：`printerName` が空の文字列で、`fileName` が空でない場合は、現在の文書が PostScript ファイルとしてディスクに保存されます。

例 3 (Acrobat 6.0)

現在の文書を PostScript ファイルとして保存します。

```
var pp = this.getPrintParams();  
pp.fileName = "/c/temp/myDoc.ps";  
pp.printerName = "";  
this.print(pp);
```

removeDataObject

5.0	D		D
-----	---	--	---

指定の名前に対応する data オブジェクトを文書から削除します。

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObject](#)、[openDataObject](#)、[createDataObject](#)、[removeDataObject](#)、[importDataObject](#)、[getDataObjectContents](#)、[setDataObjectContents](#)、[Data](#) オブジェクトがあります。

パラメータ

cName 削除する data オブジェクトの名前。

data オブジェクトの名前は、Data オブジェクトのプロパティです。オブジェクトの名前はオブジェクトが埋め込まれるときに、Acrobat UI によって自動的に命名されるか、JavaScript の `createDataObject` メソッドまたは `importDataObject` メソッドによってプログラムで命名されます。

例

```
this.removeDataObject("MyData");
```

removeField

5.0	D		F
-----	---	--	---

指定のフィールドを文書から削除します。指定のフィールドが複数のページで表示されている場合は、すべて削除されます。

注意：(Acrobat 6.0) Acrobat 6.0 以降では、文書にフォーム使用権限が追加されていれば、Adobe Reader で `removeField` を使用することができます。

パラメータ

cName 削除するフィールドの名前。

例

```
this.removeField("myBadField");
```

removelcon

5.0			
-----	--	--	--

指定の名前のアイコンを文書から削除します。

[icons](#)、[addIcon](#)、[getIcon](#)、[importIcon](#)、Field オブジェクトの[buttonGetIcon](#)、[buttonImportIcon](#)、[buttonSetIcon](#) の各メソッド、[Icon](#) オブジェクトも参照してください。

パラメータ

cName	削除するアイコンの名前。
-------	--------------

アイコンの名前は Icon オブジェクトのプロパティです。オブジェクトの名前は、importIcon でアイコンファイルを取り込んだときか、addIcon で文書レベルの名前付きアイコンのツリーにアイコンを追加したときに命名されます。

例

すべての名前付きアイコンを文書から削除します。

```
for ( var i = 0; i < this.icons.length; i++ )
    this.removeIcon(this.icons[i].name);
```

removeLinks

6.0			
-----	--	--	--

文書からリンクを削除する権限がユーザにある場合は、指定したページの指定した座標内に存在するすべてのリンクを削除します。

[addLink](#)、[getLinks](#)、[Link](#) オブジェクトも参照してください。

パラメータ

nPage	リンクを削除するページのインデックス (0 から数えます)。
-------	--------------------------------

oCoords	矩形を表す 4 つの数値の配列。これらの数値は、回転ユーザースペースにおける矩形の左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。
---------	--

例

文書からすべてのリンクを削除します。

```
// 文書からすべてのリンクを削除
for ( var p = 0; p < this.numPages; p++)
{
    var b = this.getPageBox("Crop", p);
    this.removeLinks(p, b);
}
```

削除されたリンクの数は、[getLinks](#) を利用して算出できます。

removeRequirement

7.0.5	D	S	X
-------	---	---	---

PDF 文書に存在する既存の要件を削除します。要件を削除すると、Acrobat でその要件が満たされていなくとも、文書を開けるようになります。要件を削除すると、文書が適切に機能しなくなることがあります。

注意：このメソッドを呼び出せるのは、コンソールイベントまたはバッチャイ vent のみです。

パラメータ

cType	削除する要件のタイプ。タイプは Requirements 列挙子オブジェクトで指定します。
-------	---

removeScript

7.0	D		X
-----	---	--	---

スクリプトを削除する権限が付与されている場合は、文書レベルの JavaScript を削除します。

パラメータ

cName	削除するスクリプトの名前を指定する文字列。
-------	-----------------------

戻り値

成功した場合は `undefined` という値。スクリプトが見つからない場合は例外が発生します。

例

文書レベルのスクリプトを追加して、それを削除します。

```
this.addScript("myScript", "app.alert('A.C. Robat welcomes you!')");
```

このスクリプトを削除します。

```
this.removeScript("myScript");
```

removeTemplate

5.0	D	S	X
-----	---	---	---

文書から指定の名前のテンプレートを削除します。

[templates](#)、[createTemplate](#)、[getTemplate](#)、[Template](#) オブジェクトも参照してください。

注意：このメソッドを実行できるのは、バッチャイ vent またはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cName	削除するテンプレートの名前。
-------	----------------

テンプレート名は `Template` オブジェクトのプロパティです。テンプレートの名前はテンプレートが作成されるときに、Acrobat UI によって命名されるか、JavaScript の `getTemplate` メソッドによって命名されます。

removeThumbnails

5.0	D		X
-----	----------	--	---

指定したページのサムネールを文書から削除します。[addThumbnails](#) も参照してください。

パラメータ

nStart	(オプション) ページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) ページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。

removeWeblinks

5.0	D		X
-----	----------	--	---

指定したページをスキャンして、Web 上の URL にジャンプするリンクを削除します。[addWeblinks](#) も参照してください。

注意： このメソッドは、Acrobat の UI で作成された Web リンクのみを削除します。JavaScript (`getURL` など) によって実行される Web リンクは削除されません。

パラメータ

nStart	(オプション) ページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) ページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。

戻り値

文書から削除された Web リンクの数。

例

文書からすべての Web リンクを削除して、結果をコンソールウィンドウに表示します。

```
var numWeblinks = this.removeWeblinks();  
console.println("There were " + numWeblinks +  
    " web links removed from the document.");
```

replacePages

5.0	D	S	X
-----	---	---	---

現在の文書のページをソース文書のページで置換します。

[deletePages](#)、[extractPages](#)、[insertPages](#) も参照してください。

注意：このメソッドを実行できるのは、バッヂイベントまたはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

nPage	(オプション) 置換を開始するページのインデックス (0 から数えます)。デフォルトは 0 です。
cPath	デバイスに依存しない、置換対象となるページが含まれている PDF ファイルのパス。このパスには、現在の文書の位置に対する相対パスも指定できます。
nStart	(オプション) 置換に使用するソース文書のページ範囲の開始を定義するインデックス (0 から数えます)。 nStart と nEnd を指定しなかった場合は、ソース文書のすべてのページが対象になります。nStart のみを指定した場合は、nstart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 置換に使用するソース文書のページ範囲の終了を定義するインデックス (0 から数えます)。 nStart と nEnd を指定しなかった場合は、ソース文書のすべてのページが対象になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。

resetForm

3.01	D		F
------	---	--	---

文書に含まれるフィールド値をリセットします。フィールドをリセットすると、値がデフォルトに戻ります。テキストフィールドの場合は、通常は空白になります。

注意：フォームに署名フィールドが含まれている場合、Adobe Reader でこのメソッドを使用するには署名権限が必要です。

パラメータ

aFields	(オプション) リセットするフィールドを指定する配列。このパラメータを指定しなかつた場合や、null を指定した場合は、フォーム内のすべてのフィールドがリセットされます。配列には親のフィールド名を含めることができます。
---------	---

例 1

フィールドを選択してリセットします。

```
var fields = new Array();
fields[0] = "P1.OrderForm.Description";
fields[1] = "P1.OrderForm.Qty";
this.resetForm(fields);
```

1行のコードで同じフィールドをリセットできます。

```
this.resetForm(["P1.OrderForm.Description", "P1.OrderForm.Qty"]);
```

例 2

次の例では、サブツリー全体をリセットする方法を示します。例えば、フィールドの配列の一部として「name」を渡すと、name.first や name.last などのすべての name フィールドがリセットされます。

```
this.resetForm(["name"]);
```

saveAs

5.0			
-----	--	--	--

必須のパラメータ cPath で指定したデバイスに依存しないパスに、ファイルを保存します。保存時に Web 表示用の最適化は行われません。Acrobat 6.0 以降では、cConvID パラメータに値を設定して、文書を PDF 以外のファイルタイプに変換して保存することができます。

注意：このメソッドを実行できるのは、バッチャイベントまたはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

(Adobe Reader) : Adobe Reader では、保存権限が付与されている文書でのみこのメソッドを使用できます。

パラメータ

cPath	デバイスに依存しない、ファイルの保存先となるパス。
	注意： パラメータ cPath はセーフパスであり (31 ページの「セーフパス」 を参照)、cConvID の値に応じた拡張子を持つことが必要です。後述の cConvID の値と有効な拡張子 の表を参照してください。これらのセキュリティ条件が満たされていない場合は、NotAllowedError という例外 (Error オブジェクトを参照) が発生してメソッドが失敗します。

cConvID	(オプション、Acrobat 6.0) 変換ファイルタイプを指定する変換 ID の文字列。現在サポートされている cConvID の値については、 app.fromPDFConverters のリストを参照してください。cConvID を指定しない場合は、PDF になります。
---------	--

cFS	(オプション、Acrobat 7.0) ファイルシステムの名前を指定する文字列。サポートされている値は 2 つで、デフォルトのファイルシステムを表す空の文字列と、「CHTTP」です。デフォルト値は、デフォルトのファイルシステムです。Web サーバが WebDAV をサポートする場合にのみ、このパラメータは意味を持ちます。
bCopy	(オプション、Acrobat 7.0) ブーリアン値。true の場合は、PDF ファイルがコピーとして保存されます。デフォルトは false です。
bPromptToOverwrite	(オプション、Acrobat 7.0) ブーリアン値。true にすると、保存先ファイルが既に存在している場合にユーザにプロンプトが表示されます。デフォルトは false です。

戻り値

成功した場合は、`undefined` という値が返されます。エラーが発生した場合は例外が返されます。例えば、ユーザが上書きを許可しなかった場合は、`NotAllowedError` ([Error](#) オブジェクトを参照) が返されます。

注意：Acrobat 7.0 より前のバージョンでは、戻り値はありません。

cConvID の値と有効な拡張子

cConvID	有効な拡張子
com.adobe.Acrobat.eps	eps
com.adobe.Acrobat.html-3-20	html, htm
com.adobe.Acrobat.html-4-01-css-1-00	html, htm
com.adobe.Acrobat.jpeg	jpeg, jpg, jpe
com.adobe.Acrobat.jp2k	jp2f, jpx, jp2, j2k, j2c, jpc
com.adobe.Acrobat.doc	doc
com.callas.preflight.pdfa	pdf
com.callas.preflight.pdfx	pdf
com.adobe.Acrobat.png	png
com.adobe.Acrobat.ps	ps
com.adobe.Acrobat.rtf	rft
com.adobe.Acrobat.accesstext	txt
com.adobe.Acrobat.plain-text	txt
com.adobe.Acrobat.tiff	tiff, tif
com.adobe.Acrobat.xml-1-00	xml

注意：`jpeg`、`jp2k`、`png`、`tiff` に対応する変換 ID を指定した場合は、各ページが個別に保存されます。ファイル名は、指定したファイルの基本名に「`_Page_#`」が追加されたものになります。例えば、`cPath` の値が「`/c/temp/mySaveAsDocs/myJPGs.jpg`」の場合、生成されるファイルの名前は `myJPGs_Page_1.jpg`、`myJPGs_Page_2.jpg` のようになります。

例 1

次のコードは、スクリプトの概要を示しています。これはバッチシーケンスで使用できます。ここでは、フォームフィールドを含む PDF ファイルが開いているものとしています。フィールドには、データベースや保存されている文書からデータを取り込む必要があります。

```
// データベースからデータを読み取ってフォームに入力するコードをここに記述
// 次に、ファイルをフォルダに保存する。データベースレコードの customerID を
// 名前として使用
var row = statement.getRow();
.....
this.saveAs("/c/customer/invoices/" + row.customerID + ".pdf");
```

例 2

`newDoc` と `addField` を使用してフォームを動的にレイアウトし、データベースからフォームに入力して保存します。

```
var myDoc = app.newDoc()
// 動的なフォームフィールドを配置
// データベースに接続し、データベースからデータを取り込む
.....
// 文書を保存または印刷、ここでは silent で
// デフォルトプリンタを使用
myDoc.saveAs("/c/customer/invoices/" + row.customerID + ".pdf");
myDoc.closeDoc(true); // 通知せずに文書を閉じる
```

例 3 (Acrobat 6.0)

現在の文書をリッチテキスト形式で保存します。

```
this.saveAs("/c/myDocs/myDoc.rtf", "com.adobe.Acrobat.rtf");
```

サポートされている変換 ID 文字列については、[fromPDFConverters](#) を参照してください。

例 3 (Acrobat 7.0)

文書を WebDAV フォルダに保存します。

```
this.saveAs({
  cPath: "http://www.example.com/WebDAV/myDoc.pdf",
  bPromptToOverwrite: true,
  cFS: "CHTTP"
});
```

scroll

3.01			
------	--	--	--

現在のページ上の指定のポイントを、現在のビューの中央にスクロールします。座標は回転ユーザースペースで指定する必要があります。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

パラメータ

nX	スクロールするポイントの x 座標。
nY	スクロールするポイントの y 座標。

selectPageNthWord

5.0			
-----	--	--	--

ページを移動し、そのページ上の指定の単語を選択します。

[getPageNthWord](#)、[getPageNthWordQuads](#)、[getPageNumWords](#) も参照してください。

パラメータ

nPage	(オプション) 操作するページのインデックス（0 から数えます）。デフォルトは 0（文書の最初のページ）です。
nWord	(オプション) 取得する単語のインデックス（0 から数えます）。デフォルトは 0（ページの最初の単語）です。
bScroll	(オプション) 選択した単語が表示されていない場合、表示されるようにスクロールするかどうかを示します。デフォルトは true です。

例

特定の単語を取得して選択します。

```
// 2 ページ目（0 から始まるインデックス番号で数えると、1 に相当）にある 20 番目の単語を取得。  
var cWord = this.getPageNthWord(1, 20);  
// ユーザが確認できるようにその単語を選択（強調表示）し、  
// 必要に応じてページを変更。  
this.selectPageNthWord(1, 20);
```

setAction

6.0	④		X
-----	---	--	---

トリガに対する文書の JavaScript アクションを設定します。

[addScript](#)、[setPageAction](#)、Bookmark オブジェクトの [setAction](#) メソッド、Field オブジェクトの [setAction](#) メソッドも参照してください。

注意：このメソッドを実行すると、選択したトリガで既に定義されているアクションが上書きされます。

パラメータ

cTrigger	アクションを割り当てるトリガの名前。有効な値は、次のとおりです。 WillClose WillSave DidSave WillPrint DidPrint
cScript	トリガが発生したときに実行する JavaScript 式。

例

WillSave アクションと DidSave アクションを挿入します。ファイルの保存前と保存後にファイルサイズを取得して比較します。

```
// 「WillSave」用スクリプト
var myWillSave = 'var filesizeBeforeSave = this.filesize;¥r'
+ 'console.println("File size before saving is " + '
+ 'filesizeBeforeSave );';

// 「DidSave」用スクリプト
var myDidSave = 'var filesizeAfterSave = this.filesize;¥r'
+ 'console.println("File size after saving is "'
+ 'filesizeAfterSave );¥r'
+ 'var difference = filesizeAfterSave - filesizeBeforeSave;¥r'
+ 'console.println("The difference is " + difference );¥r'
+ 'if ( difference < 0 )¥r¥t'
+ 'console.println("Reduced filesize!");¥r'
+ 'else¥r¥t'
+ 'console.println("Increased filesize!");'

// 文書のアクションを設定 ...
this.setAction("WillSave", myWillSave);
this.setAction("DidSave", myDidSave);
```

setDataObjectContents

7.0			D
-----	--	--	---

cName パラメータで指定した添付ファイルを oStream パラメータのコンテンツで置き換えます。

パラメータ

cName	<code>oStream</code> で置き換える Data オブジェクトに関連付けられている名前。
oStream	添付ファイルのコンテンツを表す <code>ReadStream</code> オブジェクト。
cCryptFilter	(オプション) 言語に依存しない、暗号フィルタの名前。この <code>data</code> オブジェクトを暗号化する際に利用します。暗号フィルタは、 <code>addRecipientListCryptFilter</code> メソッドを使用して、文書の暗号フィルタのリストにあらかじめ追加しておく必要があります。追加されていない場合は、例外が発生します。 <code>data</code> オブジェクトをファイル内で暗号化たくない場合は、定義済みの <code>Identity</code> 暗号フィルタを使用します。使用しない場合は、 <code>encryptForRecipients</code> メソッドによって暗号化されます。

例 1

[298 ページの「例」](#) を参照してください。

例 2

この文書には `Acrobat.xml` という名前の添付ファイルがあります。添付ファイルを開き、XML データを更新し、新しい XML 文書を添付ファイルに保存します。この XML 添付ファイルを送信することもできます。`submitForm` メソッドの [346 ページの「例 5 \(Acrobat 7.0\)」](#) を参照してください。次の例では、`XMLData`.[applyXPath](#) の例で定義した XML データを使用します。

```
// 添付ファイルのファイルストリームオブジェクトを取得
var Acrobat = this.getDataObjectContents("Acrobat.xml");

// 文字列に変換
var cAcrobat = util.stringFromStream(Acrobat, "utf-8");

// 解析して XFAObject を取得
var myXML = XMLData.parse(cAcrobat, false);

// grandad の income の値を変更
myXML.family.grandad.personal.income.value = "300000";

// XML 文書を文字列変数 cAcrobat に保存
var cAcrobat = myXML.saveXML('pretty');

// ファイルストリームに変換
var Acrobat = util.streamFromString(cAcrobat, "utf-8");

// 添付ファイル Acrobat.xml をこのファイルストリームで更新
this.setDataObjectContents("Acrobat.xml", Acrobat);
```

関連するオブジェクト、プロパティ、メソッドとしては、[dataObjects](#)、[getDataObject](#)、[openDataObject](#)、[createDataObject](#)、[importDataObject](#)、[getDataObjectContents](#)、[removeDataObject](#)、[Data](#) オブジェクトがあります。

setOCGOrder

7.0			
-----	--	--	--

文書の OCGOrder 配列を設定します。この配列は UI にレイヤーを表示する方法を表します。

最も簡単な順序配列は OCG オブジェクトの単層配列です。この場合は、リストされた OCG と同じ順番で、単層のリストとして表示されます。順序配列の中にサブ配列があり、サブ配列の最初の要素が文字列である場合は、その文字列の下にサブ配列の残りの要素がネストした形でリストに表示されます。サブ配列の最初の要素が文字列でない場合は、サブ配列の前の OCG の下に、サブ配列全体がネストした形で表示されます。

関連するメソッドとしては、[getOCGs](#)、[getOCGOrder](#)、[OCG](#) オブジェクトがあります。

パラメータ

`oOrderArray` この文書の OCG 順序配列として使用する配列。

例

UI にリストされる OCG の順序を逆にします。

```
var ocgOrder = this.getOCGOrder();
var newOrder = new Array();
for (var j=0; j < ocgOrder.length; j++)
    newOrder[j] = ocgOrder[ocgOrder.length-j-1];
this.setOCGOrder(newOrder);
```

setPageAction

6.0			
-----	--	--	--

トリガに対する文書のページのアクションを設定します。

[setAction](#)、[addScript](#)、Bookmark オブジェクトの [setAction](#) メソッド、Field オブジェクトの [setAction](#) メソッドも参照してください。

注意：このメソッドを実行すると、選択されたページおよびトリガで既に定義されているアクションが上書きされます。

パラメータ

`nPage` アクションの追加先を示す文書のページのインデックス（0 から数えます）。

`cTrigger` アクションを行うトリガ。有効な値は、次のとおりです。

Open
Close

`cScript` トリガが発生したときに実行する JavaScript 式。

例

次の例では、最初のページが開かれたときにビープ音を鳴らします。

```
this.setPageAction(0, "Open", "app.beep(0);");
```

setPageBoxes

5.0			
-----	--	--	--

指定のページにある指定の名前のボックスを設定します。

[getPageIndex](#) も参照してください。

パラメータ

cBox	(オプション) ボックスの種類を示す値。次のいずれかになります。 Art Bleed Crop Media Trim
BBox	「BBox」という種類のボックスは読み取り専用で、 getPageIndex でのみサポートされています。この種類のボックスは、文書のすべてのページに適用され、複数のページにまたがる場合でも、各ページに個別に適用されます。この種類のボックスは、 getPageIndex でのみサポートされています。
nStart	(オプション) 対象文書のページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 対象文書のページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。
rBox	(オプション) 指定するボックスを表す、回転ユーザースペースにおける 4 つの数値の配列。指定しない場合、ボックスは削除されます。

setPageLabels

5.0			
-----	--	--	--

ページ番号のスタイルを設定します。設定したスタイルは、指定のページから、別のスタイルが設定されているページまで有効になります。

[getPageLabel](#) も参照してください。

パラメータ

nPage	(オプション) ラベルを付けるページのインデックス (0 から数えます)。
aLabel	(オプション) 3 つの必須項目を含む配列 [cStyle, cPrefix, nStart]。 <ul style="list-style-type: none">cStyle はページ番号のスタイルです。有効な値は、次のとおりです。<ul style="list-style-type: none">D — 10 進数R または r — ローマ数字 (大文字または小文字)A または a — アルファベット (大文字または小文字)これらのスタイルの正確な定義については、『PDF Reference』バージョン 1.7 を参照してください。cPrefix は、ページラベルの数字の前に付ける文字列です。nStart は、ページラベルの開始番号を示す序数です。 指定しなかった場合は、指定のページから次にページ番号スタイルが設定されているページまで、スタイルの設定が解除されます。 aLabel の値を null にすることはできません。

例 1

10 ページの文書があり、最初の 3 ページには小文字のローマ数字、次の 5 ページには数字 (1 から開始)、最後の 2 ページには「Appendix-」のプレフィックスと英字のページラベルを付けるとします。

```
this.setPageLabels(0, [ "r", "", 1]);
this.setPageLabels(3, [ "D", "", 1]);
this.setPageLabels(8, [ "A", "Appendix-", 1]);
var s = this.getPageLabel(0);
for (var i = 1; i < this.numPages; i++)
    s += " " + this.getPageLabel(i);
console.println(s);
```

この結果、コンソールには次のように出力されます。

```
i, ii, iii, 1, 2, 3, 4, 5, Appendix-A, Appendix-B
```

例 2

文書からページラベルをすべて削除します。

```
for (var i = 0; i < this.numPages; i++) {
    if (i + 1 != this.getPageLabel(i)) {
        // ページラベルがページ番号の序数と一致していない
        this.setPageLabels(i);
    }
}
```

setPageRotations

5.0	D		X
-----	---	--	---

現在の文書の指定のページを回転します。

[getPageRotation](#) も参照してください。

パラメータ

nStart	(オプション) 対象文書のページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 対象文書のページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
nRotate	(オプション) 対象ページに適用する回転角度。有効な値は、0、90、180、270 です。デフォルトは 0 です。

例

現在の文書のページ 0 ~ 10 を回転します。

```
this.setPageRotations(0, 10, 90);
```

setPageTabOrder

6.0	D		X
-----	---	--	---

ページ上のフォームフィールドのタブの順序を設定します。タブの順序は、行順、列順または構造順に設定できます。

PDF 1.4 文書を Acrobat 6.0 で表示した場合、フィールド間のタブの順序は Acrobat 5.0 の場合と同じになります。同様に、PDF 1.5 文書を Acrobat 5.0 で開いた場合、フィールドのタブの順序は Acrobat 6.0 の場合と同じになります。

パラメータ

nPage	タブの順序を設定するページのインデックス (0 から数えます)。
cOrder	使用する順序。有効な値は、次のとおりです。 rows columns structure

例

すべてのページのタブの順序を rows に設定します。

```
for (var i = 0; i < this.numPages; i++)  
    this.setPageTabOrder(i, "rows");
```

setPageTransitions

5.0	D		X
-----	---	--	---

指定のページ範囲のページ効果を設定します。

[getPageTransition](#) も参照してください。

パラメータ

nStart	(オプション) 対象文書のページ範囲の開始を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nStart のみを指定した場合は、nStart で指定した単一のページが対象範囲になります。
nEnd	(オプション) 対象文書のページ範囲の終了を定義するインデックス (0 から数えます)。nStart と nEnd を指定しなかった場合は、文書のすべてのページが対象になります。nEnd のみを指定した場合は、0 から nEnd までが対象範囲になります。
aTrans	(オプション) ページ効果を表す 3 つの値の配列 [nDuration, cTransition, nTransDuration]。 <ul style="list-style-type: none">• nDuration は、ビューアが自動的にページを切り替えるまでの最大表示時間です。-1 に設定すると、ページは自動的に切り替わらなくなります。• cTransition は、画面が切り替わる際にページに適用される効果の名前です。効果の有効な値については、fullScreen.transitions を参照してください。• nTransDuration は、効果の継続時間 (秒単位) です。 aTrans を指定しなかった場合は、指定したページのページ効果がすべて削除されます。

例

文書をフルスクリーンモードにし、効果を適用します。

```
this.setPageTransitions({ aTrans: [-1, "Random", 1] } );
app.fs.isFullScreen=true;
```

spawnPageFromTemplate

X	D		F
---	---	--	---

注意：このメソッドに代わりに、`templates`、`createTemplate`、`Template` オブジェクトの `spawn` メソッドを使用してください。

テンプレートを使用して、文書にページを作成します。テンプレートは `getNthTemplate` で取得できます。テンプレート機能は、Adobe Reader では動作しません。

パラメータ

cTemplate	テンプレート名。
nPage	(オプション) ページの番号 (0 から数えます)。bOverlay の値に応じて、このページに重ねられるか、このページの直前に挿入されるか決定されます。nPage を省略した場合は、文書の最後に新しいページが追加されます。
bRename	(オプション) フィールドの名前を変更するかどうかを示します。デフォルトは true です。
bOverlay	(オプション、Acrobat 4.0) false の場合、テンプレートは nPage で指定したページの前に挿入されます。true (デフォルト) の場合は、指定したページ上に重ねられます。
oXObject	(オプション、Acrobat 6.0) このパラメータの値には、spawnPageFromTemplate を前回呼び出したときの戻り値を指定します。

戻り値

6.0 より前のバージョンの Acrobat では、このメソッドは何も返しませんでした。現在、このメソッドは、作成したページのページコンテンツを表すオブジェクトを返します。返されたこのオブジェクトは、spawnPageFromTemplate を再び呼び出すときに、オプションパラメータ oXObject の値として使用できます。

注意：同じ内容のページを繰り返し作成すると、ファイルサイズが非常に大きくなります。この問題を回避するために、spawnPageFromTemplate は、作成されたページのページコンテンツを表すオブジェクトを返すようになりました。spawnPageFromTemplate メソッドを再び呼び出すときに、この戻り値を oXObject パラメータの値として使用すれば、同じページを作成することができます。

例 1

各テンプレートをもとに、ページを現在の文書に作成します。

```
var n = this.numTemplates;
var cTempl;
for (i = 0; i < n; i++) {
    cTempl = this.getNthTemplate(i);
    this.spawnPageFromTemplate(cTempl);
}
```

例 2 (Acrobat 6.0)

次の例では、oXObject パラメータと戻り値を使用して、同じテンプレートをもとに、ページを 31 回作成します。この手法を使用すると、ファイルサイズが大きくなりすぎるのを回避できます。

```
var t = this.getNthTemplate(0)
var XO = this.spawnPageFromTemplate(t, this.numPages, false, false);
for (var i=0; i < 30; i++)
    this.spawnPageFromTemplate(t, this.numPages, false, false, XO);
```

submitForm

3.01			
------	--	--	--

指定した URL にフォームを送信します。このメソッドを呼び出すには、Web ブラウザで文書を表示しているか、Acrobat Web Capture プラグインがインストールされている必要があります（「mailto」を使用した URL は、Web ブラウザで文書を表示していくなくても、SendMail プラグインがインストールされていれば正しく機能します）。Adobe Reader 6.0 以降では、Web ブラウザで文書を表示していくなくてもこのメソッドを呼び出せるようになりました。

注意：(Acrobat 6.0) 渡すパラメータによって、このメソッドの使用に制限が課される場合があります。パラメータの説明に記載されている注意を参照してください。

https プロトコルがサポートされており、セキュア接続が可能です。

パラメータ

cURL	送信先の URL。送信するデータの内容が FDF または XFDF であり (<code>cSubmitAs</code> の値が「FDF」または「XFDF」であり)、ブラウザウィンドウで文書を表示する場合は、この文字列の末尾が #FDF であることが必要です。
bFDF	(オプション) <code>true</code> (デフォルト) の場合、データは FDF として送信されます。 <code>false</code> の場合は、URL エンコードされた HTML として送信されます。 注意： このオプションは非推奨になりました。代わりに <code>cSubmitAs</code> を使用してください。
bEmpty	(オプション) <code>true</code> の場合は、値のないフィールドも含め、すべてのフィールドが送信されます。 <code>false</code> (デフォルト) の場合は、現在値のないフィールドは除外されます。 注意： XDP、XML または XFD としてデータを送信する場合 (後述の <code>cSubmitAs</code> パラメータを参照)、このパラメータは無視されます。空のフィールドも含めてすべてのフィールドが送信されます。 <code>aFields</code> を参照してください。
aFields	(オプション) 送信するフィールド名の配列、または 1 つのフィールド名を表す文字列。 <ul style="list-style-type: none">このパラメータを指定すると、指定したフィールドのみが送信されます (<code>bEmpty</code> で除外したフィールドは除かれます)。指定しない場合または <code>null</code> の場合は、<code>bEmpty</code> で除外されるものを除いてすべてのフィールドが送信されます。空の配列の場合は、フィールドは送信されません。ただし、この場合でも、<code>bAnnotations</code> が <code>true</code> であれば、送信される FDF ファイルにデータが含まれることがあります。 親のフィールド名を指定すれば、フィールドのサブツリー全体を書き出すことができます。 注意： XDP、XML または XFD としてデータを送信する場合 (<code>cSubmitAs</code> パラメータを参照)、このパラメータは無視されます。空のフィールドも含めてすべてのフィールドが送信されます。 <code>bEmpty</code> を参照してください。
bGet	(オプション、Acrobat 4.0) <code>true</code> の場合は、HTTP の GET メソッドを使用して送信します。 <code>false</code> (デフォルト) の場合は、POST を使用します。GET が許可されるのは、Acrobat Web Capture を使用して (ブラウザインターフェイスでは POST のみがサポートされます)、データを HTML として送信する (<code>cSubmitAs</code> が HTML である) 場合のみです。
bAnnotations	(オプション、Acrobat 5.0) <code>true</code> の場合、送信される FDF や XML ファイルに注釈が含まれます。デフォルトは <code>false</code> です。 <code>cSubmitAs</code> が FDF または XFDF の場合にのみ使用できます。

bXML	(オプション、Acrobat 5.0) true の場合は、XML として送信します。デフォルトは false です。 注意： このオプションは非推奨になりました。代わりに cSubmitAs を使用してください。
bIncrChanges	(オプション、Acrobat 5.0) true の場合は、送信される FDF ファイルに PDF 文書の追加変更が含まれます。デフォルトは false です。cSubmitAs が FDF の場合にのみ使用できます。Adobe Reader では使用できません。
bPDF	(オプション、Acrobat 5.0) true の場合、PDF 文書全体を送信します。デフォルトは false です。true の場合は、cURL を除くすべてのパラメータが無視されます。Adobe Reader では使用できません。 注意： このオプションは非推奨になりました。代わりに cSubmitAs を使用してください。
bCanonical	(オプション、Acrobat 5.0) true の場合は、送信される日付が標準形式 (D:YYYYMMDDHHmmSSOHH'mm' の形式。『PDF Reference』バージョン 1.7 を参照) に変換されます。デフォルトは false です。
bExclNonUserAnnots	(オプション、Acrobat 5.0) true の場合は、現在のユーザによって作成された注釈以外の注釈をすべて除外します。デフォルトは false です。
bExclFKey	(オプション、Acrobat 5.0) true の場合は、F エントリを除外します。デフォルトは false です。
cPassword	(オプション、Acrobat 5.0) 送信する前に FDF ファイルを暗号化する必要がある場合に、暗号キーを生成するために使用するパスワード。 暗号化された FDF ファイルを送信または受信するためにユーザが (現在の Acrobat セッションで) 既に入力したパスワードを使用する場合は、true (引用符なし) という値を渡します。パスワードが入力されていない場合は、パスワードを入力するプロンプトが表示されます。 パスワードが既に入力されていたか、このメソッドの実行時に入力されるかに関係なく、新しいパスワードが現在の Acrobat セッションで保存され、送信または受信される暗号化された FDF ファイルに使用されます。 cSubmitAs が FDF の場合にのみ使用できます。 警告： Acrobat 8.0 以降では、パスワードで暗号化した FDF ファイルは作成できません。このパラメータを使用した場合、パスワードで暗号化した FDF をフォームから送信しようとすると、ESErrorInvalidArgs という例外が発生し、フォームの送信は行われません。
bEmbedForm	(オプション、Acrobat 6.0) true の場合は、送信するデータが含まれているフォーム全体が FDF ファイルに埋め込まれます。 cSubmitAs が FDF の場合にのみ使用できます。
oJavaScript	(オプション、Acrobat 6.0) 送信する FDF ファイルに Before、After、Doc スクリプトを含めることができます。これを指定した場合は、値が類似の CosObj に直接変換され、FDF ファイル内で JavaScript 属性として使用されます。例えば、次のとおりです。 oJavaScript: { Before: 'app.alert("before!")', After: 'app.alert("after")', Doc: ["MyDocScript1", "myFunc1()", "MyDocScript2", "myFunc2()"] } cSubmitAs が FDF の場合にのみ使用できます。

cSubmitAs	(オプション、Acrobat 6.0) 送信形式を指定します。有効な値は、次のとおりです。 FDF — (デフォルト) FDF として送信 XFDF — XFDF として送信 HTML — HTML として送信 XDP — XDP として送信 XML — XML として送信。Acrobat 7.0 では、 <code>oXML</code> というパラメータ (Acrobat 7.0 の新しいパラメータ) に有効な <code>XMLData</code> オブジェクトが含まれていない場合に、フォームデータが XML 形式で送信されます。含まれている場合は、代わりにその <code>XMLData</code> オブジェクトが送信されます。 XFD — Adobe Form Client Data File として送信 PDF — 完全な PDF 文書を送信。 <code>cURL</code> 以外のすべてのパラメータが無視されます。
	注意： (保存権限が必要) Adobe Reader では、文書に保存権限が付与されていない場合は PDF を使用できません。
	このパラメータは、個々のフォーマットパラメータより優先されますが、次の優先順位 (高から低) が適用されます。 <code>cSubmitAs</code> 、 <code>bPDF</code> 、 <code>bXML</code> 、 <code>bFDF</code> 。
bInclNMKey	(オプション、Acrobat 6.0) <code>true</code> の場合は、注釈の NM エントリが含まれます。デフォルトは <code>false</code> です。
aPackets	(オプション、Acrobat 6.0) XDP の送信に含める情報を指定する文字列の配列。このパラメータは <code>cSubmitAs</code> が <code>XDP</code> の場合にのみ使用できます。有効な文字列は、次のとおりです。 <code>config</code> <code>datasets</code> <code>sourceSet</code> <code>stylesheet</code> <code>template</code> <code>pdf</code> — PDF を埋め込みます。 <code>pdf</code> を指定しない場合は、PDF のリンクのみが XDP に含まれます。 <code>xfdf</code> — XDP に注釈を含めます (注釈の情報は XFDF 形式であるため)。 <code>*</code> — すべての情報を XDP に含めます。 <code>pdf</code> は、デフォルトで参照として含められます。PDF ファイルを XDP に埋め込むには、明示的に <code>pdf</code> を指定します。
	注意： (保存権限が必要) Adobe Reader で、明示的に <code>pdf</code> をリストして、 <code>XDP</code> として文書を送信するには、その文書に保存権限が付与されている必要があります。
	デフォルトは、 <code>["datasets", "xfdf"]</code> です。
cCharset	(オプション、Acrobat 6.0) 送信する値のエンコーディング。 <code>utf-8</code> 、 <code>utf-16</code> 、 <code>Shift-JIS</code> 、 <code>BigFive</code> 、 <code>GBK</code> 、 <code>UHC</code> のいずれかの文字列を値として指定できます。このパラメータを渡さない場合は、Acrobat の標準の動作が適用されます。XML ベースの形式には <code>utf-8</code> が使用されます。他の形式の場合は、送信する値に最適なホストエンコーディングが Acrobat によって選択されます。 <code>XFDF</code> の送信ではこの値が無視され、常に <code>utf-8</code> が使用されます。

oXML	(オプション、Acrobat 7.0) このパラメータは cSubmitAs が XML の場合にのみ使用できます。これは、送信する XMLData オブジェクトです。
cPermID	(オプション、Acrobat 7.0) cSubmitAs が PDF であるか、bEmbedForm が true である場合に、送信する PDF に割り当てる永続的 ID を指定します。永続的 ID は docID 配列の最初のエントリ (docID[0]) です。 現在の文書には影響しません。
cInstID	(オプション、Acrobat 7.0) cSubmitAs が PDF であるか、bEmbedForm が true である場合に、送信する PDF に割り当てるインスタンス ID を指定します。インスタンス ID は docID 配列の 2 番目のエントリ (docID[1]) です。 現在の文書には影響しません。
cUsageRights	(オプション、Acrobat 7.0) cSubmitAs が PDF であるか、bEmbedForm が true である場合に、送信する PDF に適用する追加の使用権限を指定します。有効な値は submitFormUsageRights.RMA のみです。 現在の文書には影響しません。

例 1

サーバにフォームを送信します。

```
this.submitForm("http://www.example.com/cgi-bin/myscript.cgi#FDF");
```

例 2

選択したフォームフィールドを FDF としてサーバ側のスクリプトに送信します。

```
var aSubmitFields = new Array( "name", "id", "score" );
this.submitForm({
    cURL: "http://www.example.com/cgi-bin/myscript.cgi#FDF",
    aFields: aSubmitFields,
    cSubmitAs: "FDF" // デフォルト、ここでは不要
});
```

例 3

この例は、サブツリー全体を簡単に送信する方法を示しています。field パラメータの一部として「name」を渡すと、「name.title」、「name.first」、「name.middle」、「name.last」などが送信されます。

```
this.submitForm("http://www.example.com/cgi-bin/myscript.cgi#FDF",
    true, false, "name");
```

例 4

フォームを XFDF としてサーバ側のスクリプトに送信します。

```
this.submitForm({
    cURL: "http://www.example.com/cgi-bin/myscript.cgi#FDF",
    cSubmitAs: "XFDF"
});
```

例 5 (Acrobat 7.0)

次のスクリプトでは、PDF ファイルに添付ファイルとして含まれている XFA フォームから XML データを収集し、それらを連結します。その後、連結されたデータを送信します。

```
var oParent = event.target;
var oDataObjects = oParent.dataObjects;
if (oDataObjects == null)
    app.alert("This form has no attachments!");
else {
    var nChildren = oDataObjects.length;
    var oFirstChild = oParent.openDataObject(oDataObjects[0].name);
    var oSubmitData = oFirstChild.xfa.data.nodes.item(0).clone(true);
    for (var iChild = 1; iChild < nChildren; iChild++) {
        var oNextChild = oParent.openDataObject(
            oDataObjects[iChild].name);
        oSubmitData.nodes.append(oNextChild.xfa.data.nodes.item(0));
        oNextChild.closeDoc();
    }
    oParent.submitForm({
        cURL: "http://www.example.com/cgi-bin/myCGI.pl#FDF",
        cSubmitAs: "XML",
        oXML: oSubmitData
    });
    oFirstChild.closeDoc();
}
```

この例では、`dataObjects`、`openDataObject`、XFA オブジェクトのプロパティやメソッドを使用しています。

例 6 (Acrobat 7.0)

現在の文書を PDF として送信します。次のスクリプトでは、`cPermID`、`cInstID`、`cUsageRights` を使用しています。

```
this.submitForm({
    cUrl: myURL,
    cSubmitAs: "PDF",
    cPermID: someDoc.docID[0],
    cInstID: someDoc.docID[1],
    cUsageRights: submitFormUsageRights.RMA});
```

syncAnnotScan

5.0			
-----	--	--	--

このメソッドの処理が終わるまでに、すべての注釈が確実にスキャンされます。

文書全体の注釈を表示または処理するためには、すべての注釈が検出されている必要があります。全ページを調べて注釈を検索する処理は時間がかかるので、通常、このスキャンはバックグラウンドタスクとしてアイドル時に実行されます。バックグラウンドのスキャン処理によって注釈の完全なリストが取得されないなくても、注釈はほとんど問題なく動作します。

このメソッドは、注釈の完全なリストを必要とする場合に呼び出すのが一般的です。

[getAnnots](#) も参照してください。

例

すべての注釈がスキャンされるまで待ち、文書内の注釈の配列を取得し、作成者で並べ替えます。

このコードの 2 行目は、`syncAnnotScan` が終了するまで、つまり、文書の注釈のスキャンが完了するまで実行されません。

```
this.syncAnnotScan();  
annots = this.getAnnots({nSortBy:ANSB_Author});  
// 注釈に対して何らかの処理を行う。
```

Doc.media

各文書の `media` プロパティが参照するオブジェクトには、マルチメディア関連のプロパティやメソッドが用意されています。

Doc.media のプロパティ

canPlay

6.0			
-----	--	--	--

文書でのマルチメディア再生が許可されているかどうかを示します。再生は、信頼性管理マネージャの環境設定やその他の要因に依存します。例えば、オーサリングモードでは再生は許可されません。

`doc.media.canPlay` は、`yes` であるか `no` であるかと、`no` である場合はその理由を返します。

型

オブジェクト

アクセス

R

再生が許可される場合は、`canPlay.yes` が存在します（これは空のオブジェクトですが、今後は他の情報が含まれる可能性があります。）次のように単純なテストを行えます。

```
if( doc.media.canPlay.yes )
{
    // この文書ではマルチメディアが再生できる
}
```

再生が許可されない場合は、代わりに `canPlay.no` オブジェクトが存在します。`canPlay.yes` と同様に、単純に `canPlay.no` の存在をテストすることもできますし、またはこのオブジェクトの内容を調べて、再生が許可されない理由を確認することもできます。`canPlay.no` には、次のプロパティ（または今後プロパティが追加された場合には他のプロパティ）のうちの少なくとも 1 つが存在します。

プロパティ	説明
authoring	オーサリングモードの間は再生できません。
closing	文書が閉じているので再生できません。
saving	文書が保存されているので再生できません。
security	セキュリティ設定が原因となって再生できません。
other	他の理由で再生できません。

また、`canPlay.canShowUI` は、マルチメディアの再生ができないことを意味する警告ボックスや他のユーザインターフェイスを表示できるかどうかを示します。

例

canPlay オブジェクトを取得し、文書でメディアを再生できない理由を分析します。

```
var canPlay = doc.media.canPlay;
if( canPlay.no )
{
    // 再生できない。その理由は？
    if( canPlay.no.security )
    {
        // ユーザのセキュリティ設定で再生が禁止されている。
        // ここで警告を表示することは許可されているか？
        if( canPlay.canShowUI )
            app.alert( "Security prohibits playback" );
        else
            console.println( "Security prohibits playback" );
    }
    else
    {
        // 他の理由で再生ができない場合の処理
    }
}
```

Doc.media のメソッド

deleteRendition

6.0			
-----	--	--	--

文書から指定の名前のレンディションを削除します。以後、そのレンディションは JavaScript からアクセスできなくなります。レンディションが存在していない場合は何も行われません。

パラメータ

cName	レンディションの名前を表す文字列。
-------	-------------------

例

特定のレンディションを削除して、成功したかどうかを表示します。

```
this.media.deleteRendition("myMedia");
if ( this.media.getRendition("myMedia") == null)
    console.println( "Rendition successfully deleted" );
```

getAnnot

6.0			
-----	--	--	--

指定されたページ番号と、名前またはタイトルを使用して、文書内の ScreenAnnot を検索し、返します。一致する画面注釈がなかった場合は null を返します。名前とタイトルの両方を指定した場合には、両方ともに一致するもののみが返されます。

パラメータ

args このメソッドに渡すプロパティが含まれたオブジェクト。プロパティについては、次で説明しています。

この表では args のプロパティを示します。

nPage 注釈が存在するページ番号（0 から数えます）

cAnnotName (オプション) 画面注釈の名前。

注意：Acrobat の GUI で追加した画面注釈の場合、cAnnotName は設定されていません。

cAnnotTitle (オプション) 画面注釈のタイトル。

注意：このメソッドのパラメータは、順番付きのパラメータのリストとしてではなく、オブジェクトリテラルとして渡す必要があります。

戻り値

ScreenAnnot オブジェクト

例

Acrobat のユーザインターフェイスでは、画面注釈のタイトルは指定できますが名前は指定できません。したがって、getAnnot の典型的な使用方法は、次のようにになります。

```
var annot = myDoc.media.getAnnot
({ nPage: 0, cAnnotTitle: "My Annot Title" });
```

別の例については、[getRendition](#) の例を参照してください。

getAnnots

6.0			
-----	--	--	--

doc.media.getAnnots メソッドは、文書の指定のページにあるすべての ScreenAnnot オブジェクトの配列を返します。または、nPage が省略されている場合は、文書のすべてのページにあるすべての ScreenAnnot オブジェクトの配列を返します。ScreenAnnot がない場合、配列は空になります。

パラメータ

nPage ScreenAnnot が存在するページ番号（0 から数えます）。

戻り値

ScreenAnnot オブジェクトの配列

例

ページ 0 の ScreenAnnot のリストを取得し、そのリストからランダムに選択した画面注釈で、メディアアップを再生します。

```
var annots = this.media.getAnnots({ nPage: 0 });
var rendition = this.media.getRendition("myClip");
var settings = { windowType: app.media.windowType.docked }
var l = annots.length
var i = Math.floor( Math.random() * l ) % l
var args = { rendition:rendition, annot:annots[i], settings:settings };
app.media.openPlayer( args );
```

getOpenPlayers

7.0			
-----	--	--	--

現在開かれているメディアプレーヤーを表す `MediaPlayer` オブジェクトの配列を返します。配列内のプレーヤーは、開かれた順序でリストされています。この配列を使用して、開かれている一部またはすべてのプレーヤーを操作できます。例えば、文書で開いているすべてのプレーヤーを、そのリストを保持していくなくても、停止したり閉じたりすることができます。

`getOpenPlayers` を呼び出すたびに、配列の新しいコピーが返され、その時点で開いているプレーヤーがリストされます。その後で開かれた新しいプレーヤーは、既に返された配列には反映されません。配列に含まれているプレーヤーが閉じられても、そのプレーヤーオブジェクトは配列に残りますが、`player.isOpen` は `false` になります。`doc.media.getOpenPlayers` メソッドを再度呼び出せば、最新のプレーヤー配列が新たに取得できます。

`doc.media.getOpenPlayers` を直接使用して反復処理を行わないように注意してください。

```
for( var i in doc.media.getOpenPlayers() ) // 間違い
```

プレーヤー配列のコピーを取得して、それに対して反復処理を行います。

```
var players = doc.media.getOpenPlayers();
for( var i in players ) {
    ...
}
```

これによって、ループの実行中にプレーヤーが開いたり閉じたりしても、ループは正しく実行されます。

戻り値

`MediaPlayer` オブジェクトの配列。

例

次の 2 つの関数は `Doc` をパラメータとして取得して、`Doc` に関連付けられている実行中のプレーヤーを操作します。

```
// 実行中のすべてのプレーヤーを停止。
function stopAllPlayers( doc ) {
    var players = doc.media.getOpenPlayers();
    for( var i in players ) players[i].stop();
}
// 実行中のすべてのプレーヤーを閉じる。プレーヤーを閉じても配列からは削除
// されない。
```

```
function closeAllPlayers( doc ) {
    var players = doc.media.getOpenPlayers();
    for( var i in players )
        players[i].close( app.media.closeReason.general );
}
```

getRendition

6.0			
-----	--	--	--

指定の名前を使用して、文書内のレンディションを検索し、一致したものを返します。その名前のレンディションがない場合は `null` を返します。

パラメータ

cName cName (文字列) はレンディションの名前です。

戻り値

Rendition オブジェクト

例

次のスクリプトは、フォームボタンの「マウスボタンを放す」アクションから実行します。画面注釈で、ドッキングされたメディアクリップを再生します。

```
app.media.openPlayer({
    rendition: this.media.getRendition( "myClip" ),
    annot: this.media.getAnnot( {nPge:0,cAnnotTitle:"myScreen"} ),
    settings: { windowType: app.media.windowType.docked }
});
```

newPlayer

6.0			
-----	--	--	--

このメソッドは `MediaPlayer` オブジェクトを作成して返します。args パラメータには、`settings` プロパティが含まれている必要があります。オプションで `events` プロパティを含めることもできます。また、ユーザ定義のプロパティを含めることもできます。args のプロパティは、新しい `MediaPlayer` オブジェクトにコピーされます。このコピーは、シャローコピーです。つまり、コピーされるプロパティがオブジェクトそのものである場合、そのオブジェクトは args と新しいプレーヤーとの間で共有されます。

`newPlayer` メソッドは、標準の Acrobat メディアプレーヤーの動作に必要な標準の `EventListener` を持たない、最低限の機能のみを備えたプレーヤーを作成します。必要な `EventListener` は、`app.media.addStockEvents` を使用して追加します。

多くの場合、メディアプレーヤーの作成には `doc.media.newPlayer` よりも `app.media.createPlayer` のほうが適しています。`createPlayer` メソッドでは、標準の `EventListener` やその他のプレーヤーのプロパティが自動的に設定されます。

パラメータ

args PlayerArgs オブジェクト。

戻り値

MediaPlayer オブジェクト。

例

例については、`Events.dispatch` を参照してください。

Embedded PDF

PDF が何らかの外部アプリケーション（ホストコンテナ）のデータに埋め込まれている場合、ここで説明するインターフェイスを使用することで、外部アプリケーションは PDF を操作すること（埋め込まれた PDF からメッセージを送受信することなど）ができます。例えば、<OBJECT> タグを使用して HTML 文書に PDF 文書が埋め込まれている場合、その PDF オブジェクトは、ブラウザのスクリプト機能の処理対象となります。

このような処理を実現するために、`HostContainer` オブジェクトには、PDF スクリプトモデルに対するインターフェイスが用意されています。この通信を行うためには、セキュリティ上の理由から、コンテナと PDF 文書の双方で明示的に通信を許可する必要があります。

Embedded PDF のプロパティ

messageHandler

7.0.5			
-------	--	--	--

Web ブラウザのスクリプト環境で実行されるスクリプトは、このプロパティを使用して、通知オブジェクトを登録することができます。PDF 文書のスクリプトが `Doc` の `hostContainer.postMessage` メソッドを呼び出すると、この通知オブジェクトが呼び出されます。

このプロパティの値が表すオブジェクトでは、次のメソッドを定義できます。

メソッド	説明
<code>onMessage</code>	このメソッドを定義した場合は、 <code>Doc</code> の <code>hostContainer.postMessage</code> メソッドが呼び出されると、このメソッドが呼び出されます。メッセージの送信は非同期で行われます。このメソッドには、 <code>postMessage</code> メソッドに渡された配列を含む单一の配列パラメータが渡されます。
<code>onError</code>	このメソッドを定義した場合は、エラーに応答してこのメソッドが呼び出されます。このメソッドには、 <code>Error</code> オブジェクトと、エラーを発生させたメッセージを表す文字列の配列が渡されます。エラーが発生してもこのプロパティが定義されていない場合、エラーは送信されません（メッセージと異なり、エラーはキューに入れられません）。 <code>Error</code> オブジェクトの <code>name</code> プロパティは、次のいずれかの文字列に設定されます。 <ul style="list-style-type: none">「<code>MessageGeneralError</code>」：一般エラーが発生しました。「<code>MessageNotAllowedError</code>」：セキュリティ上の理由で操作が失敗しました。「<code>MessageDocNotDisclosedError</code>」：ホストコンテナに対して文書が公開されるように設定されていません。<code>Doc</code> の <code>hostContainer.messageHandler.onDisclose</code> プロパティが正しく初期化されている必要があります。「<code>MessageDocRefusedDisclosureError</code>」： <code>hostContainer.messageHandler.onDisclose</code> メソッドが <code>false</code> を返したため、URL に基づいてホストコンテナに文書を公開する操作が拒否されました。

これらのメソッドを呼び出した場合、`this` オブジェクトは、そのメソッドを呼び出している `messageHandler` のインスタンスになります。`messageHandler` プロパティの `on` で始まるプロパティは、通知メソッドとして将来使用するために予約されています。

このメソッドが登録される前に、PDF 文書で `postMessage` メソッドを呼び出すと、メッセージはキューに入れられます。`messageHandler` オブジェクトが設定されると、キューに入れられたすべてのメッセージがこのオブジェクトに渡されます。

メッセージはポストされた順番に送信され、エラーは発生した順番に送信されます。ただし、メッセージとエラーの送信順序に特定の関係はありません。

ハンドラメソッドで発生した例外は廃棄されます。メッセージやエラーは、`onMessage` ハンドラや `onError` ハンドラにある間は送信されません。

注意：このプロパティは、Macintosh プラットフォームでは実装されていません。

型

オブジェクト

アクセス

R / W

Embedded PDF のメソッド

postMessage

7.0.5			
-------	--	--	--

PDF 文書が公開されている場合 (`HostContainer` オブジェクトの `messageHandler` プロパティの `onDisclosed` メソッドで `true` が返される場合)、PDF 文書のメッセージハンドラに非同期でメッセージを送信します。

このメッセージは、`messageHandler` の `onMessage` メソッドに渡されます。

ホストコンテナに対して PDF 文書が公開されていない場合は、`postMessage` から処理が戻された後に、`messageHandler` プロパティの `onError` メソッドにエラーが渡されます。PDF 文書で、`Doc` の `hostContainer.messageHandler` プロパティを設定してイベントを受信するように登録していない場合は、PDF 文書でこのプロパティが設定されるまで、イベントはキューに入れられます。

メッセージは、送信されるまでメッセージのキューに入れられます。キューのサイズが最大値を超えた場合は、キュー内のいくつかのメッセージが送信されるまで、エラーが発生します。

パラメータ

`aMessage` `onMessage` に渡す 1 つ以上の文字列の配列。

注意：このメソッドは、Macintosh プラットフォームでは実装されていません。

Error

Error オブジェクトは、JavaScript に実装されているメソッドやプロパティで例外が発生するたびに、動的に作成されます。コア JavaScript では、Error オブジェクトのサブクラス (EvalError, RangeError, SyntaxError, TypeError, ReferenceError, URLError) が発生することがあります。これらのサブクラスはすべて、Error オブジェクトを継承して作成されています。JavaScript では、このような例外の一部を発生させたり、必要に応じて Error オブジェクトのサブクラスを実装したりできます。スクリプトで try/catch によるエラー処理を行っている場合、発生するオブジェクトは、次の表にリストされているタイプのいずれかになります。

Error オブジェクト	簡単な説明
RangeError	引数値が有効範囲外です。
TypeError	引数値の型が不正です。
ReferenceError	存在しない変数を読み込もうとしています。
MissingArgError	必須の引数がありません。
NumberOfArgsError	メソッドに指定されている引数の数が不正です。
InvalidSetError	設定したプロパティが無効です。または、このプロパティは設定できません。
InvalidGetError	取得したプロパティが無効です。または、このプロパティは取得できません。
OutOfMemoryError	メモリ不足です。
NotSupportedError	この構成（例えば Adobe Reader など）では、この機能はサポートされていません。
NotSupportedHFTError	HFT が使用できません（プラグインがない可能性があります）。
NotAllowedError	メソッドまたはプロパティは、セキュリティ上の理由により使用が許可されていません。
GeneralError	原因不明のエラーです。
RaiseError	Acrobat の内部エラーです。
DeadObjectError	オブジェクトが存在していません。
HelpError	ユーザがメソッドに関するヘルプを要求しました。

JavaScript で実装されている Error オブジェクト型は、コア Error オブジェクトのプロパティやメソッドを継承しています。一部の JavaScript オブジェクトは、独自の例外を実装しています。Error サブクラス（および追加のメソッドやプロパティ）については、該当するオブジェクトに関するマニュアルを参照してください。

例

Error オブジェクトのすべてのプロパティをコンソールに出力します。

```
try {
    app.alert(); // alert には引数が 1 つ必要
} catch(e) {
    for (var i in e)
        console.println( i + ":" + e[i])
}
```

Error のプロパティ

fileName

6.0			
-----	--	--	--

例外を発生させたスクリプトの名前。

型

文字列

アクセス

R

lineNumber

6.0			
-----	--	--	--

例外が発生した JavaScript コード内の不正な行の番号。

型

整数

アクセス

R

extMessage

7.0			
-----	--	--	--

例外に関する追加の詳細を提供するメッセージ。

型

文字列

アクセス

R

message

6.0			
-----	--	--	--

例外の詳細について説明するエラーメッセージ。

型

文字列

アクセス

R

name

6.0			
-----	--	--	--

Error オブジェクトサブクラスの名前。Error オブジェクトのインスタンスの型を示します。

型

文字列

アクセス

R / W

Error のメソッド

toString

6.0			
-----	--	--	--

例外の詳細について説明するエラーメッセージを取得します。

戻り値

エラーメッセージの文字列 ([message](#) プロパティを参照)。

event

すべての JavaScript スクリプトは特定のイベントの結果として実行されます。JavaScript では、イベントごとに `event` オブジェクトが作成されます。各イベントの発生時にはこのオブジェクトにアクセスすることができます。イベントの現在の状態を取得したり操作したりすることができます。

各イベントは、`type` と `name` の組み合わせによって一意に識別されます。ここでは、すべてのイベント（タイプと名前の組み合わせ）について説明します。追加のプロパティが定義される場合はそれも示します。

イベントの `rc` プロパティは戻りコードです。各イベントの説明では、イベントが戻りコードに応答する（影響を受ける）かどうかを示します。

JavaScript の作成時には、イベントが発生するタイミングとその処理順序を理解しておく必要があります。一部のメソッドやプロパティは、特定のイベントでしかアクセスできません。

イベントのタイプと名前の組み合わせ

App / Init

このイベント（アプリケーション初期化イベント）は、ビューアの起動時に発生します。スクリプトファイル（フォルダレベルの JavaScript と呼ばれます）が、アプリケーションやユーザの JavaScript フォルダから読み込まれます。スクリプトファイルは、`config.js`、`glob.js`、他のすべてのアプリケーションレベルの JavaScript ファイル、ユーザレベルの JavaScript ファイルの順にロードされます。

このイベントは、`rc` 戻りコードに応答しません。

Batch / Exec

5.0			
-----	--	--	--

このイベントは、バッチシーケンスで各文書が処理される時に発生します。バッチシーケンスの一部として作成されたスクリプトでは、実行時にこの `event` オブジェクトにアクセスすることができます。

このイベントの `target` は、`Doc` です。

このイベントは、`rc` 戻りコードに応答します。`rc` が `false` の場合、バッチシーケンスは停止します。

Bookmark / Mouse Up

5.0			
-----	--	--	--

このイベントは、JavaScript を実行するしおりをユーザがクリックするたびに発生します。

このイベントの `target` は、クリックされた `bookmark` オブジェクトです。

このイベントは、`rc` 戻りコードに応答しません。

Console / Exec

5.0			
-----	--	--	--

このイベントは、ユーザがコンソールで JavaScript を実行するたびに発生します。

このイベントは、rc 戻りコードに応答しません。

Doc / DidPrint

5.0			
-----	--	--	--

このイベントは、文書の印刷後に発生します。

このイベントの target は、Doc です。

このイベントは、rc 戻りコードに応答しません。

Doc / DidSave

5.0			
-----	--	--	--

このイベントは、文書の保存後に発生します。

このイベントの target は、Doc です。

このイベントは、rc 戻りコードに応答しません。

Doc / Open

4.0			
-----	--	--	--

このイベントは、文書を開くたびに発生します。文書を開くと、文書レベルのスクリプト関数がスキャンされ、公開されているスクリプトが実行されます。

このイベントの target は、Doc です。このイベントでは、`targetName` プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Doc / WillClose

5.0			
-----	--	--	--

このイベントは、文書を閉じる前に発生します。

このイベントの target は、Doc です。

このイベントは、rc 戻りコードに応答しません。

Doc / WillPrint

5.0			
-----	--	--	--

このイベントは、文書の印刷前に発生します。

このイベントの `target` は、`Doc` です。

このイベントは、`rc` 戻りコードに応答しません。

Doc / WillSave

5.0			
-----	--	--	--

このイベントは、文書の保存前に発生します。

このイベントの `target` は、`Doc` です。

このイベントは、`rc` 戻りコードに応答しません。

External / Exec

5.0			
-----	--	--	--

このイベントは、OLE や AppleScript などを使用した外部アクセス、または FDF をロードした結果として発生します。

このイベントは、`rc` 戻りコードに応答しません。

Field / Blur

4.05			
------	--	--	--

このイベントは、フィールドがフォーカスを失うときに、すべてのイベントの最後に発生します。フォーカスの消失がマウスクリックによるものでなくても（例えば、`Tab` キーが使用された場合でも）このイベントは発生します。

定義されている追加プロパティ：

- `target` : 検証スクリプトを実行しているフィールド。
- `modifier`、`shift`、`targetName`、`value`。

このイベントは、`rc` 戻りコードに応答しません。

Field / Calculate

3.01			
------	--	--	--

このイベントは、フォームに加えられた変更によって、計算スクリプトが割り当てられているすべてのフィールドで計算が実行されるときに発生します。変更されたフィールドの値に依存しているフィールドは、

この時点ですべて再計算されます。それらのフィールドでは、`Field／Validate`、`Field／Blur`、`Field／Focus` の各イベントが順次発生します。

計算フィールドは、他の計算フィールドの計算結果に依存していることがあります。計算順序の配列には、文書内で計算スクリプトが割り当てられているすべてのフィールドが順番にリストされています。フィールド全体の計算が必要とされる場合、この配列インデックスの 0 番目から最後まで、各フィールドが順に計算されます。

フィールドの計算順序を変更するには、「フォームを編集」モードにし、フォーム／フィールドを編集／フィールドの計算順序の設定というメニュー項目を使用します。

このイベントの `target` は、計算スクリプトを実行しているフィールドです。このイベントでは、`source` および `targetName` プロパティも定義されます。

このイベントは、`rc` 戻りコードに応答します。戻りコードが `false` に設定されるとフィールドの値は変更されず、`true` に設定されると `value` プロパティの値が使用されます。

Field／Focus

4.05			
------	--	--	--

このイベントは、フィールドをマウスでクリックしてフォーカスを移した後、マウスボタンを放す前に発生します。このイベントは、フィールドをアクティブにしたのがマウスクリックでなくても（例えば、`Tab` キーが使用された場合でも）発生します。ユーザがフィールドを操作する前に実行したい前処理がある場合は、このイベントで実行するのが最適です。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`modifier`、`shift`、`targetName` の各プロパティも定義されます。

このイベントは、`rc` 戻りコードに応答しません。

Field／Format

3.01			
------	--	--	--

このイベントは、関連する計算がすべて終了した後に発生します。このイベントで、フィールドに割り当てる `JavaScript` を実行して、データ値の表示方法（表現または外観）を変更することができます。例えばデータ値が数値で、これを通貨として表示したい場合、フォーマットスクリプトで値の前にドル記号 (\$) を付け、小数点以下 2 衔までに制限することができます。

このイベントの `target` は、フォーマットスクリプトを実行しているフィールドです。このイベントでは、`commitKey`、`targetName`、`willCommit` の各プロパティも定義されます。

このイベントは、`rc` 戻りコードに応答しません。ただし、`value` を使用してフィールドの値をフォーマットすることができます。

Field / Keystroke

3.01			
------	--	--	--

このイベントは、テキストボックスやコンボボックスでユーザがキーストローク（テキストをカット＆ペーストする場合も含みます）を行うか、コンボボックスやリストボックスで項目を選択するたびに発生します。キーストロークスクリプトは、ユーザが使用できるキーのタイプを制限したい場合などに使用できます。例えば、数値フィールドでは、数値のみを入力可能にすることができます。

リストボックスでは、ユーザインターフェイスから `Selection Change` スクリプトを割り当てることができます。このスクリプトは、項目が選択されるたびに実行されますが、この動作はキーストロークイベントとして実装されており、ユーザの選択した項目がキーストローク値に相当します。コンボボックスも同様で、ドロップダウンリストで選択した値がテキストフィールドに貼り付けられる動作を「キーストローク」と考えることができます。

キーストロークスクリプトの最後の呼び出しが行われた後で、検証イベントが発生し、`willCommit` が `true` に設定されます。値が確定される前に、キーストロークが行われている段階でフィールド値の確認を行うと便利なことがあります。このようにすると、キーストローク単位で部分的にしかチェックできないような、特に複雑なフォーマットを効果的に処理することができます。

テキストフィールドの `keystroke` イベントは、ユーザがキーボードでテキストを入力したときや、フィールド値を確定したときに呼び出されますが、それ以外の状況で呼び出されることもあります。UI や JavaScript によってデフォルト値がフィールドに設定されたときや、オートフィルによってエントリの入力が行われたときにも呼び出され、検証を行います。このような場合、イベントのすべてのプロパティが定義されるわけではありません。具体的には、デフォルト値を検証する場合は `event.target` が `undefined` になり、オートフィルのエントリを検証する場合は `event.richChange` と `event.richValue` が `undefined` になります。

このイベントの `target` は、キーストロークスクリプトを実行しているフィールドです。このイベントでは、`commitKey`、`change`、`changeEx`、`keyDown`、`modifier`、`selEnd`、`selStart`、`shift`、`targetName`、`value`、`willCommit` の各プロパティも定義されます。

このイベントは、`rc 戻りコード` に応答します。`false` に設定されていると、キーストロークは無視されます。入力された文字をスクリプトで置換したい場合は、`change` をキーストロークとして使用できます。フィールド内の現在のテキスト選択範囲は、`selEnd` と `selStart` プロパティで変更できます。

Field / Mouse Down

3.01			
------	--	--	--

このイベントは、フォームフィールドがクリックされて、マウスボタンがまだ押されているときに発生します。`mouse enter` イベントがまだ発生していない場合、`mouse down` イベントは発生しません。このイベントで実行する処理はできるだけ少なくしてください（短い音を鳴らすなど）。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`modifier`、`shift`、`targetName` の各プロパティも定義されます。

このイベントは、`rc 戻りコード` に応答しません。

Field / Mouse Enter

3.01			
------	--	--	--

このイベントは、フィールドを囲む矩形内にポインタが移動したときに発生します。このイベントは、テキストフィールドにヘルプを表示する処理などに適しています。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`modifier`、`shift`、`targetName` の各プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Field / Mouse Exit

3.01			
------	--	--	--

このイベントは、フィールドを囲む矩形からマウスポインタが出たときに発生します。`mouse enter` イベントがまだ発生していない場合、`mouse exit` イベントは発生しません。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`modifier`、`shift`、`targetName` の各プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Field / Mouse Up

3.01			
------	--	--	--

このイベントは、フォームフィールドをクリックして、マウスボタンを放したときに発生します。このイベントは、フォームを送信するアクションなどを割り当てるのに適しています。`mouse down` イベントがまだ発生していない場合、`mouse up` イベントは発生しません。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`modifier`、`shift`、`targetName` の各プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Field / Validate

3.01			
------	--	--	--

ユーザは、様々なフィールドで新しい値を入力します。ユーザがフィールドの外側をクリックしたり、別のフィールドに Tab キーで移動したり、Enter キーを押したりすると、入力された新しいデータ値が確定されます。

このイベントは、値が確定された後に最初に発生するイベントです。これを利用すれば、正しい値が入力されたかどうかを JavaScript で確認することができます。`validate` イベントが正常に処理されると、次のイベントである `calculate` イベントが発生します。

このイベントの `target` は、検証スクリプトを実行しているフィールドです。このイベントでは、`change`、`changeEx`、`keyDown`、`modifier`、`shift`、`targetName`、`value` の各プロパティも定義されます。

このイベントは、`rc 戻りコード`に応答しません。戻りコードが `false` に設定されていると、フィールドの値は無効であると見なされ、値は変更されません。

Link / Mouse Up

5.0			
-----	--	--	--

このイベントは、JavaScript アクションを含むリンクをユーザがアクティブにすると発生します。

このイベントの `target` は、`Doc` です。

このイベントは、`rc 戻りコード`に応答しません。

Menu / Exec

5.0			
-----	--	--	--

このイベントは、メニュー項目に割り当てられた JavaScript が実行されるたびに発生します。ユーザは、メニュー項目を追加して、それに JavaScript アクションを関連付けることができます。例えば、次のようにします。

```
app.addMenuItem({ cName: "Hello", cParent: "File",
  cExec: "app.alert('Hello',3);", nPos: 0});
```

`app.alert('Hello',3)` というスクリプトは、メニューイベントで実行されます。メニューイベントは、次の 2 つの場合に発生します。

- ユーザがユーザインターフェイスでメニュー項目を選択した場合。
- プログラムで（例えば、ボタンフィールドの `mouse up` イベントなどで）
`app.execMenuItem("Hello")` が実行された場合。

イベントの `target` は、現在アクティブな文書がある場合、その文書です。このイベントでは、`targetName` プロパティも定義されます。

このイベントは、マークの付いた使用可能なプロシージャがメニュー項目に設定されている場合、`rc 戻りコード`に応答します（`app.addMenuItem` の `cEnabled` および `cMarked` パラメータを参照）。戻りコードが `false` の場合は、メニュー項目が使用不可になるか、マークが解除されます。戻りコードが `true` の場合は、メニュー項目が使用可能になるか、マークが付けられます。

Page / Open

4.05			
------	--	--	--

このイベントは、新しいページが表示されてページが描画された後に発生します。

このイベントの `target` は、`Doc` です。

このイベントは、`rc 戻りコード`に応答しません。

Page / Close

4.05			
------	--	--	--

このイベントは、表示されるページが現在のページでなくなるとき（新しいページに移動したときや、文書を閉じたときなど）に発生します。

このイベントの `target` は、`Doc` です。

このイベントは、`rc 戻りコード` に応答しません。

Screen / Blur

6.0			
-----	--	--	--

このイベントは、画面注釈がフォーカスを失うときに、すべてのイベントの最後に発生します。フォーカスの消失がマウスクリックによるものでなくとも（例えば、`Tab` キーが使用された場合でも）このイベントは発生します。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier` および `shift` プロパティも定義されます。

このイベントは、`rc 戻りコード` に応答しません。

Screen / Close

6.0			
-----	--	--	--

このイベントは、表示されるページが現在のページでなくなるとき（新しいページに移動したときや、文書を閉じたときなど）に発生します。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier`、`shift`、`target` の各プロパティも定義されます。

このイベントは、`rc 戻りコード` に応答しません。

Screen / Focus

6.0			
-----	--	--	--

このイベントは、フィールドをマウスでクリックしてフォーカスを移した後、マウスボタンを放す前に発生します。このイベントは、画面注釈をアクティブにしたのがマウスクリックでなくとも（例えば、`Tab` キーが使用された場合でも）発生します。ユーザがフィールドを操作する前に実行したい前処理がある場合は、このイベントで実行するのが最適です。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier` および `shift` プロパティも定義されます。

このイベントは、`rc 戻りコード` に応答しません。

Screen / InView

6.0			
-----	--	--	--

このイベントは、新しいページが最初に表示されるときに発生します。ページレイアウトが「連続ページ」または「連続見開きページ」に設定されている場合、Screen / Open イベントの前にこのイベントが発生します。

このイベントの target は、このイベントを開始した ScreenAnnot オブジェクトです。targetName は、画面注釈のタイトルです。このイベントでは、modifier および shift プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / Mouse Down

6.0			
-----	--	--	--

このイベントは、画面注釈がクリックされて、マウスボタンがまだ押されているときに発生します。このイベントで実行する処理はできるだけ少なくしてください（短い音を鳴らすなど）。mouse enter イベントがまだ発生していない場合、mouse down イベントは発生しません。

このイベントの target は、このイベントを開始した ScreenAnnot オブジェクトです。targetName は、画面注釈のタイトルです。このイベントでは、modifier および shift プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / Mouse Enter

6.0			
-----	--	--	--

このイベントは、画面注釈を囲む矩形内にマウスポインタが移動したときに発生します。

このイベントの target は、このイベントを開始した ScreenAnnot オブジェクトです。targetName は、画面注釈のタイトルです。このイベントでは、modifier および shift プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / Mouse Exit

6.0			
-----	--	--	--

このイベントは Mouse Enter イベントの逆で、画面注釈を囲む矩形からマウスポインタが出たときに発生します。Mouse Enter イベントがまだ発生していない場合、Mouse Exit イベントは発生しません。

このイベントの target は、このイベントを開始した ScreenAnnot オブジェクトです。targetName は、画面注釈のタイトルです。このイベントでは、modifier および shift プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / Mouse Up

6.0			
-----	--	--	--

このイベントは、画面注釈をクリックして、マウスボタンを放したときに発生します。このイベントは、マルチメディアクリップの開始などを割り当てるのに適しています。`mouse down` イベントがまだ発生していない場合、`mouse up` イベントは発生しません。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier` および `shift` プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / Open

6.0			
-----	--	--	--

このイベントは、新しいページが表示されてページが描画された後に発生します。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier` および `shift` プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

Screen / OutView

6.0			
-----	--	--	--

このイベントは、ページが最初に表示されなくなるときに発生します。ページレイアウトが「連続ページ」または「連続見開きページ」に設定されている場合、`Screen/Close` イベントの後にこのイベントが発生します。

このイベントの `target` は、このイベントを開始した `ScreenAnnot` オブジェクトです。`targetName` は、画面注釈のタイトルです。このイベントでは、`modifier` および `shift` プロパティも定義されます。

このイベントは、rc 戻りコードに応答しません。

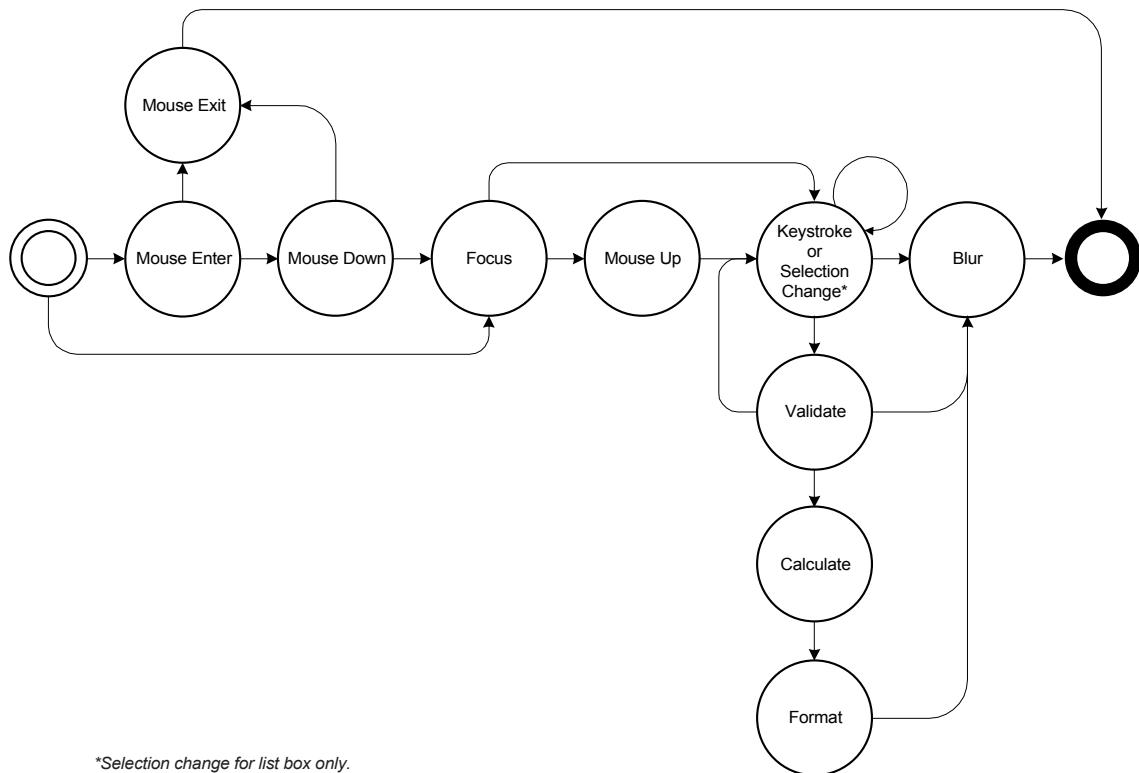
文書イベントの処理

文書を開くと、`Doc/Open` イベントが発生し、関数がスキャンされ、公開されている（トップレベルの）スクリプトがすべて実行されます。次に、PDF ファイルの `NeedAppearances` エントリが `AcroForm` ディクショナリで `true` に設定されている場合は、文書内のすべてのフィールドのフォーマットスクリプトが実行されます（『PDF Reference』バージョン 1.7 を参照）。最後に、`Page/Close` イベントが発生します。

注意：`NeedAppearances` エントリを `true` に設定して、フォームフィールドを含む PDF ファイルを作成した場合は、Web サイトなどで公開する前に、「名前を付けて保存（別名で保存）」を実行してください。「名前を付けて保存（別名で保存）」を実行すると、フォームの外観が生成されてファイルに保存されます。これにより、Web ブラウザにファイルをロードする Adobe Reader のパフォーマンスが向上します。

フォームイベントの処理

次の図に、フォームイベントの発生順序を示します。各イベントの依存関係に注意してください。例えば、Focus イベントが発生していないければ、Mouse Up イベントが発生することはありません。



マルチメディアイベントの処理

マルチメディアに関するイベントが発生して EventListener にディスパッチされるたびに、マルチメディア event オブジェクトがパラメータとして EventListener に渡されます。このオブジェクトは、Acrobat の他の場所で使用される event オブジェクトと似ており、次のプロパティを持ちます。

レンディションアクション（例えば、マルチメディアのプロパティパネルの「アクション」タブで入力したカスタムの JavaScript）によって発生したマルチメディア event オブジェクトには、次のプロパティも含まれます。

action.annot	このイベントの ScreenAnnot オブジェクト。
action.rendition	このイベントの Rendition オブジェクト。

標準のマルチメディアイベントディスパッチャによってディスパッチされたマルチメディア event オブジェクトには、次のプロパティも含まれます。独自の events.dispatch メソッドを使用した場合は、これらのプロパティは含まれません。

media.doc	文書。target.doc と同じ。
media.events	events オブジェクト。target.events と同じ。
media.id	event.name のコピー（スペースは削除）。

個々のイベントには、追加のプロパティが含まれることがあります。詳しくは、`EventListener` オブジェクトの各メソッドの説明を参照してください。

標準イベントディスパッチャによって呼び出されたイベントメソッドは、次のいずれかのプロパティを設定して、それ以上のイベントのディスパッチを停止させることができます。

```
stopDispatch  
stopAllDispatch
```

現在のイベントが他の `EventListener` にディスパッチされるのを停止するには、イベントメソッドで `event.stopDispatch` を `true` に設定します。これを `on` イベントメソッドで実行した場合、イベントの `on` メソッドはそれ以上呼び出されなくなりますが、`after` メソッドはそれ以降も呼び出されます。`event.stopAllDispatch` を設定した場合は、いずれのタイプのイベントメソッドもそれ以上呼び出されなくなります。`on` および `after` `EventListener` について詳しくは、`EventListener` オブジェクトを参照してください。

event のプロパティ

change

3.01			
------	--	--	--

ユーザ入力による値の変更を示す文字列。JavaScript で、この文字列の一部またはすべてを別の文字列に置き換えることができます。値の変更は、個々のキーストロークや、文字列の入力（フィールドに貼り付けを行った場合など）を表します。

型

文字列

アクセス

R / W

例

すべてのキーストロークを大文字に変更します。

```
// テキストフィールドのカスタムキーストローク  
event.change = event.change.toUpperCase();
```

changeEx

5.0			
-----	--	--	--

変更の書き出し値が含まれます。これは、リストボックスおよびコンボボックスの `Field / Keystroke` イベントでのみ使用できます。

リストボックスの場合は、プロパティダイアログボックスの「選択の変更」タブでキーストローククリップトを入力できます。

コンボボックスの場合、`changeEx` は、ドロップダウンリストを使用している場合（マウスやキーボードで項目を選択している場合）にのみ使用できます。編集可能なコンボボックスでエントリを入力している場合、`Field / Keystroke` イベントはテキストフィールドの場合と同じ動作になります（つまり、`changeEx` または `keyDown` イベントプロパティはありません）。

Acrobat 6.0 以降では、`event.changeEx` はテキストフィールドにも定義されています。`event.fieldFull` が `true` の場合、`changeEx` はユーザが入力しようとしたテキスト文字列全体を示し、`event.change` はフィールドに収まる範囲にトリミングされたテキスト文字列になります。リッチテキストフィールドを扱う場合は、`event.richChangeEx`（および `event.richChange`）を使用します。

型

各種

アクセス

R

例 1

これは、HTML 形式の単純なオンラインヘルプファイルシステムの例です。

次のように、プログラムでコンボボックスを作成します。

```
var c = this.addField({
    cName: "myHelp",
    cFieldType: "combobox",
    nPageNum: 0,
    oCoords: [72, 12+3*72, 3*72, 0+3*72]
})
```

コンボボックスの各項目を設定します。

```
c.setItems([
    ["Online Help", "http://www.example.com/myhelp.html"],
    ["How to Print", "http://www.example.com/myhelp.html#print"],
    ["How to eMail", "http://www.example.com/myhelp.html#email"]
]);
```

アクションを設定します。

```
c.setAction("Keystroke", "getHelp()");
```

この関数は文書レベルで定義します。

```
function getHelp() {
    if ( !event.willCommit && (event.changeEx != "") )
        app.launchURL(event.changeEx);
}
```

例 2

テキストフィールドで `changeEx` を使用する例については、[fieldFull](#) の例を参照してください。

commitKey

4.0			
-----	--	--	--

どのようにしてフォームフィールドからフォーカスが失われたかを判定します。有効な値は、次のとおりです。

- 0 — 値が確定されませんでした（例えば、Esc キーが押された場合）。
- 1 — フィールドの外側がクリックされたことにより、値が確定されました。
- 2 — Enter キーにより、値が確定されました。
- 3 — Tab キーによって別のフィールドに移動したことで、値が確定されました。

型

数値

アクセス

R

例

値の確定後に警告ダイアログボックスを自動的に表示するには、次のコードをフィールドのフォーマットスクリプトに追加します。

```
if (event.commitKey != 0)
    app.alert("Thank you for your new field value.");
```

fieldFull

6.0			
-----	--	--	--

スペースの制限または文字数の上限によって、入力しようとするテキストがフィールドに収まらない場合、true が設定されます（Field オブジェクトの doNotScroll プロパティが true である場合はスペースの制限が、Field オブジェクトの charLimit プロパティが正の値である場合は文字数の上限が設定されます）。fieldFull が true の場合、event.changeEx はユーザが入力しようとしたテキスト文字列全体を示し、event.change はフィールドに収まる範囲にトリミングされたテキスト文字列になります。

テキストフィールドのキーストロークイベントでのみ使用できます。

型

ブーリアン

アクセス

R

イベント

Keystroke

例 1

文字数に上限があるテキストフィールドに対する、カスタムキーストロークスクリプトの例を次に示します。フィールドがいっぱいになるか、入力したデータをユーザが確定すると、次のフィールドにフォーカスが移動します。

```
if ( event.fieldFull || event.willCommit )
    this.getField("NextTabField").setFocus();
```

例 2

ユーザの入力がテキストフィールドに収まるかどうかをテストします。テキストフィールドに対するカスタムキーストロークスクリプトを示します。このフィールドでは、テキストがスクロールできないように設定されています。

```
if ( event.fieldFull )
{
    app.alert("You've filled the given space with text,"
        + " and as a result, you've lost some text. I'll set the field to"
        + " scroll horizontally, and paste in the rest of your"
        + " missing text.");
    this.resetForm([event.target.name]);           // フィールドをリセットしてフォーカスを削除
    event.target.doNotScroll = false;             // 変更を行う
    event.change = event.changeEx;
}
```

通常、キーストロークイベント中にフィールドプロパティを変更することはできません。データを確定するには、フォーカスをフィールドから移動させる必要があります。データの入力を続けるには、データを確定してから再度フォーカスを設定します。

keyDown

5.0			
-----	--	--	--

リストボックスとコンボボックスのキーストロークイベントでのみ使用可能です。リストボックスまたはコンボボックスのドロップダウンリストで、選択を行うときに矢印キーを使用した場合は `true` になり、それ以外の場合は `false` になります。

コンボボックスの場合、`keyDown` は、コンボボックスのドロップダウンリストを使用している場合（マウスやキーボードで項目を選択している場合）にのみ使用できます。編集可能なコンボボックスでエントリを入力している場合、`Field / Keystroke` イベントはテキストフィールドの場合と同じ動作になります（つまり、`changeEx` または `keyDown` イベントプロパティはありません）。

型

ブーリアン

アクセス

R

modifier

3.01			
------	--	--	--

特定のイベントで修飾キーが押されているかどうかを示します。Microsoft Windows プラットフォームの修飾キーは Ctrl キーで、Macintosh プラットフォームの修飾キーは Option または Command キーです。このプロパティは、UNIX ではサポートされていません。

型

ブーリアン

アクセス

R

name

4.05			
------	--	--	--

現在のイベント名を示すテキスト文字列。イベントは、`type` と `name` で一意に識別されます。有効な名前は、次のとおりです。

Keystroke	Mouse Exit
Validate	WillPrint
Focus	DidPrint
Blur	WillSave
Format	DidSave
Calculate	Init
Mouse Up	Exec
Mouse Down	Open
Mouse Enter	Close

型

文字列

アクセス

R

イベント

すべて

rc

3.01			
------	--	--	--

検証を行う場合に使用します。一連のイベント中で、特定のイベントの発生を許可するかどうかを示します。変更や、値の確定を不可にするには、`false` に設定します。デフォルトは `true` です。

型

ブーリアン

アクセス

R / W

イベント

Keystroke、Validate、Menu

richChange

6.0			
-----	--	--	--

ユーザ入力による、値の変更を示します。`richChange` プロパティはリッチテキストフィールドにのみ定義されており、`event.change` プロパティと同じように動作します。`richChange` の値は `Span` オブジェクトの配列であり、フィールドに入力されたテキストおよび書式の両方を表します。リッチテキストフィールドをキーボードからの入力で更新した場合、キーストロークは、単一メンバの配列として表されます。一方、リッチテキストフィールドに文字列をペーストした場合、その変更値は任意の長さの配列として表されます。

`event.fieldFull` が `true` の場合、`richChangeEx` はユーザが入力しようとしたリッチフォーマットのテキスト文字列全体を示し、`event.richChange` はフィールドに収まる範囲にトリミングされたリッチフォーマットのテキスト文字列になります。プレーンテキストフィールドを扱う場合は、`event.changeEx` (および `event.change`) を使用します。

関連するオブジェクトやプロパティとしては、`event.richValue`、`Span` オブジェクト、`Field` オブジェクトの `defaultStyle`、`richText`、`richValue` プロパティ、`Annotation` オブジェクトの `richContents` プロパティがあります。

型

`Span` オブジェクトの配列

アクセス

R / W

イベント

Keystroke

例

この例では、キーストロークを大文字に変更し、テキスト色を交互に青と赤に変更し、下線のオン／オフを切り替えます。

```
// リッチテキストフィールドのカスタムのキーストローク。
var span = event.richChange;
for ( var i=0; i<span.length; i++)
{
    span[i].text = span[i].text.toUpperCase();
    span[i].underline = !span[i].underline;
    span[i].textColor = (span[i].underline) ? color.blue : color.red;
}
event.richChange = span;
```

richChangeEx

6.0			
-----	--	--	--

このプロパティは、リッチテキストフィールドにのみ定義されており、テキストフィールドの `event.changeEx` プロパティと同じように動作します。値は `Span` オブジェクトの配列であり、フィールドに入力されたテキストおよび書式の両方を表します。リッチテキストフィールドをキーボードからの入力で更新した場合、キーストロークは、単一メンバの配列として表されます。一方、リッチテキストフィールドに文字列をペーストした場合、その変更値は任意の長さの配列として表されます。

`event.fieldFull` が `true` の場合、`richChangeEx` はユーザが入力しようとしたリッチフォーマットのテキスト文字列全体を示し、`event.richChange` はフィールドに収まる範囲にトリミングされたリッチフォーマットのテキスト文字列になります。プレーンテキストフィールドを扱う場合は、`event.changeEx` (および `event.change`) を使用します。

関連するオブジェクトやプロパティとしては、`event.richChange`、`event.richValue`、`Span` オブジェクト、`Field` オブジェクトの `defaultStyle`、`richText`、`richValue` プロパティ、`Annotation` オブジェクトの `richContents` プロパティがあります。

型

`Span` オブジェクトの配列

アクセス

R / W

イベント

Keystroke

例

ユーザの入力によってテキストフィールドがいっぱいになった場合は、フィールドをスクロールできるように設定し、テキストを追加できるようにします。

```
if ( event.fieldFull )
{
    app.alert("You've filled the given space with text,"
        + " and as a result, you've lost some text. I'll set the field to"
```

```
+ " scroll horizontally, and paste in the rest of your"
+ " missing text.");
this.resetForm([event.target.name]);           // フィールドをリセットしてフォーカスを削除
event.target.doNotScroll = false;             // 変更を行う
if ( event.target.richText )
    event.richChange = event.richChangeEx
else
    event.change = event.changeEx;
}
```

[event.fieldFull](#) も参照してください。

richValue

6.0			
-----	--	--	--

このプロパティは、Field オブジェクトの richValue プロパティや、各イベントの event.value プロパティと同じです。

関連するオブジェクトやプロパティとしては、[Span](#) オブジェクト、Field オブジェクトの [defaultStyle](#)、[richText](#)、[richValue](#) プロパティ、[event.richChange](#)、[event.richChangeEx](#)、Annotation オブジェクトの [richContents](#) プロパティがあります。

型

Span オブジェクトの配列

アクセス

R / W

イベント

Keystroke

例

すべての太字のテキストを下線付きの赤いテキストに変換します。

```
// リッチテキストフィールドのカスタムフォーマットイベント。
var spans = event.richValue;
for ( var i = 0; i < spans.length; i++ )
{
    if( spans[i].fontWeight >= 700 )
    {
        spans[i].textColor = color.red;
        spans[i].fontWeight = 400;           // デフォルトの太さに変更
        spans[i].underline = true;
    }
}
event.richValue = spans;
```

selEnd

3.01			
------	--	--	--

キーストロークイベントで現在テキストが選択されている範囲の終了位置。

型

整数

アクセス

R / W

例

(テキストフィールドで) 最後に行われた変更に未確定の変更を結合します。この関数では、selEnd と selStart の両方を使用しています。

```
function AFMergeChange(event)
{
    var prefix, postfix;
    var value = event.value;

    if(event.willCommit) return event.value;
    if(event.selStart >= 0)
        prefix = value.substring(0, event.selStart);
    else prefix = "";
    if(event.selEnd >= 0 && event.selEnd <= value.length)
        postfix = value.substring(event.selEnd, value.length);
    else postfix = "";
    return prefix + event.change + postfix;
}
```

selStart

3.01			
------	--	--	--

キーストロークイベントで現在テキストが選択されている範囲の開始位置。

型

整数

アクセス

R / W

例

[selEnd](#) の例を参照してください。

shift

3.01			
------	--	--	--

特定のイベントで Shift キーが押されている場合は `true`、押されていない場合は `false`。

型

ブーリアン

アクセス

R

例

次のコードは、「マウスボタンを放す」アクションです。

```
if (event.shift)
    this.gotoNamedDest ("dest2");
else
    this.gotoNamedDest ("dest1");
```

source

5.0			
-----	--	--	--

計算イベントを発生させた Field オブジェクト。このオブジェクトは通常、イベントの `target` (計算の対象となっているフィールド) とは異なります。

型

オブジェクト

アクセス

R

target

3.01			
------	--	--	--

イベントを発生させたターゲットオブジェクト。`mouse`、`focus`、`blur`、`calculate`、`validate`、`format` のすべてのイベントでは、イベントを発生させた Field オブジェクトになります。`page open` や `page close` などの他のイベントでは、`Doc` または `this` オブジェクトになります。

型

オブジェクト

アクセス

R

targetName

5.0			
-----	--	--	--

実行されている JavaScript の名前を返そうと試みます。デバッグ時に、例外を発生させたコードを識別する場合に使用できます。`targetName` の一般的な値は、次のとおりです。

- App / Init イベントのフォルダレベルスクリプトのファイル名
- Doc / Open イベントの文書レベルスクリプトの名前
- Batch / Exec イベントで処理された PDF ファイル名
- Field イベントのフィールド名
- Menu / Exec イベントのメニュー項目名
- Screen イベントの画面注釈名（マルチメディアイベント）

例外が発生した場合、識別可能な名前があると `targetName` が通知されます。

型

文字列

アクセス

R

例

`conserve.js` というフォルダレベルの JavaScript ファイルの最初の行にエラーが含まれています。ビューアが起動すると例外が発生し、メッセージに問題の原因が示されます。

```
MissingArgError: Missing required argument.  
App.alert:1:Folder-Level:App:conserve.js  
====> Parameter cMsg.
```

type

5.0			
-----	--	--	--

現在のイベントのタイプ。イベントは、`type` と `name` で一意に識別されます。有効な値は、次のとおりです。

Batch	External
Console	Bookmark
App	Link
Doc	Field
Page	Menu

型

文字列

アクセス

R

value

3.01			
------	--	--	--

このプロパティは、それぞれの `field` イベントによって意味が異なります。

- `Field` / `Validate` イベントの場合は、フィールドが確定されたときの値になります。コンボボックスの場合は、書き出し値 ([changeEx](#) を参照) ではなく、表示値になります。

例えば、次の JavaScript は、フィールドの値が 0 ~ 100 であることを検証します。

```
if (event.value < 0 || event.value > 100) {
    app.beep(0);
    app.alert("Invalid value for field " + event.target.name);
    event.rc = false;
}
```

- `Field` / `Calculate` イベントの場合、このプロパティを JavaScript で設定すると、イベント完了時のフィールド値になります。

例えば、次の JavaScript は、フィールドの値を `SubTotal` フィールドの値 + 税金に設定します。

```
var f = this.getField("SubTotal");
event.value = f.value * 1.0725;
```

- `Field` / `Format` イベントの場合、このプロパティを JavaScript で設定すると、フィールド値をフォーマットすることができます。デフォルトでは、このプロパティはユーザが確定した値を含みます。コンボボックスの場合は、書き出し値 ([changeEx](#) を参照) ではなく、表示値になります。

例えば、次の JavaScript は、フィールドの値を通貨の形式にフォーマットします。

```
event.value = util.printf("$%.2f", event.value);
```

- `Field / Keystroke` イベントの場合は、フィールドの現在の値です。例えば、テキストフィールドを変更中の場合は、キーストロークが適用される直前の、テキストフィールドのテキストを示します。
- `Field / Blur` や `Field / Focus` イベントの場合は、フィールドの現在の値です。この 2 つのイベントでは、`event.value` は読み取り専用です。つまり、`event.value` を設定してフィールド値を変更することはできません。

Acrobat 5.0 以降では、複数選択が可能なリストボックス (`field.multipleSelection` を参照) で、フィールド値が配列になっている場合 (複数の項目が選択されている場合)、`event.value` の値を取得しようとすると空の文字列が返されます。この場合は、値の設定も行えません。

型

各種

アクセス

R / W

willCommit

3.01			
------	--	--	--

データが確定される前に現在のキーストロークを検証します。例えば、数値データ用のフォームフィールドに文字データが入力されなかったかなどを確認するのに便利です。このプロパティは、最後の `keystroke` イベントが発生した後、フィールドが検証される前に、`true` に設定されます。

型

ブーリアン

アクセス

R

例

この例は、キーストロークイベントの構造を示しています。

```
var value = event.value
if (event.willCommit)
    // 値の最終的な確認。
else
    // キーストロークレベルでの確認。
```

EventListener

このオブジェクトは、オプションのローカルデータを伴うマルチメディアイベントメソッドの集合です。イベントメソッド名は `on` または `after` で始まり、その後にイベント名が続きます (`onPause` や `afterPause` など)。イベントがディスパッチされると、対応する `on` イベントメソッドが直ちに呼び出され、その少し後の次のアイドル時間に、対応する `after` イベントメソッドが呼び出されます。

`on` イベントメソッドには、次のような制限があります。

- `MediaPlayer` オブジェクトの `on` イベントメソッドは、その `MediaPlayer` のメソッドを呼び出せません。また、`MediaPlayer` のメソッドを間接的に呼び出す可能性がある他の `Acrobat` メソッドも呼び出せません。例えば、`on` メソッドでは、文書を閉じる、文書を保存する、アクティブなページを変更する、フォーカスを変更するなど、`MediaPlayer` のメソッドを呼び出すことになる操作は一切行えません。
- `on` イベントメソッドは再入可能ではありません。

`after` イベントメソッドにはこれらの制限はないので、汎用性に富んでいます。`on` イベントメソッドは、同期してイベントを処理する必要がある場合にのみ使用します (`onGetRect` メソッドなど)。

イベントメソッド内の `this` は、`EventListener` オブジェクトを表します。文書にアクセスするには `event.media.doc` を使用し、イベントターゲット (`MediaPlayer` や `ScreenAnnot`) にアクセスするには `event.target` を使用します。

`Events.add` は、ディスパッチのために `EventListener` オブジェクトをインストールします。

`Events.dispatch` は、対応するイベントメソッドにイベントをディスパッチします。`Events.remove` は、ディスパッチテーブルから `EventListener` オブジェクトを削除します。

例

2通りの方法で、`onPlay` および `afterPlay` イベントリスナをインストールします。これらのリスナが実行されると、コンソールに表示されます。

```
// 単純な MediaEvents オブジェクトを作成
var events = new app.media.Events
({
    // Play イベントで直ちに呼び出される
    onPlay: function() { console.println( "onPlay" ); },

    // Play イベント後のアイドル時間に呼び出される
    afterPlay: function() { console.println( "afterPlay" ); },
});
var player = app.media.createPlayer({events: events});
player.events.add({
    afterPlay: function( e ) {
        app.alert("Playback started, doc.URL = " + e.media.doc.URL );
    }
});
player.open();
```

EventListener のメソッド

ここに示すイベントは、マルチメディアに固有のものです。

画面注釈は、これらのイベントに加えて、Acrobat 内で一般的に使用される標準イベント（Destroy、Mouse Up、Mouse Down、Mouse Enter、Mouse Exit、Page Open、Page Close、Page Visible、Page Invisible、Focus、Blur）を受け取ることができます。それらのイベントについて詳しくは、[event](#) を参照してください。

afterBlur

6.0			
-----	--	--	--

Blur イベントは、キーボードフォーカスを得た MediaPlayer や画面注釈がキーボードフォーカスを失ったときに発生します。[onBlur](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

例

次のスクリプトは、レンディションアクションとして実行されます。ユーザが画面注釈をクリックすると、ムービークリップが開きますが、再生はされません。画面注釈の外側をクリックすると（Blur イベント）、ムービーが再生されます。ムービーの再生中に画面注釈をクリックすると（Focus イベント）、ムービーが一時停止します。画面注釈の外側を再度クリックすると、再生が続行されます。

```
var playerEvents = new app.media.Events
({
    afterBlur: function () { player.play(); },
    afterFocus: function () { player.pause(); }
});
var settings = { autoPlay: false };
var args = { settings: settings, events: playerEvents };
var player = app.media.openPlayer(args);
```

[afterFocus](#) も参照してください。

afterClose

6.0			
-----	--	--	--

Close イベントは、メディアプレーヤーが何らかの理由で閉じられるときに発生します。 [onClose](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Close イベントで別のメディアプレーヤーを起動する場合は、最初に `doc.media.canPlay` をテストして、再生が許可されていることを確認してください。文書が閉じられる場合など、再生が許可されないことがあります。

Close イベントの `event` オブジェクトには、標準の `event` プロパティに加えて、次のプロパティが含まれています。

<code>media.closeReason</code>	プレーヤーが閉じられた理由。 <code>app.media.closeReason</code> より。
<code>media.hadFocus</code>	プレーヤーが閉じられたとき、フォーカスを持っていたかどうか。

フォーカスを持っているプレーヤーを閉じると、まず Blur イベントが発生し、その後に Close イベントが発生します。Close イベントでは、`media.hadFocus` によって、プレーヤーが閉じる前にフォーカスを持っていたかどうかが示されます。

`afterClose` イベントメソッドが呼び出されたときには、既に `MediaPlayer` は削除されており、その JavaScript オブジェクトは無効になっています。

パラメータ

<code>oMediaEvent</code>	この <code>EventListener</code> に自動的に渡される <code>event</code> オブジェクト。
--------------------------	--

例

代表的な例については、 [onClose](#) を参照してください。

afterDestroy

6.0			
-----	--	--	--

Destroy イベントは、画面注釈が破棄されたときに発生します。

`afterDestroy` イベントメソッドが呼び出されたときには、既に画面注釈は文書から削除されており、その JavaScript オブジェクトは無効になっています。

[onDestory](#) も参照してください（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

パラメータ

<code>oMediaEvent</code>	この <code>EventListener</code> に自動的に渡される <code>event</code> オブジェクト。
--------------------------	--

afterDone

6.0			
-----	--	--	--

Done イベントは、メディアの再生がメディアの最後に達したときに発生します。

[onDone](#) も参照してください（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

パラメータ

<code>oMediaEvent</code>	この <code>EventListener</code> に自動的に渡される <code>event</code> オブジェクト。
--------------------------	--

afterError

6.0			
-----	--	--	--

Error イベントは、MediaPlayer でエラーが発生したときに発生します。[onError](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Error イベントの event オブジェクトには、標準の event プロパティに加えて、次のプロパティが含まれています。

`media.code` ステータスコード値

`media.serious` 重大なエラーの場合は true、警告の場合は false

`media.text` エラーメッセージのテキスト

パラメータ

`oMediaEvent` この EventListener に自動的に渡される event オブジェクト。

afterEscape

6.0			
-----	--	--	--

Escape イベントは、MediaPlayer が開かれてキーボードフォーカスを持っているときに、ユーザが Esc キーを押すと発生します。MediaPlayer は、Ready イベントを受け取る前に Escape イベントを受け取ることがあります。

[onEscape](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される event オブジェクト。

afterEveryEvent

6.0			
-----	--	--	--

Events オブジェクトに `onEveryEvent` または `afterEveryEvent` プロパティが含まれている場合、その EventListener メソッドは特定の 1 つのイベントだけでなく、各イベントに対して呼び出されます（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

`onEveryEvent` または `afterEveryEvent` プロパティの EventListener 関数は、各イベントのリスナ関数の前に呼び出されます。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される event オブジェクト。

例

onEveryEvent、afterEveryEvent、onPlay、afterPlay をコンソールに表示します。

```
var events = new app.media.Events()
{
    // これは各イベントの開始時に直ちに呼び出される。
    onEveryEvent: function( e )
    { console.println( 'onEveryEvent, event = ' + e.name ); },

    // これは Play イベントで、onEveryEvent が呼び出された後に
    // 呼び出される。
    onPlay: function() { console.println( "onPlay" ); },

    // これはイベントごとに、アイドル時間中に呼び出される。
    afterEveryEvent: function( e )
    { console.println( "afterEveryEvent, event = " + e.name ); },

    // これは Play イベント後のアイドル時間中、
    // afterEveryEvent が呼び出された後に呼び出される。
    afterPlay: function() { console.println( "afterPlay" ); },
});
```

afterFocus

6.0			
-----	--	--	--

Focus イベントは、MediaPlayer または画面注釈がキーボードフォーカスを得たときに発生します。[onFocus](#) も参照してください (イベントメソッドの on と after の違いについては、[383 ページ](#)を参照)。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

例

使用例については、[afterBlur](#) を参照してください。

afterPause

6.0			
-----	--	--	--

Pause イベントは、ユーザの操作や pause メソッドの呼び出しによって、メディアの再生が一時停止したときに発生します。[onPause](#) も参照してください (イベントメソッドの on と after の違いについては、[383 ページ](#)を参照)。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

afterPlay

6.0			
-----	--	--	--

Play イベントは、ユーザの操作や `play` メソッドの呼び出しによって、メディアの再生が開始または再開したときに発生します。[onPlay](#) も参照してください（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この `EventListener` に自動的に渡される `event` オブジェクト。

afterReady

6.0			
-----	--	--	--

Ready イベントは、新規作成された `MediaPlayer` が使用可能な状態になったときに発生します。[onReady](#) も参照してください（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

`MediaPlayer` オブジェクトのほとんどのメソッドは、Ready イベントが発生するまで呼び出せません。

パラメータ

`oMediaEvent` この `EventListener` に自動的に渡される `event` オブジェクト。

後述の [afterScript](#)、`Markers.get`、[MediaOffset](#) オブジェクトを参照してください。

例

この（文書レベルの）スクリプトでは、複数のメディアクリップを再生します。それぞれの画面注釈に対して、メディア（`OpenPlayer`）プレーヤーが 1 つ開かれます。準備ができると、`afterReady` スクリプトから `Multiplayer` にその旨が通知されます。

```
// パラメータは、文書、ページ、レンディション／注釈の名前、multiPlayer のインスタンス
function OnePlayer( doc, page, name, multiPlayer )
{
    var player = app.media.openPlayer({
        annot: doc.media.getAnnot(
            { nPage: page, cAnnotTitle: name }),
        rendition: doc.media.getRendition( name ),
        settings: { autoPlay: false },
        events: {
            afterReady: function( e ) {
                multiPlayer.afterReady( player );
            },
        }
    );
    return player;
}
```

```
// パラメータは、文書、ページ、レンディション／注釈の名前のリスト
function MultiPlayer( doc, page )
{
    var nPlayersCueing = 0; // キューアップされているプレーヤーの数
    var players = []; // SinglePlayer

    this.afterReady = function( player ) {
        if( ! player.didAfterReady ) {
            player.didAfterReady = true;
            nPlayersCueing--;
            if( nPlayersCueing == 0 ) this.play();
        }
    }
    this.play = function() {
        for( var i = 0; i < players.length; i++ ) players[i].play();
    }
    for( var i = 2; i < arguments.length; i++ ) {
        players[i-2] = new OnePlayer(doc,page,arguments[i],this);
        nPlayersCueing++;
    }
}
```

複数のメディアクリップを再生するには、フォームボタンの「マウスボタンを放す」アクションなどで次のコードを実行します。

```
var myMultiPlayer = new MultiPlayer( this, 0, "Clip1", "Clip2" );
```

afterReady の別の例については、[afterScript](#) を参照してください。

afterScript

6.0			
-----	--	--	--

Script イベントは、再生中のメディアでスクリプトトリガが発行されたときに発生します。[onScript](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Script イベントの event オブジェクトには、標準の event プロパティに加えて、次のプロパティが含まれています。

media.command	コマンド名
media.param	コマンドパラメータの文字列

この 2 つの文字列には、メディアクリップで提供される任意の値を含めることができます。これらの文字列は、実行可能な JavaScript コードであるとは限りません。その解釈は、onScript または afterScript の EventListener によって行われます。

パラメータ

oMediaEvent	この EventListener に自動的に渡される event オブジェクト。
-------------	--

例

MediaPlayer.seek で紹介されている例の一部を次に示します。このメディアは、マーカーとスクリプトがサポートされているオーディオクリップ (.wma) であり、指定された名前の引用句が収録されています。afterReady リスナは、各引用句の先頭にあるマーカーの数をカウントしています。引用句の末尾には、コマンドスクリプトが埋め込まれています。afterScript リスナはそれらのコマンドを監視し、pause コマンドである場合はプレーヤーを一時停止します。

```
var nMarkers=0;
var events = new app.media.Events;
events.add({
    // このオーディオクリップ内の引用句の数をカウントし、nMarkers として保存
    afterReady: function() {
        var g = player.markers;
        while ( (index = g.get( { index: nMarkers } ) ) != null )
            nMarkers++;
    },
    // 各引用句の末尾にはスクリプトがある。pause コマンドである場合は
    // プレーヤーを一時停止する。
    afterScript: function( e ) {
        if ( e.media.command == "pause" ) player.pause();
    }
});
var player = app.media.openPlayer({
    rendition: this.media.getRendition( "myQuotes" ),
    settings: { autoPlay: false },
    events: events
});
```

afterSeek

6.0			
-----	--	--	--

Seek イベントは、seek が呼び出されて、MediaPlayer による再生開始位置が確定したときに発生します。[onSeek](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

メディアプレーヤーの中には、Seek イベントが発生しないものもあります。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

afterStatus

6.0			
-----	--	--	--

Status イベントは、MediaPlayer によって報告される様々なステータスが変更されたときに発生します。[onStatus](#) も参照してください (イベントメソッドの on と after の違いについては、[383 ページ](#)を参照)。

Status イベントの event オブジェクトには、標準の event プロパティに加えて、次のプロパティが含まれています。

media.code app.media.status で定義されているステータスコード値。

media.text ステータスマッセージのテキスト。

次の値は、一部のメディアプレーヤーによって、media.code == app.media.status.buffering の場合にのみ使用されます。その他の場合、これらはゼロになります。

media.progress 進捗値 (0 ~ media.total)。

media.total 最大進捗値。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

例

プレイヤーのステータスを監視します。これは、画面注釈に関連付けられる Rendition イベントから実行します。

```
var events = new app.media.Events
events.add({
    afterStatus: function ( e ) {
        console.println( "Status code " + e.media.code +
            ", description: " + e.media.text);
    }
});
app.media.openPlayer({ events: events });
```

afterStop

6.0			
-----	--	--	--

Stop イベントは、ユーザの操作や stop メソッドの呼び出しによって、メディアの再生が停止したときに発生します。[onStop](#) も参照してください (イベントメソッドの on と after の違いについては、[383 ページ](#)を参照)。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

onBlur

6.0			
-----	--	--	--

Blur イベントは、キーボードフォーカスを得た MediaPlayer や画面注釈がキーボードフォーカスを失ったときに発生します。[afterBlur](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

onClose

6.0			
-----	--	--	--

Close イベントは、MediaPlayer が何らかの理由で閉じられるときに発生します。[afterClose](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Close イベントで別のメディアプレーヤーを起動する場合は、最初に doc.media.canPlay をテストして、再生が許可されていることを確認してください。文書が閉じられる場合など、再生が許可されないことがあります。

Close イベントの event オブジェクトには、標準の event プロパティに加えて、次のプロパティが含まれています。

media.closeReason プレーヤーが閉じられた理由。app.media.closeReason より。

media.hadFocus プレーヤーが閉じられたとき、フォーカスを持っていたかどうか。

フォーカスを持っているプレーヤーを閉じると、まず Blur イベントが発生し、その後に Close イベントが発生します。Close イベントでは、media.hadFocus によって、プレーヤーが閉じる前にフォーカスを持っていたかどうかが示されます。

afterClose イベントメソッドが呼び出されたときには、既に MediaPlayer は削除されており、その JavaScript オブジェクトは無効になっています。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

例

メディアクリップが閉じた理由に関する情報を取得します。このスクリプトは、レンディションアクションから実行します。app.media.[closeReason](#) を参照してください。

```
var playerEvents = new app.media.Events({
  onClose: function (e) {
    var eReason, r = app.media.closeReason;
    switch (e.media.closeReason)
    {
      case r.general: eReason = "general"; break;
```

```
        case r.error: eReason = "error"; break;
        case r.done: eReason = "done"; break;
        case r.stop: eReason = "stop"; break;
        case r.play: eReason = "play"; break;
        case r.uiGeneral: eReason = "uiGeneral"; break;
        case r.uiScreen: eReason = "uiScreen"; break;
        case r.uiEdit: eReason = "uiEdit"; break;
        case r.docClose: eReason = "Close"; break;
        case r.docSave: eReason = "docSave"; break;
        case r.docChange: eReason = "docChange"; break;
    }
    console.println("Closing...The reason is " + eReason );
}
});
app.media.openPlayer({ events: playerEvents });
```

onDestroy

6.0			
-----	--	--	--

Destroy イベントは、画面注釈が破棄されたときに発生します。[afterDestroy](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

onDone

6.0			
-----	--	--	--

Done イベントは、メディアの再生がメディアの最後に達したときに発生します。[afterDone](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

onError

6.0			
-----	--	--	--

Error イベントは、MediaPlayer でエラーが発生したときに発生します。[afterError](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Error イベントの `event` オブジェクトには、標準の `event` プロパティに加えて、次のプロパティが含まれています。

<code>media.code</code>	ステータスコード値
<code>media.serious</code>	重大なエラーの場合は <code>true</code> 、警告の場合は <code>false</code>
<code>media.text</code>	エラーメッセージのテキスト

パラメータ

`oMediaEvent` この `EventListener` に自動的に渡される `event` オブジェクト。

onEscape

6.0			
-----	--	--	--

Escape イベントは、MediaPlayer が開かれてキーボードフォーカスを持っているときに、ユーザが Esc キーを押すと発生します。MediaPlayer は、Ready イベントを受け取る前に Escape イベントを受け取ることがあります。

[afterEscape](#) も参照してください（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この `EventListener` に自動的に渡される `event` オブジェクト。

onEveryEvent

6.0			
-----	--	--	--

Events オブジェクトに `onEveryEvent` または `afterEveryEvent` プロパティが含まれている場合、その `EventListener` メソッドは特定の 1 つのイベントだけでなく、各イベントに対して呼び出されます

`onEveryEvent` または `afterEveryEvent` プロパティの `EventListener` メソッドは、各イベントのリスナ関数の前に呼び出されます（イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この `EventListener` に自動的に渡される `event` オブジェクト。

onFocus

6.0			
-----	--	--	--

Focus イベントは、MediaPlayer または画面注釈がキーボードフォーカスを得たときに発生します。[afterFocus](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

onGetRect

6.0			
-----	--	--	--

GetRect イベントは、ドッキングされた MediaPlayer の表示矩形をマルチメディアプラグインが取得する必要があるたびに発生します。

GetRect イベントの `event` オブジェクトには、標準の `event` プロパティに加えて、次のプロパティが含まれています。

`media.rect` プレーヤーの矩形を表す、デバイススペースにおける 4 つの数値の配列。

`onGetRect` メソッドは、`oMediaEvent` のこのプロパティを設定してから戻る必要があります。

注意： `afterGetRect` リスナを作成することができますが、これはあまり役立ちません。このリスナで `rect` プロパティを返しても無視されます。`rect` プロパティの設定は、`onGetRect` リスナで行う必要があります。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

例

ページ 0 には（サムネールサイズの）一連の ScreenAnnot があります。ページ 1 は空白ページです。ビューアを連続見開きモードにして、両方のページが並んで表示されるようにします。標準的なレンディションアクションまたは「マウスボタンを放す」JavaScript アクションを次に示します。

```
var rendition = this.media.getRendition("Clip1");
var settings = rendition.getPlaySettings();
var annot = this.media.getAnnot({ nPage:0,cAnnotTitle:"ScreenClip1" });
var player = app.media.openPlayer({
    rendition: rendition,
    annot: annot,
    settings: { windowType: app.media.windowType.docked },
    events:
    {
        onGetRect: function (e) {
            var width = e.media.rect[2] - e.media.rect[0];
            var height = e.media.rect[3] - e.media.rect[1];
            width *= 3; // 幅と高さを 3 倍にする
    }
});
```

```
        height *= 3;
        e.media.rect[0] = 36; // 左端と上端を
        e.media.rect[1] = 36; // 左上隅へ移動
        e.media.rect[2] = e.media.rect[0]+width;
        e.media.rect[3] = e.media.rect[1]+height;
        return e.media.rect; // これを返す
    }
}
});
player.page = 1; // ページ 1 に表示。これにより onGetRect イベントが発生する。
```

この例のバリエーションについては、MediaPlayer.[page](#) および MediaPlayer.[triggerGetRect](#) を参照してください。

onPause

6.0			
-----	--	--	--

Pause イベントは、ユーザの操作や pause メソッドの呼び出しによって、メディアの再生が一時停止したときに発生します。[afterPause](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent`

この EventListener に自動的に渡される `event` オブジェクト。

onPlay

6.0			
-----	--	--	--

Play イベントは、ユーザの操作や play メソッドの呼び出しによって、メディアの再生が開始または再開したときに発生します。[afterPlay](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

`oMediaEvent`

この EventListener に自動的に渡される `event` オブジェクト。

onReady

6.0			
-----	--	--	--

Ready イベントは、新規作成された MediaPlayer が使用可能な状態になったときに発生します。MediaPlayer オブジェクトのほとんどのメソッドは、Ready イベントが発生するまで呼び出せません。[afterReady](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

onScript

6.0			
-----	--	--	--

Script イベントは、再生中のメディアでスクリプトトリガが発行されたときに発生します。[afterScript](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Script イベントの event オブジェクトには、標準の event プロパティに加えて、次のプロパティが含まれています。

media.command	コマンド名
media.param	コマンドパラメータの文字列

この 2 つの文字列には、メディアクリップで提供される任意の値を含めることができます。これらの文字列は、実行可能な JavaScript コードであるとは限りません。その解釈は、onScript または afterScript の EventListener によって行われます。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

onSeek

6.0			
-----	--	--	--

Seek イベントは、seek が呼び出されて、MediaPlayer による再生開始位置が確定したときに発生します。メディアプレーヤーの中には、Seek イベントが発生しないものもあります。

[afterSeek](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

パラメータ

oMediaEvent この EventListener に自動的に渡される event オブジェクト。

onStatus

6.0			
-----	--	--	--

Status イベントは、MediaPlayer によって報告される様々なステータスが変更されたときに発生します。[afterStatus](#) も参照してください（イベントメソッドの on と after の違いについては、[383 ページ](#)を参照）。

Status イベントの `event` オブジェクトには、標準の `event` プロパティに加えて、次のプロパティが含まれています。

`media.code` `app.media.status` で定義されているステータスコード値

`media.text` ステータスマッセージのテキスト

次の値は、一部のメディアプレーヤーによって、`media.code == app.media.status.buffering` の場合にのみ使用されます。その他の場合、これらはゼロになります。

`media.progress` 進捗値 (0 ~ `media.total`)

`media.total` 最大進捗値

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

onStop

6.0			
-----	--	--	--

Stop イベントは、ユーザの操作や `stop` メソッドの呼び出しによって、メディアの再生が停止したときに発生します。

[afterstop](#) も参照してください (イベントメソッドの `on` と `after` の違いについては、[383 ページ](#)を参照)。

パラメータ

`oMediaEvent` この EventListener に自動的に渡される `event` オブジェクト。

Events

マルチメディアの Events オブジェクトは、EventListener オブジェクトの集合です。MediaPlayer オブジェクトや ScreenAnnot オブジェクトの events プロパティは、Events オブジェクトです。

Events オブジェクトのコンストラクタは `app.media.Events` です。

例

次のスクリプトは、レンディションアクションとして実行します。

```
console.println("Ready to play ¥" + event.action.rendition.uiName
    +"¥ from screen annot ¥" + event.targetName + "¥.");
// 単純な app.media.Events オブジェクトを作成する
var events = new app.media.Events({
    // Event オブジェクトを、すべてのイベントリスナにパラメータとして渡す
    // 下記の例では、e が Event パラメータを表す
    // Play イベントで直ちに呼び出される
    onPlay: function( e ) { console.println( "onPlay: media.id = "
        + e.media.id ); },
    // Play イベント後のアイドル時間に呼び出される
    afterPlay: function() { console.println( "afterPlay" ); },
});
var player = app.media.openPlayer({ events: events });
```

Events のメソッド

add

6.0			
-----	--	--	--

この Events オブジェクトのディスパッチテーブルに、任意の数の EventListener オブジェクトを追加します。以前のリスナはすべて保持されます。イベントが発生すると、対応するリスナメソッドがすべて呼び出されます。

標準のイベントディスパッチャはまず、すべての `onEveryEvent` メソッドを、追加された順に呼び出します。次に、ディスパッチされたイベントに対応するすべての `on` イベントを、追加された順に呼び出します。最後に、すべての `after` イベントを呼び出すための非常に短いタイム（1ミリ秒）を設定します。そのタイムが発生すると、`on` イベントの場合と同じ順で `after` イベントが呼び出されます。

[EventListener](#) オブジェクトについて概説している段落の、`on` イベントと `after` イベントの説明を参照してください。

注意：同じ EventListener を 2 回追加しようとした場合、2 回目の試行は無視されます。

イベントメソッドの内部で `EventListener` を追加した場合、新しいリスナのメソッドは、現在のイベントに対するディスパッチの一部として呼び出されます。

パラメータ

EventListener オブジェクトを表す任意の数のパラメータ。

例

onPlay イベントの EventListener を追加します。プレーヤーは MediaPlayer オブジェクトです。

```
player.events.add
(
  {
    onPlay: function() { console.println( "onPlay" ); }
  });

```

[remove](#) も参照してください。

dispatch

6.0			
-----	--	--	--

MediaPlayer でイベントが発生すると、Multimedia プラグインが event オブジェクトを作成して、
MediaPlayer.events.dispatch(event) を呼び出します。同様に、ScreenAnnot は
ScreenAnnot.events.dispatch(event) を呼び出します。

イベントディスパッチシステムのうち、Acrobat Multimedia プラグインが直接呼び出せるのは、この dispatch メソッドのみです。独自の MediaPlayer.events オブジェクトを用意することで、元の dispatch メソッドを置き換えて、まったく異なる独自のイベントディスパッチシステムを提供することができます。

dispatch メソッドでは、イベント (`oMediaEvent.name` によって識別される) に対応するすべての EventListener を呼び出す必要があります。ほとんどの場合、PDF ファイルで独自の dispatch メソッドを提供する必要はなく、標準のイベントディスパッチシステムを使用できます。

パラメータ

`oMediaEvent` event オブジェクト。

独自の dispatch メソッドを作成するときは、`oMediaEvent.name` にスペースが含まれる場合があることに注意してください。標準の dispatch メソッドでは、`oMediaEvent.media.id` に `oMediaEvent.name` をコピーする際に、名前からスペースを削除して、JavaScript のイベントメソッド名として直接使用できるようにしています。

また、作成した独自の dispatch メソッドは、各イベントの発生時に同期的に呼び出されます。実行するすべての処理には、`on` イベントメソッドと同じ制限 ([EventListener](#) オブジェクトを参照) が適用されます。したがって、このメソッドで MediaPlayer オブジェクトを呼び出したり、MediaPlayer のメソッドが間接的に呼び出されるような操作を実行することはできません。

JavaScript コードで dispatch メソッドを直接呼び出すことは可能ですが、通常は行いません。

例

カスタムイベントディスパッチャを使用して新しいメディアプレーヤーを作成します。これは高度なテクニックであり、通常の PDF JavaScript で使用することはほとんどありません。

```
var player = doc.media.newPlayer(
{
  events:
  {
    dispatch: function( e )
    {

```

```
        console.println( 'events.dispatch' + e.toSource() );
    }
}
});
// Script イベントを、メディアの再生中に発生した場合のように
// 合成してディスパッチする。標準のイベント
// ディスパッチャの場合、このイベントに対して追加されているすべてのイベントリスナーが
// 呼び出される。前述のカスタムディスパッチャの場合、メッセージがコンソールに
// 記録される。
var event = new Event;
event.name = "Script";
event.media = { command: "test", param: "value" };
player.events.dispatch( event );
```

remove

6.0			
-----	--	--	--

このメソッドは、Events.add で以前に追加された 1 つ以上の EventListener を削除します。Events.add でオブジェクトリテラルを直接使用した場合は、そのオブジェクトへの参照を渡す方法がないので、Media.remove でそのリスナーを削除することはできません。EventListener を削除できるようにするには、オブジェクトリテラルではなく変数を add メソッドに渡します。そうすることで、その変数を remove に渡すことができます。次の例を参照してください。

remove メソッドをイベントメソッドの内部で呼び出して、任意の EventListener（現在のイベントメソッドが属しているリスナーも含む）を削除できます。現在のイベントメソッドは引き続き実行されますが、同じ EventListener オブジェクトの他のイベントメソッドは呼び出されません。

パラメータ

任意の数の EventListener オブジェクト。

例

player が MediaPlayer オブジェクトであるとします。

```
var listener = { afterStop: function() { app.alert("Stopped!"); } }
player.events.add( listener );           // リスナを追加する
.....
player.events.remove( listener );        // 後で削除する
```

FDF

6.0			
-----	--	--	--

このオブジェクトは、PDF エンコードのデータ交換ファイルに相当します。通常、FDF ファイルは、PDF ファイルから書き出したフォームデータを格納するために使用しますが、汎用のデータファイルとして使用することもできます。FDF オブジェクトでは、汎用データファイルとしての使用がサポートされています。

メソッドやプロパティのうち、クイックバーに というマークが表示されているものについては、バッチイベント、コンソールイベント、アプリケーション初期化イベントでのみ使用できます。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

FDF のプロパティ

deleteOption

6.0			
-----	--	--	--

FDF ファイルを処理した後、FDF ファイルを自動的に削除するかどうかを示します。FDF ファイルの内容や処理方法によって、この値が使用される場合とされない場合があります。Acrobat 6.0 以降では、このプロパティは埋め込みファイルに対して使用されます。有効な値は、次のとおりです。

0 — (デフォルト) Acrobat は、処理後に FDF ファイルを自動的に削除します。

1 — Acrobat は、処理後に FDF ファイルを削除しません (ただし、Web ブラウザやメールによって削除される場合があります)。

2 — Acrobat は、FDF ファイルを削除するかどうかを尋ねるプロンプトを処理後に表示します (ただし、Web ブラウザやメールによって削除される場合があります)。

型

整数

アクセス

R / W

isSigned

6.0			
-----	--	--	--

FDF データファイルが署名されている場合は `true` を返します。

型

ブーリアン

アクセス

R

例

fdf が署名されているかどうかを確認します。

```
var fdf = app.openFDF("/C/temp/myDoc.fdf");
console.println( "It is " + fdf.isSigned + " that this FDF is signed");
fdf.close();
```

詳しい例については、[signatureSign](#) を参照してください。

numEmbeddedFiles

6.0			X
-----	--	--	---

FDF ファイルに埋め込まれているファイルの数。FDF オブジェクトが有効な FDF ファイルを表している場合、例外は発生しません。

FDF ファイルにファイルを埋め込むには、[addEmbeddedFile](#) メソッドを使用します。

型

整数

アクセス

R

例

新しい FDF オブジェクトを作成し、PDF 文書を埋め込んでから FDF を保存し、FDF を再度開いて埋め込みファイルの数をカウントします。

```
var fdf = app.newFDF();
fdf.addEmbeddedFile("/C/myPDFs/myDoc.pdf");
fdf.save("/c/temp/myDocWrapper.fdf");
fdf = app.openFDF("/c/temp/myDocWrapper.fdf");
console.println("The number of embedded files = "
+ fdf.numEmbeddedFiles);
fdf.close();
```

FDF のメソッド

addContact

6.0		S	X
-----	--	---	---

FDF ファイルに連絡先を追加します。

パラメータ

`oUserEntity` FDF ファイルに追加する連絡先をリストした `UserEntity` オブジェクト。

戻り値

失敗した場合は例外が発生します。

例

```
var oEntity={firstName:"Fred", lastName:"Smith", fullName:"Fred Smith"};
var f = app.newFDF();
f.addContact( oEntity );
f.save( "/c/temp/FredCert.fdf" );
```

addEmbeddedFile

6.0			
-----	--	--	--

指定したファイルを、FDF ファイルに埋め込まれているファイルの配列の末尾に追加します。`nSaveOption` の設定によっては、この FDF ファイルを開くと、埋め込みファイルを保存するように求めるプロンプトが表示されます。埋め込みファイルが PDF ファイルの場合は、保存したファイルが開かれて表示されます。埋め込みファイルが FDF ファイルの場合は、保存したファイルが開かれて処理されます。

埋め込みファイルが含まれている FDF ファイルは、Acrobat 4.05 以降でサポートされています。PDF を埋め込んだファイルは、例えば、HTTP サーバに置いて活用することができます。これによって、PDF ファイルをブラウザ内に表示せずに、直接ダウンロードして保存できるようになります。これらの埋め込みファイルと、FDF ファイルに格納されているフォームデータの作成元のファイルの間には、何の関係もありません。

パラメータ

<code>cDIPath</code>	(オプション) デバイスに依存しない、ユーザのハードドライブにあるファイルの絶対パス。指定しなかった場合は、ファイルを選択するように求めるプロンプトが表示されます。
<code>nSaveOption</code>	(オプション) FDF ファイルが開かれたときの埋め込みファイルの表示方法、保存場所、保存後にファイルを削除するかどうかを示します。有効な値は、次のとおりです。 0 — ファイルは Acrobat 文書フォルダに保存されます。Acrobat 8 では、ファイルをディスクに保存するかどうかを尋ねる確認ダイアログボックスが表示されます。 1 — (デフォルト) 埋め込みファイルの保存先のファイル名を尋ねるプロンプトが表示されます。 2 — 使用しないでください。 3 — ファイルは一時ファイルとして自動的に保存され、クリーンアップ時 (Acrobat を終了するとき) に削除されます。 Acrobat 4.05 ~ 5.05 では、ファイルの保存先フォルダを設定せずに値 0 または 3 を使用すると、保存先フォルダを尋ねるプロンプトが表示されます。 ファイルが PDF または FDF の場合は、 <code>nSaveOption</code> がどの値であっても、保存したファイルが自動的に Acrobat で開かれます。

戻り値

この処理を完了することができなかった場合は、例外が発生します。完了した場合は、FDF ファイルに現在埋め込まれているファイルの数を返します。

例

新しい FDF を作成し、PDF 文書を埋め込み、保存します。

```
var fdf = app.newFDF();
fdf.addEmbeddedFile("/C/myPDFs/myDoc.pdf");
fdf.save("/c/temp/myDocs.fdf");
```

addRequest

6.0			
-----	--	--	--

FDF ファイルにリクエストを追加します。1 つの FDF ファイルに追加できるリクエストは 1 つのみです。FDF ファイルにリクエストが既に含まれている場合は、新しいリクエストに置き換えられます。

パラメータ

cType	リクエストのタイプ。現時点で有効な値は、文字列「CMS」のみです。これは、連絡先情報のリクエストを示します。
cReturnAddress	リクエストの返信先アドレスを示す文字列。これは、mailto:、http:、https: のいずれかで始まり、"http://www.example.com/cgi.pl" または "mailto:jdoe@example.com" の形式であることが必要です。
cName	(オプション) リクエストを生成した個人または組織の名前。

戻り値

エラーがある場合は、例外が発生します。

例

```
var f = app.newFDF();
f.addRequest( "CMS", "http://www.example.com/cgi.pl", "Acme Corp" );
f.save( "/c/tmp/request.fdf" );
```

close

6.0			
-----	--	--	--

FDF ファイルを直ちに閉じます。

戻り値

エラーがある場合は、例外が発生します。

FDF オブジェクトの [save](#) メソッドも参照してください。FDF ファイルは、このメソッドを使用して閉じることができます。

例

[addEmbeddedFile](#) メソッドの例を参照してください。

mail

6.0			
-----	--	--	--

FDF オブジェクトを一時 FDF ファイルとして保存し、このファイルをすべての受信者に添付ファイルとして送信します。送信前にユーザ操作を要求するかどうかを選択できます。一時ファイルは、不要になると削除されます。

app.[mailGetAddrs](#)、app.[mailMsg](#)、Doc の [mailDoc](#) メソッドと [mailForm](#) メソッド、Report オブジェクトの [mail](#) メソッドも参照してください。

注意：Windows でこのメソッドを使用するには、クライアントコンピュータのデフォルトのメールプログラムで MAPI が有効になっている必要があります。

パラメータ

bUI (オプション) ユーザインターフェイスを表示するかどうかを指定します。true (デフォルト) の場合は、メーラーの新規メッセージウィンドウがユーザに表示され、他のパラメータ値が初期値として使用されます。false の場合、cTo パラメータが必須で、その他のパラメータはすべてオプションです。

注意：(Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、bUI パラメータは無視され、デフォルトの true の動作になります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

cTo (オプション) セミコロンで区切られた、メッセージの受信者のリスト。

cCc (オプション) セミコロンで区切られた、メッセージの CC 受信者のリスト。

cBcc (オプション) セミコロンで区切られた、メッセージの BCC 受信者のリスト。

cSubject (オプション) メッセージの件名。長さの制限は 64 KB です。

cMsg (オプション) メッセージの内容。長さの制限は 64 KB です。

戻り値

エラーがある場合は、例外が発生します。

例

```
var fdf = app.openFDF( "/c/temp/myDoc.fdf" );
/* メーラーの新規メッセージウィンドウが表示される */
fdf.mail();

/* FDF ファイルが添付されたメールを fun1@example.com と fun2@example.com に送信 */
fdf.mail( false, "fun1@example.com", "fun2@example.com", "",
"This is the subject", "This is the body." );
```

save

6.0			
-----	--	--	--

FDF オブジェクトをファイルとして保存します。保存は必ず行われます。保存するとファイルが閉じ、該当する FDF オブジェクトへの参照は無効になります。

[close](#) メソッドも参照してください。FDF ファイルは、このメソッドを使用して閉じることもできます。

パラメータ

cDIPath	デバイスに依存しない、保存するファイルのパス。 注意： cDIPath は、セーフパスであり (31 ページの「セーフパス」 を参照)、拡張子が .fdf であることが必要です。
---------	--

戻り値

エラーがある場合は、例外が発生します。

例

新しい FDF を作成し、PDF 文書を埋め込み、保存します。

```
var fdf = app.newFDF();
fdf.addEmbeddedFile( "/C/myPDFs/myDoc.pdf" );
fdf.save( "/c/temp/myDocs.fdf" );
```

signatureClear

6.0			
-----	--	--	--

FDF オブジェクトに署名がある場合は、その署名をクリアします。成功した場合は true を返します。FDF オブジェクトに署名がない場合は、何も行いません。ファイルの保存は行いません。

戻り値

成功した場合は true。

signatureSign

6.0			
-----	--	--	--

指定の security オブジェクトを使用して、FDF オブジェクトに署名します。FDF オブジェクトに署名できるのは一度のみです。FDF オブジェクトはメモリ内で署名されます。ファイルとして自動的にディスクに保存されることはありません。署名後の FDF オブジェクトを保存するには、[save](#) を呼び出します。FDF の署名をクリアするには、[signatureClear](#) を呼び出します。

パラメータ

oSig	署名に使用する SecurityHandler オブジェクト。通常、security オブジェクトは、署名に使用する前に初期化する必要があります。FDF ファイルに署名できるかどうかについては、セキュリティハンドラのマニュアルを参照してください。SecurityHandler オブジェクトの signFDF プロパティは、その security オブジェクトが FDF ファイルの署名に使用できるかどうかを示します。
oInfo	(オプション) 署名の書き込み可能なプロパティを持つ signatureInfo オブジェクト。
nUI	(オプション) 署名時に表示するダイアログボックスのタイプ。有効な値は、次のとおりです。 0 — ダイアログボックスを表示しません。 1 — 編集可能なフィールドがない簡略化されたダイアログボックスを表示します（フィールドは oInfo で指定できます）。 2 — 理由、場所、連絡先情報を指定できる編集可能なフィールドを含む詳細なダイアログボックスを表示します。 デフォルトは 0 です。
cUISignTitle	(オプション) 署名ダイアログボックスで使用するタイトル。このパラメータは、nUI がゼロでない場合にのみ使用されます。
cUISelectMsg	(オプション) 署名に必要なリソース（証明書など）をユーザが選択する必要がある場合に表示するメッセージ。このパラメータは、nUI がゼロでない場合にのみ使用されます。

戻り値

署名が正常に行われた場合は true。それ以外の場合は false。

例

既存の FDF データファイルを開いて署名します。

```
var eng = security.getHandler( "Adobe.PPKLite" );
eng.login("myPassword" ,"/c/test/Acme.pfx");
var myFDF = app.openFDF( "/c/temp/myData.fdf" );
if( !myFDF.isSigned ) {
    myFDF.signatureSign({
        oSig: eng,
        nUI: 1,
        cUISignTitle: "Sign Embedded File FDF",
        cUISelectMsg: "Please select a Digital ID to use to "
            + "sign your embedded file FDF."
    });
    myFDF.save( "/c/temp/myData.fdf" );
};
```

signatureValidate

6.0			X
-----	--	--	---

FDF オブジェクトの署名を検証し、署名のプロパティを表す `SignatureInfo` オブジェクトを返します。

パラメータ

<code>oSig</code>	(オプション) 署名の検証に使用するセキュリティハンドラ。 <code>SecurityHandler</code> オブジェクトか、次のプロパティを持つ汎用オブジェクトを指定できます。
	<code>oSecHdlr</code> — 署名の検証に使用する <code>SecurityHandler</code> オブジェクト。
	<code>bAltSecHdlr</code> — ブーリアン値。 <code>true</code> の場合は、ユーザ環境設定に基づいて選択された別のセキュリティハンドラで署名を検証できます。デフォルトは <code>false</code> です。この場合は、署名の <code>handlerName</code> プロパティで返されるセキュリティハンドラが、署名の検証に使用されます。 <code>oSecHdlr</code> が指定されている場合、このパラメータは使用されません。
	<code>oSig</code> を指定しない場合は、署名の <code>handlerName</code> プロパティで返されるセキュリティハンドラが、署名の検証に使用されます。
<code>bUI</code>	(オプション) <code>true</code> の場合は、データファイルを検証するときに、必要に応じて UI が表示されます。検証ハンドラが指定されていない場合は、UI を使用して選択できます。

戻り値

`SignatureInfo` オブジェクト。署名のステータスは、`status` プロパティで示されています。

例

FDF ファイルを開き、署名がある場合は検証し、コンソールに情報を返します。

```
fdf = app.openFDF("/c/temp/myDoc.fdf");
eng = security.getHandler("Adobe.PPKLite");
if (fdf.isSigned)
{
    var oSigInfo = fdf.signatureValidate({
        oSig: eng,
        bUI: true
    });
    console.println("Signature Status: " + oSigInfo.status);
    console.println("Description: " + oSigInfo.statusText);
} else {
    console.println("FDF not signed");
}
```

Field

このオブジェクトは、Acrobat フォームフィールド（Acrobat のフォームツールや Doc の `addField` メソッドで作成したフィールド）を表します。フォームの作成者が UI を使用して既存のフィールドプロパティ（境界線の色やフォントなど）を変更できるのと同様に、JavaScript ユーザは Field オブジェクトを使用して同様の変更を行うことができます。

フィールドにアクセスするためには、Doc のメソッドを使用して、フィールドを JavaScript 変数にバインドする必要があります。フィールドのオブジェクトプロパティを変更したり、メソッドにアクセスしたりすると、そのフィールドに複数の変数がバインドされることがあります。そのフィールドを変更すると、バインドされているすべての変数に影響を与えます。

```
var f = this.getField("Total");
```

このようにすれば、`f` という変数を使用して、「Total」というフォームフィールドを操作できるようになります。

文書内のフィールドは、階層構造をなしていることがあります。「FirstName」や「LastName」のような名前を持つフォームフィールドは、フラットネームと呼ばれます。フラットネームの間には関連付けはありません。文書内でフィールドの階層を作成するには、フィールド名を変更します。

例えば、「Name.First」および「Name.Last」という名前にすることで、フィールドのツリーを形成することができます。ピリオド（「.」）は、Acrobat フォームで階層の区切りを表すためのセパレータです。これらのフィールドの「Name」が階層の親で、「First」と「Last」が子になります。「Name」というフィールドは、対応する実体が存在しない内部フィールドです。「First」と「Last」は、ページ上に表示されるターミナルフィールドです。

同じ名前を持つ Acrobat フォームフィールドは、その値も同じになります。同じ名前を持つターミナルフィールドは、データの表現方法（配置されているページ、回転角度、フォント、背景色など）がそれ異なる場合がありますが、値はすべて同じになります。したがって、1つのターミナルフィールドで値を変更すると、同じ名前を持つ他のすべてのフィールドも自動的に更新されます。

ターミナルフィールドのそれぞれの実体のことをウィジェットと呼びます。各ウィジェットの識別には、名前ではなく、そのターミナルフィールドにおけるインデックスが使用されます（インデックスは 0 から数えます）。このインデックスは、そのフィールドの各ウィジェットが作成された順番を表しています（タブの順序とは関係ありません）。

特定のウィジェットのインデックスを調べるには、Acrobat の「フィールド」ナビゲーションタブを使用します。フィールド名の「#」記号の後に示されている数字がインデックスです（Acrobat 6.0 以降では、フィールドに複数のウィジェットがある場合にのみ、ウィジェットのインデックスが示されます）。「フィールド」パネルでエントリをダブルクリックすると、文書内の対応するウィジェットに移動できます。逆に、文書内のフィールドを選択すると、「フィールド」パネルの対応するエントリが強調表示されます。

Acrobat 6.0 以降の `getField` では、フィールドの特定のウィジェットに対応する Field オブジェクトを取得できます。パラメータには、フィールド名とウィジェットのインデックスをピリオド（「.」）で区切った文字列を指定します。この方法を使用した場合、`getField` によって返される Field オブジェクトは、1つのウィジェットのみを表しています。このようにして返された Field オブジェクトも、通常のフィールド名を渡して返される Field オブジェクトと同じように使用できます。ただし、次の表に示すように、影響を受けるノードのセットは異なります。

アクション	すべてのウィジェットを表す Field オブジェクト	特定の 1 つのウィジェットを表す Field オブジェクト
ウィジェットプロパティの取得	ウィジェット #0 のプロパティを取得します。	そのウィジェットのプロパティを取得します。
ウィジェットプロパティの設定	そのフィールドの子であるすべてのウィジェットのプロパティを設定します。 (<code>rect</code> プロパティと <code>setFocus</code> メソッドは例外で、ウィジェット #0 のみが設定されます。後述の例を参照してください。)	そのウィジェットのプロパティを設定します。

アクション	すべてのウィジェットを表す Field オブジェクト	特定の 1 つのウィジェットを表す Field オブジェクト
フィールドプロパティの取得	そのフィールドのプロパティを取得します。親フィールドのプロパティを取得します。	
フィールドプロパティの設定	そのフィールドのプロパティを設定します。親フィールドのプロパティを設定します。	

次の例では、「my radio」というフィールドの 2 番目のラジオボタン（インデックスは 0 から数えます）の rect プロパティを変更します。

```
var f = this.getField("my radio.1");
f.rect = [360, 677, 392, 646];
```

フィールドの属性とウィジェットの属性

Field オブジェクトのプロパティには、value などのように、そのフィールドに属するすべてのウィジェットに適用されるものもあれば、rect などのように、個別のウィジェットに適用されるものもあります。

次のフィールドプロパティやメソッドは、フィールドレベルの属性に影響を与えます。

```
calcOrderIndex、charLimit、comb、currentValueIndices、defaultValue、
doNotScroll、doNotSpellCheck、delay、doc、editable、exportValues、
fileSelect、multiline、multipleSelection、name、numItems、page、password、
readonly、required、submitName、type、userName、value、valueAsString、
clearItems、browseForFileToSubmit、deleteItemAt、getItemAt、insertItemAt、
setAction、setItems、signatureInfo、signatureSign、signatureValidate
```

次のフィールドプロパティやメソッドは、ウィジェットレベルの属性に影響を与えます。

```
alignment、borderStyle、buttonAlignX、buttonAlignY、buttonPosition、
buttonScaleHow、buttonScaleWhen、display、fillColor、hidden、highlight、
lineWidth、print、rect、strokeColor、style、textColor、textFont、textSize、
buttonGetCaption、buttonGetIcon、buttonImportIcon、buttonSetCaption、
buttonSetIcon、checkThisBox、defaultIsChecked、isBoxChecked、
isDefaultChecked、setAction、setFocus
```

注意：setAction メソッドが、フィールドに適用されるかウィジェットに適用されるかは、イベントによつて異なります。Keystroke、Validate、Calculate、Format イベントの場合は、フィールドに適用されます。MouseUp、MouseDown、MouseEnter、MouseExit、OnFocus、OnBlur イベントの場合は、ウィジェットに適用されます。

checkThisBox、defaultIsChecked、isBoxChecked、isDefaultChecked の各メソッドは、パラメータとしてウィジェットのインデックス (nWidget) を取ります。特定のウィジェットを表す Field オブジェクト (f) でこれらのメソッドを呼び出した場合、nWidget パラメータはオプションになり（渡しても無視され）、f で表される特定のウィジェットのみが処理されます。

Field のプロパティ

一般に、フィールドのプロパティは、PDF 文書のフィールドディクショナリや注釈ディクショナリに保存されているプロパティに相当します（『PDF Reference』バージョン 1.7 を参照）。

PDF 文書に保存されるプロパティ値には、名前として保存されるものと、文字列として保存されるものがあります（『PDF Reference』バージョン 1.7 を参照）。名前として保存されるプロパティは、文字列の長さが 127 文字までに制限されています。

127 文字の制限があるプロパティとしては、チェックボックスやラジオボタンの `value` や `defaultValue` などがあります。『PDF Reference』バージョン 1.7 には、注釈のプロパティとその保存形式がすべて記載されています。

alignment

3.01			
------	--	--	--

テキストフィールドでのテキストの整列方法を示します。有効な値は、次のとおりです。

`left`
`center`
`right`

型

文字列

アクセス

R / W

フィールド

`text`

例

```
var f = this.getField("MyText");
f.alignment = "center";
```

borderStyle

3.01			
------	--	--	--

フィールドの境界線のスタイル。有効な境界線スタイルは、次のとおりです。

`solid`
`dashed`
`beveled`
`inset`
`underline`

境界線のスタイルによって、矩形の境界線の描画方法が決まります。border オブジェクトは、フィールドの境界線スタイルがすべて定義されている静的定数です。次の表にこれを示します。

型	キーワード	説明
solid	border.s	矩形の周囲を実線で描きます。
beveled	border.b	solid スタイルの境界線の内側に、ベベル（浮き出た外観）が加えられます。
dashed	border.d	周囲を破線で描きます。
inset	border.i	solid スタイルの境界線の内側に、切り込み（くぼんだ外観）が加えられます。
underline	border.u	矩形の底辺（下線）を描きます。

型

文字列

アクセス

R / W

フィールド

すべて

例

次の例では、フィールドの境界線スタイルを solid に設定します。

```
var f = this.getField("MyField");
f.borderStyle = border.s; /* border.s は「solid」 */
```

buttonAlignX

5.0	①		F
-----	---	--	---

ボタン内でアイコンの左に配分するスペースの割合を制御します。有効な値は 0 ~ 100 (パーセント単位) です。デフォルト値は 50 です。

アイコンの倍率設定で、元の縦横比を保たないようにしている場合は、(スペースの配分を考慮する必要がないので) このプロパティは使用されません。

型

整数

アクセス

R / W

フィールド

button

buttonAlignY

5.0			
-----	--	--	--

ボタン内でアイコンの下に配分するスペースの割合を制御します。有効な値は 0 ~ 100 (パーセント単位) です。デフォルト値は 50 です。

アイコンの倍率設定で、元の縦横比を保たないようにしている場合は、(スペースの配分を考慮する必要がないので) このプロパティは使用されません。

型

整数

アクセス

R / W

フィールド

button

例

この例は、エレベータのアニメーションです。「myElevator」は、縦長のボタンフォームフィールドです。アイコンは、ボタンの外観として既に取り込まれています。

```
function MoveIt()
{
    if ( f.buttonAlignY == 0 ) {
        f.buttonAlignY++;
        run.dir = true;
        return;
    }
    if ( f.buttonAlignY == 100 ) {
        f.buttonAlignY--;
        run.dir = false;
        return;
    }
    if (run.dir) f.buttonAlignY++;
    else f.buttonAlignY--;
}
var f = this.getField("myElevator");
f.buttonAlignY=0;
run = app.setInterval("MoveIt()", 100);
```

```
run.dir=true;
toprun = app.setTimeOut(
    "app.clearInterval(run); app.clearTimeOut(toprun)", 2*20000+100);
```

buttonFitBounds

6.0			
-----	--	--	--

true の場合、ボタンフィールドの境界線に合わせてアイコンのサイズが設定されます。このプロパティを true にしても、他のアイコン配置プロパティを使用して、アイコンの倍率やボタン内での位置を調整できます。

以前のバージョンの Acrobat では、ボタンに合わせてアイコンのサイズを変更した場合、フィールドの境界色が指定されていなくても、境界幅が常に考慮されていました。ボタンの境界色が設定されている状態でこのプロパティを true に設定すると、例外が発生します。

型

ブーリアン

アクセス

R / W

フィールド

button

buttonPosition

5.0			
-----	--	--	--

ボタン内でのテキストとアイコンの配置方法を制御します。position オブジェクトには、有効なオプションがすべて定義されています。

アイコン／テキストの配置	キーワード
テキストのみ	position.textOnly
アイコンのみ	position.iconOnly
アイコンを上、テキストを下	position.iconTextV
テキストを上、アイコンを下	position.textIconV
アイコンを左、テキストを右	position.iconTextH
テキストを左、アイコンを右	position.textIconH
テキストをアイコンに重ねる	position.overlay

型

整数

アクセス

R / W

フィールド

button

buttonScaleHow

5.0			
-----	--	--	--

ボタンに合わせて（必要に応じて）アイコンのサイズを変更する方法を制御します。scaleHow オブジェクトには、有効なオプションがすべて定義されています。

アイコンの拡大／縮小方法	キーワード
--------------	-------

元の縦横比を保つ	scaleHow.proportional
----------	-----------------------

ボタンに合わせる	scaleHow.anamorphic
----------	---------------------

型

整数

アクセス

R / W

フィールド

button

buttonScaleWhen

5.0			
-----	--	--	--

ボタンに合わせて（必要に応じて）アイコンのサイズを変更する条件を制御します。scaleWhen オブジェクトには、有効なオプションがすべて定義されています。

アイコンの拡大／縮小条件	キーワード
--------------	-------

常に調整	scaleWhen.always
------	------------------

調整しない	scaleWhen.never
-------	-----------------

アイコンの拡大／縮小条件	キーワード
アイコンが大きすぎる場合	scaleWhen.tooBig
アイコンが小さすぎる場合	scaleWhen.tooSmall

型

整数

アクセス

R / W

フィールド

button

calcOrderIndex

3.01	D		F
------	---	--	---

文書に含まれているフィールドの計算順序を変更します。計算可能な text フィールドや combo box フィールドを文書に追加すると、計算順序の配列にフィールド名が追加され、フィールドの計算順序が決まります。calcOrderIndex プロパティは、フォームツールのフィールドの計算順序の設定に似た機能を提供します。

型

整数

アクセス

R / W

フィールド

combobox、text

例

```
var a = this.getField("newItem");
var b = this.getField("oldItem");
a.calcOrderIndex = b.calcOrderIndex + 1;
```

この例では、「oldItem」フィールドの後に「newItem」フィールドが追加されたものとしています。このスクリプトでは、「newItem」フィールドの calcOrderIndex プロパティを変更して、「oldItem」フィールドの前に「newItem」が計算されるようにしています。

charLimit

3.01			
------	--	--	--

テキストフィールドに入力できる文字数を制限します。

文字数の制限に達したことを検出する方法については、`event.fieldFull` を参照してください。

型

整数

アクセス

R / W

フィールド

`text`

例

フィールドに入力できる文字数の制限を設定します。

```
var f = this.getField("myText");
f.charLimit = 20;
```

comb

6.0			
-----	--	--	--

`true` に設定すると、フィールドの背景にマス目が描画され（フィールド値の 1 文字につき 1 マス）、フィールド内の文字がマス目の中に表示されます。描画されるマス目の数は、`charLimit` プロパティで指定します。

これは、テキストフィールドにのみ適用されます。フィールドプロパティの `multiline`、`password`、`fileSelect` のいずれかが設定されているときにこのプロパティを設定すると、例外が発生します。このプロパティを設定すると、`doNotScroll` プロパティも設定されます。

型

ブーリアン

アクセス

R / W

フィールド

`text`

例

新規文書の左上隅に、マス目フィールドを作成します。

```
var myDoc = app.newDoc(); // 空の文書を作成
var Bbox = myDoc.getPageBox("Crop"); // トリミング領域を取得
var inch = 72;

// 文書の一番上にテキストフィールドを追加
var f = myDoc.addField("Name.Last", "text", 0,
    [ inch, Bbox[1]-inch, 3*inch, Bbox[1]-inch - 14 ] );
// このテキストフィールドにプロパティを追加
f.strokeColor = color.black;
f.textColor = color.blue;
f.fillColor = ["RGB",1,0.66,0.75];

f.comb = true; // マス目フィールドとして宣言
f.charLimit = 10; // 最大文字数
```

commitOnSelChange

6.0			
-----	--	--	--

選択が変更された直後にフィールド値を確定するかどうかを制御します。

- `true` の場合、選択した直後にフィールド値が確定されます。
- `false` の場合は、フィールド値を確定せずに何度も選択を変更できるようになります。値が確定されるのは、ユーザがフィールドの外側をクリックするなどして、フィールドがフォーカスを失った場合のみです。

型

ブーリアン

アクセス

R / W

フィールド

combobox、listbox

currentValueIndices

5.0			
-----	--	--	--

リストボックスやコンボボックスで選択されている 1 つ以上の値を、取得または設定します。

取得

リストボックスやコンボボックスフィールドの値（文字列）が格納されているオプション配列におけるインデックスを返します。インデックスは 0 から数えます。選択されているフィールド値が 1 つの場合は、整数を返します。複数の場合は、昇順でソートされた整数の配列を返します。フィールドの現在値がリストに含まれていない場合は（編集可能なコンボボックスではその可能性があります）、-1 を返します。

設定

リストボックスやコンボボックスの現在値を設定します。引数には、1 つの整数か、整数の配列を指定できます。1 つの文字列を現在値として設定するには、オプション配列におけるその文字列のインデックス（0 から数えます）を表す整数を渡します。編集可能なコンボボックスで、設定したい値がリストに含まれていない場合は、[value](#) を設定する必要があります。それ以外の場合は、[currentValueIndices](#) を使用してリストボックスやコンボボックスの値を設定します。

複数選択が可能なリストボックスで複数選択を行うには、オプション配列におけるそれらの文字列のインデックスの配列（昇順でソートしたもの）を、引数としてこのプロパティに渡します。これは、JavaScript でリストボックスの項目を複数選択するための唯一の方法です。リストボックスで複数選択を可能にするには、[multipleSelection](#) を設定します。

関連するメソッドやプロパティとしては、[numItems](#)、[getItemAt](#)、[insertItemAt](#)、[deleteItemAt](#)、[setItems](#) があります。

型

整数または配列

アクセス

R / W

フィールド

combobox、listbox

例（取得）

ボタンの「マウスボタンを放す」アクションです。これにより、リストボックスの現在値が取得されます。

```
var f = this.getField("myList");
var a = f.currentValueIndices;
if (typeof a == "number") // 1 つの値が選択されている
    console.println("Selection: " + f.getItemAt(a, false));
else { // 複数の値が選択されている
    console.println("Selection:");
    for (var i = 0; i < a.length; i++)
        console.println("  " + f.getItemAt(a[i], false));
}
```

例（設定）

次のコードでは、リストボックスの 2 番目と 4 番目（0 から数えたインデックスでは 1 と 3）の項目を選択します。

```
var f = this.getField("myList");
f.currentValueIndices = [1,3];
```

defaultStyle

6.0	D		F
-----	---	--	---

このプロパティは、フォームフィールドのデフォルトのスタイル属性を定義します。ユーザが、ツールバーでプロパティを変更せずに、空のフィールドをクリックしてテキストを入力した場合に、デフォルトのプロパティが使用されます。このプロパティは、text プロパティを持たない単一の span オブジェクトです。デフォルトスタイルを表すこの span オブジェクトのプロパティの一部は、Field オブジェクトのプロパティと連動しています。Field オブジェクトでそれらのプロパティを変更すると、フィールドの defaultStyle プロパティも変更されます。逆に、フィールドの defaultStyle プロパティを変更すると、Field オブジェクトの対応するプロパティも変更されます。

次の表に、Field オブジェクトとデフォルトスタイルのプロパティを示し、その相違について説明します。

Field のプロパティ	defaultStyle (Span のプロパティ)	説明
alignment	alignment	alignment プロパティの値は、デフォルトスタイルと Field オブジェクトで同じです。
textFont	fontFamily fontWeight fontStyle	Field では、フォントのファミリー、太さ、スタイルをすべて表す完全なフォント名を、単一のプロパティで指定します。デフォルトスタイルでは、各属性を別個のプロパティで指定します。デフォルトスタイルのフォントプロパティと完全に一致するフォントが見つからない場合は、類似のフォントが使用されるか、合成されます。
textColor	textColor	textColor プロパティの値は、デフォルトスタイルと Field オブジェクトで同じです。
textSize	textSize	textSize プロパティの値は、デフォルトスタイルと Field オブジェクトで同じです。

注意： defaultStyle のスタイルは、空のフィールドにテキストを入力した場合に使用されます。フィールドに既にテキストが入力されている場合は、defaultStyle を変更しても、テキストのスタイルは変更されません。テキストを追加入力した場合は、挿入先のテキストの属性（またはツールバーで指定されている属性）が使用されます。

フィールドにリッチテキストを貼り付けた場合、そのリッチテキストで指定されていない属性は、defaultStyle に基づいて設定されます。

Superscript と Subscript は defaultStyle では無視されます。

型

Span オブジェクト

アクセス

R / W

フィールド

rich text

例

テキストフィールドのデフォルトスタイルを変更します。

```
var style = this.getField("Text1").defaultStyle;  
style.textColor = color.red;  
style.textSize = 18;  
  
// Courier Std がない場合は monospace フォントを使用  
style.fontFamily = ["Courier Std", "monospace"];  
  
this.getField("Text1").defaultStyle = style;
```

defaultValue

3.01	D		F
------	----------	--	----------

フィールドのデフォルト値。フォームがリセットされると、この値がフィールドに設定されます。コンボ ボックスやリストボックスのデフォルト値は、書き出し値と項目名のどちらでも指定できます。競合が生じた場合（例えば、書き出し値で一致する項目と、項目名で一致する項目が、リスト内に同時に存在する場合）は、書き出し値で一致する項目が優先されます。

型

文字列

アクセス

R / W

フィールド

button、signature 以外のすべて

例

```
var f = this.getField("Name");  
f.defaultValue = "Enter your name here.;"
```

doNotScroll

5.0	D		F
-----	----------	--	----------

true の場合、テキストフィールドはスクロールせず、フィールドを囲む矩形領域に表示が限定されます。このプロパティを true または false に設定することは、フィールドの「オプション」タブにある「長いテキストをスクロール」フィールドにチェックを付けるまたは解除することと同じです。

型

ブーリアン

アクセス

R / W

フィールド

text

doNotSpellCheck

5.0	D		F
-----	----------	--	----------

`true` の場合は、この編集可能なテキストフィールドでスペルチェックが実行されません。このプロパティを `true` または `false` に設定することは、フィールドのプロパティダイアログボックスの「オプション」タブにある「スペルチェック」属性にチェックを付けるまたは解除することと同じです。

型

ブーリアン

アクセス

R / W

フィールド

combobox (編集可能なもの)、text

delay

3.01			F
------	--	--	----------

フィールドの外観が再描画されるのを一時的に停止します。複数のフィールドプロパティに対する一連の変更が終了してから、フィールドの外観を再描画したい場合に使用するのが一般的です。このプロパティを `true` に設定すると、`delay` が `false` に設定されるまでそのフィールドが待ち状態になります。`false` に設定するとフィールドが再描画され、最新の設定が反映されます。

複数のフィールドを同時に再描画したい場合は、Doc の [delay](#) フラグを使用します。

型

ブーリアン

アクセス

R / W

フィールド

すべて

例

チェックボックスの外観を変更します。delay を true に設定し、変更を加えてから delay を false に設定します。

```
// myCheckBox フィールドを取得
var f = this.getField("myCheckBox");
// delay を設定し、フィールドのプロパティを変更して
// エッジをペベルにし、線の太さを標準にする
f.delay = true;
f.borderColor = border.b;
...
f.strokeWidth = 2;
f.delay = false; // 変更を反映
```

display

4.0	D		F
-----	----------	--	----------

画面や印刷での、フィールドの表示／非表示を制御します。有効な値は、次のとおりです。

説明	キーワード
画面や印刷でフィールドを表示する	display.visible
画面や印刷でフィールドを非表示にする	display.hidden
画面ではフィールドを表示するが、印刷では非表示にする	display.noPrint
画面ではフィールドを非表示にするが、印刷では表示する	display.noView

hidden や print の代わりにこのプロパティを使用してください。

型

整数

アクセス

R / W

フィールド

すべて

例

```
// display プロパティを設定
var f = getField("myField");
f.display = display.noPrint;

// フィールドが画面と印刷で非表示であるかをテスト
if (f.display == display.hidden) console.println("hidden");
```

doc

3.01			
------	--	--	--

フィールドが属する文書の Doc を返します。

型

オブジェクト

アクセス

R

フィールド

すべて

editable

3.01	D		F
------	---	--	---

コンボボックスを編集可能にするかどうかを制御します。true の場合、ユーザはフィールドに入力できます。false の場合、ユーザは所定の項目から選択する必要があります。

型

ブーリアン

アクセス

R / W

フィールド

combobox

例

```
var f = this.getField("myComboBox");
f.editable = true;
```

exportValues

5.0	D		F
-----	---	--	---

フィールドの書き出し値を表す文字列の配列。この配列には、フィールドに含まれている注釈と同じ数の要素が含まれます。要素と注釈は、注釈が作成された順にマッピングされます（タブの順序とは関係ありません）。

ラジオボタンフィールドの場合、フィールドがグループとして機能するためには、このプロパティを適切に設定する必要があります。チェックされているボタンの値が、フィールド全体の値として使用されます。

チェックボックスフィールドは、書き出し値を設定しなくても、そのまま使用できます。その場合、フィールドにチェックを付けたときのデフォルト値は「はい」になります。フィールドのチェックを解除したときのデフォルト値は「Off」になります（これは、どのボタンもチェックされていないラジオボタンフィールドと同じです）。

型

配列

アクセス

R / W

フィールド

checkbox, radiobutton

例

ラジオボタンフィールドを作成し、書き出し値を設定します。

```
var d = 40;
var f = this.addField("myRadio", "radiobutton", 0, [200, 510, 210, 500]);
this.addField("myRadio", "radiobutton", 0, [200+d, 510-d, 210+d, 500-d]);
this.addField("myRadio", "radiobutton", 0, [200, 510-2*d, 210, 500-2*d]);
this.addField("myRadio", "radiobutton", 0, [200-d, 510-d, 210-d, 500-d]);
f.strokeColor = color.black;
// 各ラジオフィールドに書き出し値を設定
f.exportValues = ["North", "East", "South", "West"];
```

fileSelect

5.0	D	S	F
-----	---	---	---

true の場合は、テキストフィールドの「オプション」タブにあるファイル選択フラグ（「ファイルの選択に使用する」）を設定するのと同じになります。この場合、フィールド値は、フォームとともに内容を送信するファイルなどのパスを表します。

ユーザは、フィールドにパスを直接入力することも、ファイルの参照ボタンを使用して入力することもできます ([browseForFileToSubmit](#) メソッドを参照)。

注意：ファイル選択フラグは、multiline、charLimit、password、defaultValue の各プロパティと同時に設定することはできません。Macintosh プラットフォームでは、ファイル選択フラグを設定するとフィールドが読み取り専用になります。したがって、ファイルの参照ボタンを使用してフィールドに入力する必要があります ([browseForFileToSubmit](#) を参照)。

このプロパティを設定できるのは、バッヂイベントまたはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

型

ブーリアン

アクセス

R / W

フィールド

text

fillColor

4.0	D		F
-----	----------	--	----------

フィールドの背景色を指定します。フィールドの背景色とは、フィールドの矩形を塗りつぶす色のことです。値の定義には、transparent、gray、RGB または CMYK カラーを使用します。カラー配列の定義や、このプロパティでの値の使用方法について詳しくは、[191 ページの「カラー配列」](#) を参照してください。

旧バージョンでは、このプロパティは bgColor という名前でした。bgColor の使用は現在推奨されていませんが、下位互換性のために残されています。

型

配列

アクセス

R / W

フィールド

すべて

例

テキストフィールドの背景色を変更します。現在の色が赤である場合は青色に、それ以外の場合は黄色に変更します。

```
var f = this.getField("myField");
if (color.equal(f.fillColor, color.red))
    f.fillColor = color.blue;
else
    f.fillColor = color.yellow;
```

hidden

X	D		F
---	---	--	---

注意：このプロパティの使用は推奨されません。代わりに `display` プロパティを使用してください。

値が `false` であればフィールドが表示され、`true` であれば非表示になります。デフォルト値は `false` です。

型

ブーリアン

アクセス

R / W

フィールド

すべて

highlight

3.01	D		F
------	---	--	---

ユーザがボタンをクリックしたときのボタンの状態を示します。サポートされている 4 つの強調表示モードは、次のとおりです。

`none` — ボタンをクリックしても視覚的な変化は起きません。

`invert` — ボタンを囲む矩形領域が一時的に反転します。

`push` — ボタン上のアイコンまたはテキスト（存在する場合）が一時的に下に押されて表示されます。

`outline` — ボタンを囲む境界線が一時的に反転します。

`highlight` オブジェクトには、これらの状態が定義されています。

タイプ	キーワード
none	highlight.n
invert	highlight.i
push	highlight.p
outline	highlight.o

型

文字列

アクセス

R / W

フィールド

button

例

ボタンの強調表示プロパティを「invert」に設定します。

```
var f = this.getField("myButton");
f.highlight = highlight.i;
```

lineWidth

4.0	D		F
-----	---	--	---

フィールドを囲む矩形の境界線の太さを示します。線の色が透明の場合、このプロパティによる影響はありません（境界線のスタイルが beveled の場合を除く）。有効な値は、次のとおりです。

- 0 — なし
- 1 — 細
- 2 — 標準
- 3 — 太

旧バージョンでは、このプロパティは `borderWidth` という名前でした。`borderWidth` の使用は現在推奨されていませんが、下位互換性のために残されています。

`lineWidth` のデフォルト値は 1（細）です。任意の整数を指定できますが、値が 5 を超えると、フィールドの外観が歪む可能性があります。

型

整数

アクセス

R / W

フィールド

すべて

例

テキストボックスの境界線の太さを標準に変更します。

```
f.lineWidth = 2
```

multiline

3.01	D		F
------	---	--	---

フィールドでのテキストの折り返し方法を制御します。`false`（デフォルト）の場合は、単一行のテキストフィールドになります。`true` の場合は、複数行が有効となり、フィールドの右端で行が折り返されます。

型

ブーリアン

アクセス

R / W

フィールド

text

例

Doc の [getField](#) メソッドの[例 1](#) を参照してください。

multipleSelection

5.0	D		F
-----	---	--	---

`true` の場合、リストボックスで複数項目を選択できます。

[type](#)、[value](#)、[currentValueIndices](#) も参照してください。

型

ブーリアン

アクセス

R / W

フィールド

listbox

name

3.01			
------	--	--	--

このプロパティは、文字列オブジェクトとしてフィールドの完全修飾名を返します。

Acrobat 6.0 以降では、Field オブジェクトがウィジェットである場合は、返される名前の末尾に「.」とウィジェットのインデックスが付加されます。

型

文字列

アクセス

R

フィールド

すべて

例

Field オブジェクトを取得して、フィールドの name をコンソールに書き込みます。

```
var f = this.getField("myField");
// コンソールウィンドウに「myField」が表示される
console.println(f.name);
```

numItems

3.01			
------	--	--	--

コンボボックスまたはリストボックスに含まれている項目数。

型

整数

アクセス

R

フィールド

combobox、listbox

例

リストボックスに含まれている項目の数を取得します。

```
var f = this.getField("myList");
console.println("There are " + f.numItems + " in this list box");
```

コンボボックスやリストボックスの表示項目名や値にアクセスするには、[getItemAt](#) メソッドを使用します。numItems の別の例については、このメソッドを参照してください。

page

5.0			
-----	--	--	--

フィールドのページ番号またはその配列。フィールドのウィジェットが文書内に 1 つのみ存在する場合は、そのウィジェットが存在するページ番号（0 から数えます）を整数で返します。フィールドのウィジェットが複数存在する場合は、各ウィジェットが存在するページ番号（0 から数えます）を列挙した整数の配列を返します。この配列では、ウィジェットが作成された順にページ番号が列挙されます（タブの順序とは関係ありません）。非表示のテンプレートページにあるウィジェットに対しては、-1 を返します。

型

整数または配列

アクセス

R

フィールド

すべて

例 1

特定のフィールドが、単一のページで表示されているか、複数のページで表示されているかを調べます。

```
var f = this.getField("myField");
if (typeof f.page == "number")
    console.println("This field only occurs once on page " + f.page);
else
    console.println("This field occurs " + f.page.length + " times");
```

例 2 (Acrobat 6.0)

page プロパティを使用して、フィールドのウィジェットの数を取得することができます。この例では、1 つのラジオボタンフィールドに含まれるラジオボタンの数を取得します。

```
var f = this.getField("myRadio");
if ( typeof f.page == "object" )
    console.println("There are " + f.page.length
        + " radios in this field.");
```

password

3.01	D		F
------	----------	--	----------

フィールドに入力されたデータを、アスタリスクで表示するかどうかを指定します（データを送信するときには、実際に入力したデータが送信されます）。このプロパティが `true` の場合、文書をディスクに保存しても、このフィールドのデータは保存されません。

型

ブーリアン

アクセス

R / W

フィールド

`text`

print

X	D		F
----------	----------	--	----------

注意：このプロパティの使用は推奨されません。代わりに `display` プロパティを使用してください。

`true` の場合、文書を印刷したときにフィールドが表示されます。`false` の場合、印刷したときにフィールドが非表示になります。このプロパティは、印刷した時には表示する必要がない制御ボタンやフィールドを隠す場合などに使用できます。

型

ブーリアン

アクセス

R / W

フィールド

すべて

radiosInUnison

6.0	D		F
-----	----------	--	----------

`false` の場合は、同じ名前と書き出し値のボタンがラジオボタングループに存在しても、それらを同時に選択することはできません（これは HTML のラジオボタンと同じです）。新しいラジオボタンのデフォルトは `false` です。

`true` の場合は、同じ名前と書き出し値のボタンがラジオボタングループに存在すると、それらが同時にオンまたはオフになります（これは Acrobat 4.0 のラジオボタンと同じです）。

型

ブーリアン

アクセス

R / W

フィールド

radiobutton

readonly

3.01	D		F
------	----------	--	----------

フィールドの読み取り専用属性。フィールドを読み取り専用にすると、フィールドは表示されますが、変更はできなくなります。

型

ブーリアン

アクセス

R / W

フィールド

すべて

例

フィールドを取得して、読み取り専用にします。

```
var f = this.getField("myTextField");
f.value = "You can't change this message!";
f.readonly = true;
```

rect

5.0	D		F
-----	----------	--	----------

フォームフィールドのサイズと位置を指定する、回転ユーザースペースにおける 4 つの数値の配列。これらの 4 つの数値は、境界を表す矩形の 左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。

注意：Annotation オブジェクトにも rect プロパティがありますが、この座標は回転ユーザースペースではなく、Field オブジェクトの rect プロパティとは座標の順番が異なります。

型

配列

アクセス

R / W

フィールド

すべて

例 1

幅 2 インチのテキストフィールドを「myText」フィールドの右にレイアウトします。

```
var f = this.getField("myText"); // Field オブジェクトを取得
var myRect = f.rect; // その矩形を取得
myRect[0] = f.rect[2]; // 右下隅の x 座標を、新しく左上隅の x 座標にする
myRect[2] += 2 * 72; // 右下隅の x 座標を 2 インチ移動する
f = this.addField("myNextText", "text", this.pageNum, myRect);
f.strokeColor = color.black;
```

例 2

既存のボタンフィールドを 10 ポイント右に移動します。

```
var b = this.getField("myButton");
var aRect = b.rect; // b.rect のコピーを作成
aRect[0] += 10; // 最初の x 座標を 10 増加
aRect[2] += 10; // 次の x 座標を 10 增加
b.rect = aRect; // b.rect の値を更新
```

required

3.01			
------	--	--	--

フィールドに値が必要かどうかを指定します。true の場合、ユーザが送信ボタンをクリックしてフィールド値を送信するには、フィールドが null 以外の値である必要があります。フィールド値が null の場合は、ユーザに警告メッセージが表示され、送信は行われません。

型

ブーリアン

アクセス

R / W

フィールド

button 以外のすべて

例

「myField」を必須フィールドにします。

```
var f = this.getField("myField");
f.required = true;
```

richText

6.0	D		F
-----	----------	--	----------

true の場合は、そのフィールドでリッチテキストフォーマットが許可されます。デフォルトは false です。

型

ブーリアン

アクセス

R / W

フィールド

text

関連するオブジェクトやプロパティとしては、[richValue](#)、[defaultStyle](#)、[event.richValue](#)、[event.richChange](#)、[event.richChangeEx](#)、Annotation オブジェクトの[richContents](#) プロパティ、[Span](#) オブジェクトがあります。

例 1

Field オブジェクトを取得して、リッチテキストフォーマットに設定します。

```
var f = this.getField("Text1");
f richText = true;
```

詳細な例については、[richValue](#) の[例 2](#) を参照してください。

例 2

文書に含まれているリッチテキストフィールドの数をカウントします。

```
var count = 0;
for ( var i = 0; i < this.numFields; i++)
{
    var fname = this.getNthFieldName(i);
    var f = this.getField(fname);
    if ( f.type == "text" && f.richText ) count++
}
console.println("There are a total of "+ count + " rich text fields.");
```

richValue

6.0	D		F
-----	---	--	---

このプロパティは、リッチテキストフィールドのテキストコンテンツと書式を指定します。リッチテキスト以外のフィールドでは、このプロパティは `undefined` です。リッチテキストコンテンツは、フィールドのテキストコンテンツと書式を表す `Span` オブジェクトの配列として表されます。

型

`Span` オブジェクトの配列

アクセス

R / W

フィールド

rich text

関連するオブジェクトやプロパティとしては、[richText](#)、[defaultStyle](#)、`event.richValue`、`event.richChange`、`event.richChangeEx`、`Annotation` オブジェクトの [richContents](#) プロパティ、[Span](#) オブジェクトがあります。

例 1

すべての太字のテキストを下線付きの赤いテキストに変換します。

```
var f = this.getField("Text1");
var spans = f.richValue;
for ( var i = 0; i < spans.length; i++ )
{
    if( spans[i].fontWeight >= 700 )
    {
        spans[i].textColor = color.red;
        spans[i].underline = true;
    }
}
f.richValue = spans;
```

例 2

テキストフィールドを作成し、リッチテキストフォーマットを指定して、リッチテキストを挿入します。

```
var myDoc = app.newDoc();                                // 空の文書を作成
var Bbox = myDoc.getPageBox("Crop");                    // トリミング領域を取得
var inch = 72;

// 文書の一番上にテキストフィールドを追加
var f = myDoc.addField("Text1", "text", 0,
[72, Bbox[1]-inch, Bbox[2]-inch, Bbox[1]-2*inch] );
// このテキストフィールドにプロパティを追加
f.strokeColor = color.black;
```

```
f.richText = true; // リッチテキスト
f.multiline = true; // 複数行

// Span オブジェクトの配列を作成
var spans = new Array();
spans[0] = new Object();
spans[0].text = "Attention: ¥r";
spans[0].textColor = color.blue;
spans[0].textSize = 18;

spans[1] = new Object();
spans[1].text = "Adobe Acrobat 6.0 ¥r";
spans[1].textColor = color.red;
spans[1].textSize = 20;
spans[1].alignment = "center";

spans[2] = new Object();
spans[2].text = "will soon be here!";
spans[2].textColor = color.green;
spans[2].fontStyle = "italic";
spans[2].underline = true;
spans[2].alignment = "right";

// リッチフィールドにリッチテキストを指定
f.richValue = spans;
```

rotation

6.0		
-----	--	--

ウィジェットの反時計回りの回転角度。有効な値は、0、90、180、270 です。

型

整数

アクセス

R / W

フィールド

すべて

例

回転したテキストフィールドを各ページに作成し、それらのフィールドにテキストを入力します。

```
for ( var i=0; i < this.numPages; i++ ) {
    var f = this.addField("myRotatedText"+i,"text",i,[6, 6+72, 18, 6]);
    f.rotation = 90; f.value = "Confidential";
    f.textColor = color.red; f.readonly = true;
}
```

strokeColor

4.0			
-----	--	--	--

フィールドを囲む矩形を線幅で指定された太さの線で描くときの色を指定します。値の定義には、`transparent`、`gray`、`RGB` または `CMYK` カラーを使用します。カラー配列の定義や、このプロパティでの値の使用方法について詳しくは、[191 ページの「カラー配列」](#) を参照してください。

旧バージョンでは、このプロパティは `borderColor` という名前でした。`borderColor` の使用は現在推奨されていませんが、下位互換性のために残されています。

型

配列

アクセス

R / W

フィールド

すべて

例

文書に含まれている各テキストフィールドの線の色を赤に変更します。

```
for ( var i=0; i < this.numFields; i++ ) {
    var fname = this.getNthFieldName(i);
    var f = this.getField(fname);
    if ( f.type == "text" ) f.strokeColor = color.red;
}
```

style

3.01			
------	--	--	--

チェックボックスやラジオボタンのグリフスタイルを設定できます。グリフスタイルとは、項目が選択されていることを示すために使用されるグラフィックのことです。

スタイル値は、次のようにキーワードと関連付けられています。

スタイル	キーワード
チェックマーク	<code>style.ch</code>
十字形	<code>style.cr</code>
ひし形	<code>style.di</code>

スタイル	キーワード
円形	style.ci
星型	style.st
四角形	style.sq

型

文字列

アクセス

R / W

フィールド

checkbox、radiobutton

例

この例では、グリフスタイルを円形に設定します。

```
var f = this.getField("myCheckbox");
f.style = style.ci;
```

submitName

5.0	D		F
-----	----------	--	----------

空でない場合は、フォームの送信で [name](#) の代わりに使用されます。HTML 形式（URL エンコード）で送信する場合にのみ有効です。

型

文字列

アクセス

R / W

フィールド

すべて

textColor

4.0			
-----	--	--	--

フィールドの描画色。テキストフィールド、ボタンフィールド、リストボックスフィールドでは、テキストの色を表します。チェックボックスフィールドとラジオボタンフィールドでは、チェックの色を表します。値の定義方法は `fillColor` と同じです。カラー配列の定義と、このプロパティの値の設定方法や使用方法について詳しくは、[191 ページの「カラー配列」](#) を参照してください。

旧バージョンでは、このプロパティは `fgColor` という名前でした。`fgColor` の使用は現在推奨されていませんが、下位互換性のために残されています。

注意：透明カラースペースを使用して `textColor` を設定すると、例外が発生します。

型

配列

アクセス

R / W

フィールド

すべて

例

この例では、描画色を赤に設定します。

```
var f = this.getField("myField");
f.textColor = color.red;
```

textFont

3.01			
------	--	--	--

テキストフィールド、コンボボックス、リストボックス、ボタンに配置するテキストに使用するフォント。有効なフォントは、`font` オブジェクトのプロパティとして定義されています。Acrobat 5.0 以降では、任意のフォントも使用できます。[442 ページの「任意のフォントの使用」](#) を参照してください。

型

文字列

アクセス

R / W

フィールド

button、combobox、listbox、text

font オブジェクト

テキストフォント	キーワード
Times-Roman	font.Times
Times-Bold	font.TimesB
Times-Italic	font.TimesI
Times-BoldItalic	font.TimesBI
Helvetica	font.Helv
Helvetica-Bold	font.HelvB
Helvetica-Oblique	font.HelvI
Helvetica-BoldOblique	font.HelvBI
Courier	font.Cour
Courier-Bold	font.CourB
Courier-Oblique	font.CourI
Courier-BoldOblique	font.CourBI
Symbol	font.Symbol
ZapfDingbats	font.ZapfD

任意のフォントの使用

Acrobat 5.0 以降では、テキストフィールド、コンボボックス、リストボックス、ボタンで任意のフォントを使用できます。それには、PDSysFontGetName で返される PDSysFont のフォント名を、textFont に指定します（『Acrobat and PDF Library API Reference』を参照）。

▶ フォントの PDSysFont フォント名を調べるには：

1. PDF 文書内にテキストフィールドを作成します。このフィールドのテキストに使用するフォントを、UI を使用して設定します。

2. JavaScript デバッガコンソールを開いて、次のスクリプトを実行します。

```
this.getField("Text1").textFont
```

このコードでは、フィールド名が Text1 であると仮定しています。

3. テキストのフォントをプログラムで設定するために必要なフォント名が、文字列としてコンソールに返されます。

例

フォントを Helvetica に設定します。

```
var f = this.getField("myField");
f.textFont = font.Helv;
```

例 (Acrobat 5.0)

myField のフォントを Viva-Regular に設定します。

```
var f = this.getField("myField");
f.textFont = "Viva-Regular";
```

textSize

3.01	D		F
------	----------	--	----------

コントロールで使用するテキストサイズ（ポイント単位）を指定します。チェックボックスフィールドとラジオボタンフィールドでは、テキストサイズによってチェックのサイズが決まります。有効なテキストサイズは 0 ~ 32767 の範囲です。0 という値は、すべてのテキストデータがフィールドの矩形に収まる最大のポイントサイズを表します。

型

数値

アクセス

R / W

フィールド

すべて

例

「myField」のテキストサイズを 28 ポイントに設定します。

```
this.getField("myField").textSize = 28;
```

type

3.01			
------	--	--	--

フィールドのタイプを文字列として返します。有効な値は、次のとおりです。

```
button
checkbox
combobox
listbox
radiobutton
signature
text
```

型

文字列

アクセス

R

フィールド

すべて

例

文書に含まれているテキストフィールドの数をカウントします。

```
var count = 0;
for ( var i=0; i<this.numFields; i++ ) {
    var fname = this.getNthFieldName(i);
    if ( this.getField(fname).type == "text" ) count++;
}
console.println("There are " + count + " text fields.");
```

userName

3.01	D		F
------	----------	--	----------

ユーザに示すフィールド名（短い説明の文字列）。このテキストは、フィールドの上にカーソルを置いたときにツールヒントとして表示されます。エラーメッセージを表示するときは、フィールド名ではなくこの名前を示したほうが、ユーザにとってわかりやすくなります。

型

文字列

アクセス

R / W

フィールド

すべて

例

ボタンフィールドにツールヒントを追加します。

```
var f = this.getField("mySubmit");
f.userName = "Press this button to submit your data.;"
```

value

3.01	D		F
------	----------	--	----------

ユーザが入力したフィールドデータの値。フィールドの `type` に応じて、文字列、日付、数値のいずれかになります。`value` は、計算フィールドの作成に使用するのが一般的です。

Acrobat 6.0 以降では、フィールドにリッチテキストフォーマットが含まれている場合、このプロパティを変更すると書式が失われます。フィールド値と外観は、`defaultStyle` とプレーンテキスト値を使用して再描画されます。書式を維持してフィールド値を変更するには、`richValue` プロパティを使用します。

注意：署名済みの署名フィールドでは、`null` でない文字列が値として返されます。

Acrobat 5.0 以降では、フィールドが複数選択可能なリストボックス ([multipleSelection](#) を参照) である場合は、配列を渡してフィールドの `value` を設定することができます。また、`value` を取得すると、リストボックスで現在選択されている複数の値が配列として返されます。

複数選択可能なリストボックスで値の取得や設定を行う場合は、`currentValueIndices` を使用するのが最も効果的でよい方法です。

[valueAsString](#) および `event.type` も参照してください。

型

各種

アクセス

R / W

フィールド

`button` 以外のすべて

例

この例では、「Oil」フィールドと「Filter」フィールドの値を合計し、それに州の消費税を加えた値を計算フィールドの `value` に設定しています。

```
var oil = this.getField("Oil");
var filter = this.getField("Filter");
event.value = (oil.value + filter.value) * 1.0825;
```

valueAsString

5.0	D		
-----	----------	--	--

フィールドの値を JavaScript 文字列として返します。

フィールドコンテンツを有効な形式に変換しようとする `value` とは動作が異なります。例えば、値が「020」のフィールドの場合、`value` は整数 20 を返し、`valueAsString` は文字列「020」を返します。

型

文字列

アクセス

R

フィールド

button 以外のすべて

Field のメソッド

browseForFileToSubmit

5.0			
-----	--	--	--

`fileSelect` フラグが設定された（チェックされた）テキストフィールドで実行すると、標準のファイル選択ダイアログボックスが表示されます。ダイアログボックスから入力されたパスは、テキストフィールドの値として自動的に割り当てられます。

`fileSelect` フラグがクリアされた（チェックされていない）テキストフィールドで実行すると、例外が発生します。

例

次のコードでは、ファイル選択フラグが設定されたテキストフィールドを参照しています。これは、ボタンフィールドの「マウスボタンを放す」アクションです。

```
var f = this.getField("resumeField");
f.browseForFileToSubmit();
```

buttonGetCaption

5.0			
-----	--	--	--

ボタンに関連付けられているキャプションを取得します。

キャプションを設定するには、`buttonSetCaption` を使用します。

パラメータ

<code>nFace</code>	(オプション) これを指定した場合は、指定のタイプのキャプションが取得されます。
0	—（デフォルト）通常の状態のキャプション
1	— ボタンを押下した状態のキャプション
2	— ボタン上にマウスポインタを置いた状態（ロールオーバー）のキャプション

戻り値

ボタンに関連付けられているキャプション文字列。

例

この例では、アイコンとテキストが表示されているボタンで、キャプションの左右に矢印を表示します。

```
// Mouse Enter イベント
event.target.buttonSetCaption("=> " + event.target.buttonGetCaption()
+ " <=");

// Mouse Exit イベント
var str = event.target.buttonGetCaption();
str = str.replace(/=> | <=/g, "");
event.target.buttonSetCaption(str);
```

ロールオーバーに同じアイコンと、テキストに矢印を挿入したキャプションを設定しても、同じ効果が得られます。ただし、この方法は処理速度が遅く、アイコンがちらつく原因になります。上記のコードではキャプションのみを変更してアイコンは変更しないので、速やかに表示を切り替えられます。

buttonGetIcon

5.0			
-----	--	--	--

ボタンに関連付けられている指定したタイプの Icon オブジェクトを取得します。

ボタンにアイコンを割り当てる方法については、[buttonSetIcon](#) メソッドを参照してください。

パラメータ

nFace	(オプション) これを指定した場合は、指定のタイプのアイコンが取得されます。 0 — (デフォルト) 通常の状態のアイコン 1 — ボタンを押下した状態のアイコン 2 — ボタン上にマウスポインタを置いた状態（ロールオーバー）のアイコン
-------	---

戻り値

Icon オブジェクト。

例

```
// 2 つのボタンアイコンを交換。
var f = this.getField("Button1");
var g = this.getField("Button2");
var temp = f.buttonGetIcon();
f.buttonSetIcon(g.buttonGetIcon());
g.buttonSetIcon(temp);
```

[buttonSetIcon](#) および [buttonImportIcon](#) も参照してください。

buttonImportIcon

3.01			
------	--	--	--

ボタンの外観を別の PDF ファイルから取り込みます。どちらのオプションパラメータも渡さなかった場合は、ファイルを選択するように求めるプロンプトが表示されます。

[buttonGetIcon](#)、[buttonSetIcon](#)、[addIcon](#)、[getIcon](#)、[importIcon](#)、[removeIcon](#) も参照してください。

注意：(Acrobat 8.0) `cPath` を指定した場合、このメソッドを実行できるのは、バッヂイベントとコンソールイベントのみになります。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

<code>cPath</code>	(オプション、Acrobat 5.0) デバイスに依存しないファイルパス。 Acrobat 6.0 以降では、最初に <code>cPath</code> を PDF ファイルとして開こうと試みます。失敗した場合は、ファイルを既知のグラフィック形式 (BMP、GIF、JPEG、PCX、PNG、TIFF) から PDF に変換して、変換されたファイルをボタンアイコンとして取り込もうと試みます。
<code>nPage</code>	(オプション、Acrobat 5.0) アイコンに変換するファイル内のページの番号 (0 から数えます)。デフォルトは 0 です。

戻り値

次の整数のいずれか。

- 1 — ユーザがダイアログボックスをキャンセル
- 0 — エラーなし
- 1 — 選択したファイルを開けない
- 2 — 選択したページが無効

例 (Acrobat 5.0)

社員情報データベースに接続中であると仮定します。データベースとのやり取りには、ADBC オブジェクトや関連するオブジェクトを使用しています。社員のレコードを取得し、`FirstName`、`SecondName`、`Picture` という 3 つの列を使用します。データベースの `Picture` 列には、PDF 形式で保存された社員の写真を示す、デバイスに依存しないパス名が保存されています。この場合、次のようなスクリプトが考えられます。

```
var f = this.getField("myPicture");
f.buttonSetCaption(row.FirstName.value + " " + row.LastName.value);
if (f.buttonImportIcon(row.Picture.value) != 0)
    f.buttonImportIcon("/F/employee/pdfs/NoPicture.pdf");
```

ボタンフィールドである「`myPicture`」では、アイコンとキャプションの両方が表示されるよう設定されています。コードで社員の姓と名を連結し、写真のキャプションに使用しています。アイコンの取得に失敗した場合は、代わりのアイコンを取り込んでいます。

buttonSetCaption

5.0			
-----	---	--	--

ボタンに関連付けるキャプションを設定します。

現在のキャプションを取得するには、`buttonGetCaption` を使用します。

ボタンにアイコンやキャプションを配置する方法について詳しくは、[buttonAlignX](#)、[buttonAlignY](#)、[buttonFitBounds](#)、[buttonPosition](#)、[buttonScaleHow](#)、[buttonScaleWhen](#) を参照してください。

パラメータ

cCaption	ボタンに関連付けるキャプション。
nFace	(オプション) これを指定した場合は、指定のタイプのキャプションが設定されます。 0 — (デフォルト) 通常の状態のキャプション 1 — ボタンを押下した状態のキャプション 2 — ボタン上にマウスポインタを置いた状態 (ロールオーバー) のキャプション

例

```
var f = this.getField("myButton");
f.buttonSetCaption("Hello");
```

buttonSetIcon

5.0			
-----	---	--	--

ボタンに関連付けるアイコンを設定します。

ボタンにアイコンやキャプションを配置する方法について詳しくは、[buttonAlignX](#)、[buttonAlignY](#)、[buttonFitBounds](#)、[buttonPosition](#)、[buttonScaleHow](#)、[buttonScaleWhen](#) を参照してください。

このメソッドの`oIcon` パラメータとして使用する `Icon` オブジェクトを取得するには、`buttonGetIcon` や `doc.getIcon` を使用します。

注意：このメソッドは、文書に埋め込まれたスクリプトから実行する必要があります。例えば、`buttonSetIcon` を含むスクリプトを Adobe Reader のコンソールで実行すると、`NotAllowedError` 例外が発生します。

パラメータ

oIcon	ボタンに関連付ける Icon オブジェクト。
nFace	(オプション) これを指定した場合は、指定のタイプのアイコンが設定されます。 0 — (デフォルト) 通常の状態のアイコン 1 — ボタンを押下した状態のアイコン 2 — ボタン上にマウスポインタを置いた状態 (ロールオーバー) のアイコン

例

この例では、文書に含まれている名前付きアイコンから、アイコン名のリストボックスを作成します。リストボックスで項目を選択すると、その名前のアイコンが「myPictures」フィールドのボタンに表示されます。次のコードを、「myButton」ボタンフィールドの「マウスボタンを放す」アクションに割り当てます。

```
var f = this.getField("myButton")
var aRect = f.rect;
aRect[0] = f.rect[2]; // リストボックスを「myButton」からの
aRect[2] = f.rect[2] + 144; // 相対位置で配置
var myIcons = new Array();
var l = addField("myIconList", "combobox", 0, aRect);
l.textSize = 14;
l.strokeColor = color.black;
for (var i = 0; i < this.icons.length; i++)
    myIcons[i] = this.icons[i].name;
l.setItems(myIcons);
l.setAction("Keystroke",
    'if (!event.willCommit) {'
    + 'var f = this.getField("myPictures");'
    + 'var i = this.getIcon(event.change);'
    + 'f.buttonSetIcon(i);'
    + '}');
```

addIcon の例に示されている対話的な方法やバッヂシーケンスによって、名前付きアイコンを文書に取り込むこともできます。

詳しい例については、[buttonGetCaption](#) を参照してください。

checkThisBox



指定のウィジエットにチェックを付けるかチェックを解除します。

チェックを解除できるのはチェックボックスのみです。ラジオボタンのチェックの解除は、このメソッドでは行えません。デフォルトの状態が `unchecked` である場合は ([defaultIsChecked](#) を参照)、Doc の `resetForm` メソッドを使用して、解除された状態にリセットできます。

注意：ラジオボタングループで書き出し値が重複していない場合は、チェックを付けたいウィジエットの書き出し値を `value` に渡すことで、チェックを付けることができます（チェックをすべて解除するには、空の文字列を渡します）。

パラメータ

nWidget	フィールドのチェックボックスウェイジエットまたはラジオボタンウェイジエットのインデックス (0 から数えます)。このインデックスは、そのフィールドの各ウェイジエットが作成された順番を表しています (タブの順序とは関係ありません)。 フィールドパネルのすべてのエントリには、インデックスを示すサフィックスが示されています (MyField #0 など)。
bCheckIt	(オプション) ウェイジエットにチェックを付けるかどうかを指定します。デフォルトは true です。

例

チェックボックスにチェックを付けます。

```
var f = this.getField("ChkBox");
f.checkThisBox(0,true);
```

clearItems

4.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
-----	-------------------------------------	--------------------------	-------------------------------------

リストボックスまたはコンボボックスのすべての値をクリアします。

関連するメソッドやプロパティとしては、[numItems](#)、[getItemAt](#)、[deleteItemAt](#)、[currentValueIndices](#)、[insertItemAt](#)、[setItems](#) があります。

例

「myList」フィールドのすべての項目をクリアします。

```
this.getField("myList").clearItems();
```

defaultIsChecked

5.0	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
-----	----------------------------------	-----------------------	-----------------------

指定したウェイジエットをデフォルトでチェックするかどうかを設定します。

注意： ラジオボタングループで書き出し値が重複していない場合は、チェックを付けたいウェイジエットの書き出し値を `defaultValue` に渡することで、デフォルトでチェックを付けることができます (デフォルトでチェックをすべて解除するには、空の文字列を渡します)。

パラメータ

nWidget	フィールドのラジオボタンウィジェットまたはチェックボックスウィジェットのインデックス（0 から数えます）。このインデックスは、そのフィールドの各ウィジェットが作成された順番を表しています（タブの順序とは関係ありません）。
bIsDefaultChecked	（オプション） <code>true</code> （デフォルト）の場合、ウィジェットはデフォルトで（例えば、フィールドがリセットされた場合に）チェックされます。 <code>false</code> の場合、チェックはデフォルトで解除されます。

戻り値

成功した場合は `true`。

例

「ChkBox」がデフォルトでチェックされるように変更してから、フィールドをリセットしてデフォルト値を反映させます。

```
var f = this.getField("ChkBox");
f.defaultIsChecked(0,true);
this.resetForm(["ChkBox"]);
```

deleteItemAt

4.0	D		F
-----	---	--	---

コンボボックスまたはリストボックスの項目を削除します。

リストボックスの場合、現在選択されている項目を削除すると、どの項目も選択されていない状態になります。そのフィールドに対してパラメータを指定せずに再度このメソッドを実行すると、削除する項目が選択されていないので、動作が不安定になることがあります。したがって、このメソッドを正しく実行するためには、(`currentValueIndices` メソッドを使用するなどして) あらかじめ項目を選択しておく必要があります。

パラメータ

nIdx	（オプション）リストから削除する項目のインデックス（0 から数えます）。指定しなかった場合は、現在選択されている項目が削除されます。
------	--

例

リストで現在選択されている項目を削除してから、リストの最初の項目を選択します。

```
var a = this.getField("MyListBox");
a.deleteItemAt(); // 現在の項目を削除し、
a.currentValueIndices = 0; // リストの一番上の項目を選択
```

getArray

3.01			
------	--	--	--

この Field オブジェクト（親フィールド）に属しているターミナルフィールド（値を持つことができる子フィールド）の配列を返します。

戻り値

Field オブジェクトの配列。

例

親フィールドに属している子フィールドの値を使用して計算します。

```
// f には f.v1、f.v2、f.v3 の 3 つの子がある
var f = this.getField("f");
var a = f.getArray();
var v = 0.0;
for (j = 0; j < a.length; j++) v += a[j].value;
// フィールド「f」のすべての子の合計が v に設定される
```

getItemAt

3.01			
------	--	--	--

コンボボックスまたはリストボックスの項目の内部値を取得します。

リストに含まれている項目数は、`numItems` を使用して取得できます。[insertItemAt](#)、[deleteItemAt](#)、[clearItems](#)、[currentValueIndices](#)、[setItems](#) も参照してください。

パラメータ

nIdx	リスト内の項目のインデックス（0 から数えます）。-1 は、リストの最後の項目を表します。
bExportValue	<p>（オプション、Acrobat 5.0）書き出し値を返すかどうかを指定します。</p> <ul style="list-style-type: none">• <code>true</code>（デフォルト）の場合、項目に書き出し値があれば、その値を返します。書き出し値がない場合は、項目名を返します。• <code>false</code> の場合は、項目名を返します。

戻り値

指定した項目の書き出し値または名前。

例

次の 2 つの例では、「myList」に 3 つの項目が含まれていると仮定しています。「First」の書き出し値は 1、「Second」の書き出し値は 2、「Third」には書き出し値がないとします。

```
// リストの最初の項目の値 (1) を返す
var f = this.getField("myList");
var v = f.getItemAt(0);
```

次の例は、オプションの第 2 パラメータの使用方法を示しています。このパラメータを `false` に設定すると、書き出し値が存在する場合でも項目名（表示値）を取得できます。

```
for (var i=0; i < f.numItems; i++)
    console.println(f.getItemAt(i,true) + ": " + f.getItemAt(i,false));
```

コンソールの出力は、次のようにになります。

1:	First
2:	Second
Third:	Third

getLock

6.0			
-----	--	--	--

`Lock` オブジェクトを取得します。これは、署名フィールドのロックプロパティを含む汎用オブジェクトです。

[setLock](#) も参照してください。

戻り値

フィールドの `Lock` オブジェクト。

insertItemAt

3.01	①		②
------	---	--	---

コンボボックスまたはリストボックスに新しい項目を挿入します。

関連するメソッドやプロパティとしては、[numItems](#)、[getItemAt](#)、[deleteItemAt](#)、[clearItems](#)、[currentValueIndices](#)、[setItems](#) があります。

パラメータ

cName	フォームに表示される項目名。
cExport	(オプション) この項目が選択されたときのフィールドの書き出し値。指定しなかった場合は、 <code>cName</code> が書き出し値として使用されます。
nIdx	(オプション) リスト内の項目の挿入箇所を示すインデックス。0（デフォルト）の場合、新しい項目はリストの先頭に挿入されます。-1 の場合、新しい項目はリストの最後に挿入されます。

例

```
var l = this.getField("myList");
l.insertItemAt("sam", "s", 0); /* sam をリスト 1 の一番上に挿入 */
```

isBoxChecked

5.0			
-----	--	--	--

指定のウィジェットにチェックが付いているかどうかを判定します。

注意：ラジオボタングループで書き出し値が重複していない場合は、`value` を取得することで、現在チェックが付いているウィジェットの書き出し値がわかります（チェックが付いているウィジェットがない場合は、空の文字列が返されます）。

パラメータ

nWidget	フィールドのラジオボタンウィジェットまたはチェックボックスウィジェットのインデックス（0 から数えます）。このインデックスは、そのフィールドの各ウィジェットが作成された順番を表しています（タブの順序とは関係ありません）。 フィールドパネルのすべてのエントリには、インデックスを示すサフィックスが示されています（MyField #0 など）。
---------	---

戻り値

指定のウィジェットに現在チェックが付いている場合は `true`、それ以外の場合は `false`。

例

チェックボックスの状態を確認し、警告ボックスにそれを表示します。

```
var f = this.getField("ChkBox");
var cbStatus = (f.isBoxChecked(0)) ? " " : " not ";
app.alert("The box is" + cbStatus + "checked");
```

isDefaultChecked

5.0			
-----	--	--	--

指定のウィジェットにデフォルトで（例えば、フィールドがリセットされた場合に）チェックが付けられるかどうかを判定します。

注意：ラジオボタングループで書き出し値が重複していない場合は、`defaultValue` を取得することで、デフォルトでチェックが付けられるウィジェットの書き出し値がわかります（デフォルトでチェックが付けられるウィジェットがない場合は、「Off」が返されます）。

パラメータ

nWidget	フィールドのラジオボタンウィジェットまたはチェックボックスウィジェットのインデックス (0 から数えます)。このインデックスは、そのフィールドの各ウィジェットが作成された順番を表しています (タブの順序とは関係ありません)。 フィールドパネルのすべてのエントリには、インデックスを示すサフィックスが示されています (MyField #0 など)。
---------	--

戻り値

指定のウィジェットにデフォルトでチェックが付けられる場合は `true`、それ以外の場合は `false`。

例

指定のチェックボックスにデフォルトでチェックが付けられるかどうかを判別します。

```
var f = this.getField("ChkBox");
var cbStatus = (f.isDefaultChecked(0)) ? "Checked" : "Unchecked";
app.alert("The Default: " + cbStatus);
```

setAction



トリガに対するフィールドの JavaScript アクションを設定します。

関連するメソッドとしては、Bookmark オブジェクトの [setAction](#) メソッドと、Doc の [setAction](#)、[addScript](#)、[setPageAction](#) メソッドがあります。

注意：このメソッドを実行すると、選択されたトリガで既に定義されているアクションが上書きされます。

パラメータ

cTrigger	アクションのトリガを示す文字列。有効な値は、次のとおりです。 MouseUp MouseDown MouseEnter MouseExit OnFocus OnBlur Keystroke Validate Calculate Format
----------	--

リストボックスの選択の変更イベントには、Keystroke トリガを使用します。

cScript	トリガが発生したときに実行する JavaScript コード。
---------	---------------------------------

例

`addField` を使用してボタンフィールドを作成し、プロパティを追加し、`setAction` を使用して「マウスボタンを放す」アクションを追加します。

```
var f = this.addField("actionField", "button", 0, [20, 100, 100, 20]);
f.setAction("MouseUp", "app.beep(0);");
f.delay = true;
f.fillColor = color.ltGray;
f.buttonSetCaption("Beep");
f.borderStyle = border.b;
f.lineWidth = 3;
f.strokeColor = color.red;
f.highlight = highlight.p;
f.delay = false;
```

setFocus

4.05			
------	--	--	--

フィールドにキーボードフォーカスを設定します。フォーカスを設定したフィールドが画面の外にある場合は、そのフィールドを表示するために、表示ページの変更やスクロールが行われます。フィールドが含まれている文書が手前に表示されていない場合は、手前に表示されます。

[bringToFront](#) も参照してください。

例

開いている文書を検索して、目的のフィールドにフォーカスします。このスクリプトで使用している `app.activeDocs` は、`disclosed` プロパティが `true` の文書か、コンソールイベント、バッチャイベント、メニューイベントでのみ実行できます。

```
var d = app.activeDocs;
for (var i = 0; i < d.length; i++) {
    if (d[i].info.Title == "Response Document") {
        d[i].getField("name").value="Enter your name here: "
        // 文書も手前に表示
        d[i].getField("name").setFocus();
        break;
    }
}
```

setItems

4.0	D		F
-----	---	--	---

コンボボックスまたはリストボックスの項目のリストを設定します。

関連するメソッドやプロパティとしては、[numItems](#)、[getItemAt](#)、[deleteItemAt](#)、[currentValueIndices](#)、[clearItems](#) があります。

パラメータ

oArray	文字列に変換可能なオブジェクトまたは配列を要素とする配列。 <ul style="list-style-type: none">文字列に変換可能な要素は、名前と書き出し値がその文字列に等しいリスト項目を表します。配列要素は、文字列に変換可能な 2 つのサブ要素で構成されます。サブ要素の最初が項目の名前、2 番目が書き出し値を表します。
--------	---

例

各種のフィールド（コンボボックスまたはリストボックス）を取得し、様々な方法でリストの項目を設定します。

```
var l = this.getField("ListBox");
l.setItems(["One", "Two", "Three"]);

var c = this.getField("StateBox");
c.setItems([["California", "CA"], ["Massachusetts", "MA"],
           ["Arizona", "AZ"]]);

var c = this.getField("NumberBox");
c.setItems(["1", 2, 3, ["PI", Math.PI]]);
```

setLock

6.0			
-----	--	--	--

この署名フィールドが署名されたときにロックするフィールドを制御します。ロックしたフィールドは変更できなくなります。署名がクリアされると、ロックされていたすべてのフィールドのロックが解除されます。このプロパティを取得するには、`getLock` を使用します。

注意：このメソッドを実行できるのは、バッチイベント、アプリケーション初期化イベント、コンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

このメソッドは、既に署名されている文書内のフィールドには適用できません。

パラメータ

oLock	ロックプロパティを含む <code>Lock</code> オブジェクト。
-------	---------------------------------------

戻り値

成功した場合は `true`、それ以外の場合は `false` または例外。

Lock オブジェクト

ロックプロパティを含む汎用 JavaScript オブジェクト。このオブジェクトは `setLock` に渡し、署名フィールドの `getLock` によって返されます。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
<code>action</code>	文字列	R / W	言語に依存しない、アクションの名前。有効な値は、次のとおりです。 <code>All</code> — 文書に含まれているすべてのフィールドがロックされます。 <code>Include</code> — <code>fields</code> で指定されているフィールドのみがロックされます。 <code>Exclude</code> — <code>fields</code> で指定されているフィールド以外のすべてのフィールドがロックされます。
<code>fields</code>	文字列の配列	R / W	フィールド名を表す文字列の配列。 <code>action</code> で <code>Include</code> または <code>Exclude</code> を指定した場合は、このプロパティも指定する必要があります。

signatureGetModifications

7.0			
-----	--	--	--

署名フィールドに署名された後に文書に加えられた変更に関する情報が含まれているオブジェクトを返します。この情報には、署名されたときの文書と現在の文書との違いのみが含まれています。一時オブジェクト（署名の後に追加されたが、その後削除されたオブジェクトなど）については報告されません。

戻り値

変更情報が含まれているオブジェクト。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	説明
<code>formFieldsCreated</code>	Field オブジェクトの配列	署名の後に作成されたフォームフィールドの配列。
<code>formFieldsDeleted</code>	汎用オブジェクトの配列	署名の後に削除されたフォームフィールドの配列。配列内の汎用オブジェクトは、 <code>name : type</code> という形式の文字列です。
<code>formFieldsFilledIn</code>	Field オブジェクトの配列	署名の後に入力されたフォームフィールドの配列。
<code>formFieldsModified</code>	Field オブジェクトの配列	署名の後に変更されたフォームフィールドの配列。ここでは、入力されたフォームフィールドは、変更されたフォームフィールドとは見なされず、除外されます。
<code>annotsCreated</code>	Annotation オブジェクトの配列	署名の後に作成された注釈の配列。注釈が一時的なもの（動的に作成されたポップアップなど）である場合、対応する配列要素は、 <code>author : name : type</code> という形式の文字列になります。

プロパティ	型	説明
annotsDeleted	汎用オブジェクトの配列	署名の後に削除された注釈の配列。配列内の汎用オブジェクトは、 <code>author : name : type</code> という形式の文字列です。
annotsModified	Annotation オブジェクトの配列	署名の後に変更された注釈の配列。注釈が一時的なもの（動的に作成されたポップアップなど）である場合、対応する配列要素は、 <code>author : name : type</code> という形式の文字列になります。
numPagesCreated	整数	署名の後に追加されたページの数。
numPagesDeleted	整数	署名の後に削除されたページの数。
numPagesModified	整数	署名の後に内容が変更されたページの数。ここでは、注釈やフォームフィールドの追加、削除、変更は、内容の変更とは見なされず、除外されます。
spawnedPagesCreated	文字列の配列	署名の後に作成されたページのリスト。作成されたページごとに、ソーステンプレートの名前が示されます。
spawnedPagesDeleted	文字列の配列	署名の後に作成されたが、削除されたページのリスト。作成されたページごとに、ソーステンプレートの名前が示されます。
spawnedPagesModified	文字列の配列	署名の後に作成されたが、変更されたページのリスト。作成されたページごとに、ソーステンプレートの名前が示されます。

例

変更情報をコンソールに書き込みます。

```
var sigField = this.getField( "mySignature" );
var sigMods = sigField.signatureGetModifications();

var formFieldsCreated = sigMods.formFieldsCreated;
for( var i = 0; i < formFieldsCreated.length; i++ )
    console.println( formFieldsCreated[i].name );

var formFieldsDeleted = sigMods.formFieldsDeleted;
for( var i = 0; i < formFieldsDeleted.length; i++ )
    console.println( formFieldsDeleted[i].name );

var formFieldsFilledIn = sigMods.formFieldsFilledIn;
for( var i = 0; i < formFieldsFilledIn.length; i++ )
    console.println( formFieldsFilledIn[i].name );
```

```
var formFieldsModified = sigMods.formFieldsModified;
for( var i = 0; i < formFieldsModified.length; i++ )
    console.println( formFieldsModified[i].name );

var spawnedPages = sigMods.spawnedPagesCreated;
for( var i = 0; i < spawnedPages.length; i++ )
    console.println( spawnedPages[i] );

console.println( sigMods.numPagesDeleted );
```

signatureGetSeedValue

6.0			
-----	--	--	--

署名フィールドのシード値プロパティを含む `SeedValue` オブジェクトを返します。シード値は、署名の表示、署名の理由、署名者などの署名のプロパティを制御する場合に使用されます。

[signatureSetSeedValue](#) を参照してください。

戻り値

`SeedValue` オブジェクト。

例

署名フィールドのシード値にアクセスします。

```
var f = this.getField( "sig0" );
var seedValue = f.signatureGetSeedValue();
// シード値のフィルタとフラグを表示
console.println( "Filter name:" + seedValue.filter );
console.println( "Flags:" + seedValue.flags );
// 証明書のシード値の制限を表示
var certSpec = seedValue.certspec;
console.println( "Issuer:" + certspec.issuer );
```

signatureInfo

5.0			
-----	--	--	--

署名のプロパティを含む `signatureInfo` オブジェクトを返します。このオブジェクトは、このメソッドが呼び出された時点における署名のスナップショットです。セキュリティハンドラによって、そのセキュリティハンドラ固有のプロパティが追加されることもあります。

注意：このメソッドはいつでも呼び出せますが、指定のセキュリティハンドラ (`osig`) が常に使用できるとは限りません。詳しくは、[security.getHandler](#) メソッドを参照してください。

署名が検証されるまで、電子署名ハンドラの一部のプロパティ (`certificates` などの `SignatureInfo` オブジェクトのプロパティ) は `null` 値を返すことがあります。したがって、`signatureValidate` の後に `signatureInfo` を再度呼び出す必要があります。

パラメータ

oSig	(オプション) 署名プロパティの取得に使用する SecurityHandler オブジェクト。指定しなかった場合は、ユーザ環境設定に基づいてセキュリティハンドラが設定されます（通常は、署名の作成に使用したハンドラになります）。
------	---

戻り値

署名のプロパティを含む SignatureInfo オブジェクト。このオブジェクトは、署名フィールドや FDF オブジェクトに署名したり、FDF オブジェクトの signatureValidate メソッドを呼び出したりするときにも使用します。

例

SignatureInfo オブジェクトのプロパティにアクセスします。

```
// すべての情報を取得
var f = getField( "Signature1" );
f.signatureValidate();
var s = f.signatureInfo();
console.println( "Signature Attributes:" );
for(i in s) console.println( i + " = " + s[i] );

// 特定の情報を取得
var f = this.getField("Signature1"); // ppklite 署名ハンドラを使用
var Info = f.signatureInfo();
// いくつかの標準の signatureInfo プロパティ
console.println("name = " + Info.name);
console.println("reason = " + Info.reason);
console.println("date = " + Info.date);

// PPKLite 固有の signatureInfo プロパティ
console.println("contact info = " + Info.contactInfo);

// 最初の（唯一の）証明書を取得
var certificate = Info.certificates[0];

// 署名者の共通名
console.println("subjectCN = " + certificate.subjectCN);
console.println("serialNumber = " + certificate.serialNumber);

// 署名者の識別名に関する情報を表示
console.println("subjectDN.cn = " + certificate.subjectDN.cn);
console.println("subjectDN.o = " + certificate.subjectDN.o);
```

signatureSetValue

6.0	D	S	X
-----	---	---	---

署名フィールドの署名に使用するプロパティを設定します。このプロパティは、署名フィールドに保存されます。フィールドに署名したり、署名をクリアしたり、resetForm を呼び出したりしても変更されません。このプロパティの設定を取得するには、signatureGetSeedValue を使用します。

シード値を設定するときは、次のことに注意してください。

- 署名された文書では、シード値を設定しないでください。証明済み文書では、シード値の設定は行えません。シード値は、主に未署名の文書でフィールドを設定するために使用します。
- シード値を設定すると、Acrobat でデフォルト設定が表示されなかったり、使用されなかったりすることがあります。例えば、プリセットの署名理由はレジストリに格納されていますが、署名理由を指定するとレジストリが無視され、カスタムの理由のみが表示されます。

注意：このメソッドを実行できるのは、バッチイベント、アプリケーション初期化イベント、コンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

証明用の署名とは、SignatureInfo オブジェクトの mdp プロパティの値が allowNone、default または defaultAndComments である署名です。

Adobe Reader では使用できません。

署名シード値の使用法について詳しくは、『Acrobat 8.0 Security Feature User Reference』を参照してください。

パラメータ

oSigSeedValue	署名シード値プロパティを含む SeedValue オブジェクト。
---------------	----------------------------------

SeedValue オブジェクト

署名シード値を表す汎用 JavaScript オブジェクト。signatureSetSeedValue に渡し、signatureGetSeedValue によって返されます。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
certspec	オブジェクト	R / W	シード値の CertificateSpecifier オブジェクト。このオブジェクトのプロパティについては、 466 ページ を参照してください。
digestMethod	文字列の配列	R / W	(Acrobat 8) 署名に使用可能なダイジェストメソッドの配列。有効な文字列値は、SHA1、SHA256、SHA384、SHA512、RIPEMD160 です。
			<p>注意：これが使用できるのは、DigitalID に RSA の公開鍵／秘密鍵が含まれている場合のみです。DSA の公開鍵／秘密鍵が含まれている場合、値は常に SHA1 になります。</p> <p>この制限が必須であるかどうかは、flags プロパティで示されます。</p>
filter	文字列	R / W	言語に依存しない、署名に使用するセキュリティハンドラの名前。
			<p>filter が指定されており、この制限が必須であることが flags プロパティで示されている場合は、このエントリで指定されている署名ハンドラを使用して署名する必要があります。それ以外の署名ハンドラでは署名できません。この制限がオプションであることが flags で示されている場合は、使用可能であればこのハンドラを使用しますが、使用できない場合は別のハンドラを使用できます。</p>

プロパティ	型	アクセス	説明
flags	数値	R / W	<p>このオブジェクトの必須プロパティを表す一連のビットフラグ。この値は、次の値の論理和です。各ビットは、該当するプロパティが必須である場合に設定します。</p> <ul style="list-style-type: none"> 1: filter 2: subFilter 4: version 8: reasons 16: legalAttestations 32: shouldAddRevInfo 64: digestMethod <p>使用例：1 = フィルタ、3 = フィルタとサブフィルタ、11 = フィルタ、サブフィルタ、理由。このフィールドがない場合、すべてのプロパティはオプションになります。</p>
legalAttestations	文字列の配列	R / W	<p>(Acrobat 7.0) MDP 用の（証明用の）署名を作成するときに使用できる法的な証明のリスト。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p>
mdp	文字列	R / W	<p>(Acrobat 7.0) フィールドの署名時に使用する MDP 設定。有効な値は、次のとおりです。</p> <ul style="list-style-type: none"> allowNone default defaultAndComments <p>allowAll も有効な値ですが、これを使用すると MDP の効果が取り消され、証明用の署名がこのフィールドで使用できなくなります。したがって、この署名は証明用の署名ではなく、承認用の署名になります。</p> <p>値には一意の識別子を使用します。653 ページの「MDP 値」の表を参照してください。</p>
reasons	文字列の配列	R / W	<p>署名時に使用可能な理由のリスト。</p> <p>(Acrobat 8.0) この配列が、ピリオド「.」という単一の要素のみで構成されており、reasons が必須であることが flags プロパティで示されている場合は、理由を指定するための UI が署名時に表示されなくなります。これは、ユーザの環境設定をオーバーライドします。</p> <p>この配列が空であり、reasons が必須であることが示されている場合は、例外が発生します。</p>

プロパティ	型	アクセス	説明
shouldAddRevInfo	ブーリアン	R / W	(Acrobat 8) true に設定すると、署名に使用された証明書（と該当するチェーン）の失効確認が行われ、すべての失効情報が署名に含められます。 これは、subFilter が adbe.pkcs7.detached または adbe.pkcs7.sha1 の場合にのみ有効です。 adbe.x509.rsa_sha1 の場合は使用できません。したがって、subFilter が adbe.x509.rsa_sha1 であり、失効情報の追加を指定した場合、署名処理は失敗します。 shouldAddRevInfo が true であり、この制約が必須であることが flags プロパティで示されている場合は、前述のタスクを実行する必要があります。このタスクを実行できない場合、署名は失敗します。 デフォルト値は false です。
subFilter	文字列の配列	R / W	署名に使用可能な形式の配列。既知の形式のリストについては、Signature Info オブジェクトの subFilter プロパティを参照してください。 署名ハンドラでサポートされているエンコーディングに一致する、配列内の最初のエンコーディングが署名に使用されます。subFilter が指定されており、この制限が必須であることが flags プロパティで示されている場合は、最初に一致するエンコーディングを使用して署名する必要があります。それ以外のエンコーディングでは署名できません。この制限がオプションであることが flags プロパティで示されている場合は、使用可能であれば最初に一致するエンコーディングを使用しますが、使用できない場合は別のエンコーディングを使用できます。
timeStampspec	オブジェクト	R / W	(Acrobat 7.0) シード値の timeStamp 指定子オブジェクト。url プロパティと flags プロパティを使用してタイムスタンプサーバを指定します。 469 ページ を参照してください。
version	数値	R / W	署名フィールドの署名に使用する署名ハンドラの最低バージョン。有効な値は、1 と 2 です。値が 1 の場合は、PDF 1.5 で定義されたすべてのシード値ディクショナリエントリをパーサが認識できる必要があります。値が 2 の場合は、PDF 1.7 以前に定義されたすべてのシード値ディクショナリエントリをパーサが認識できる必要があります。 この制限が必須であるかどうかは、flags プロパティで示されます。
注意 : 『PDF Reference』バージョン 1.6 以前では、version エントリ (version パラメータに対応するキー) が整数型であると誤って記載されています。このエントリは実数型です。			

CertificateSpecifier オブジェクト

この汎用 JavaScript オブジェクトには、署名シード値の証明書指定子プロパティが含まれています。SeedValue オブジェクトの certSpec プロパティで使用します。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
flags	数値	R / W	<p>このオブジェクトの必須プロパティを表す一連のビットフラグ。この値は、次の値の論理和です。各ビットは、該当するプロパティが必須である場合に設定します。</p> <p>1: subject 2: issuer 4: oid 8: subjectDN 16: Reserved 32: keyUsage 64: url</p> <p>このフィールドがない場合、すべてのプロパティはオプションになります。</p>
issuer	Certificate オブジェクトの配列	R / W	<p>署名に使用可能な Certificate オブジェクトの配列。</p> <p>注意：これを指定した場合、署名の証明書は、この配列に含まれているいずれかの証明書と完全に一致する証明書によって発行されている必要があります。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p>

プロパティ	型	アクセス	説明
keyUsage	オブジェクト	R / W	<p>(Acrobat 8.0) 署名の証明書に存在する必要がある使用可能な鍵使用エクステンションを表す整数の配列。各整数は、次のように構成されます。</p> <p>それぞれの鍵使用タイプ（RFC 3280 で定義）について、最下位ビットから 2 ビットずつ指定します。</p> <ul style="list-style-type: none"> digitalSignature (ビット 2、1) nonRepudiation (4、3) keyEncipherment (6、5) dataEncipherment (8、7) keyAgreement (10、9) keyCertSign (12、11) cRLSign (14、13) encipherOnly (16、15) decipherOnly (18、17) <p>この 2 ビット値の意味は、次のようになります。</p> <ul style="list-style-type: none"> 00 : 該当する keyUsage が設定されていない 01 : 該当する keyUsage が設定されている 10 と 11 : 該当する keyUsage の状態は問題にならない <p>例えば、digitalSignature という鍵使用タイプが設定されている必要があり、他のすべての鍵使用タイプの状態は問題にならない場合、対応する整数は 0x7FFFFFFD になります。つまり、digitalSignature を表すビット 2 と 1 を 01 に設定し、他のすべての鍵使用タイプを 11 に設定します。</p> <p>この制限が必須であるかどうかは、flags プロパティの該当するフラグ値で示されます。</p>
oid	文字列の配列	R / W	<p>署名の証明書に存在する必要がある Policy OID を表す文字列の配列。このプロパティが使用できるのは、issuer プロパティが存在する場合のみです。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p>

プロパティ	型	アクセス	説明
subject	Certificate オブジェクトの 配列	R / W	<p>署名に使用可能な Certificate オブジェクトの配列。</p> <p>注意：これを指定した場合、署名の証明書は、この配列に含まれているいずれかの証明書と完全に一致する必要があります。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p>
subjectDN	オブジェクトの 配列	R / W	<p>(Acrobat 8.0) 署名に使用可能なユーザの識別名 (DN) を表すオブジェクトの配列。署名の証明書は、DN で指定されている属性がすべて含まれていれば、DN の要件を満たします（したがって、証明書に他の DN 属性が含まれていてもかまいません）。このエントリを指定した場合、署名の証明書は、この配列に含まれている 1 つ以上の DN を満たす必要があります。</p> <p>DN 属性の制限を指定するには、その属性をプロパティとしてこのオブジェクトに追加します。プロパティのキーには、該当する属性の名前と OID (RFC 3280 で定義) の両方が使用できます。プロパティの値は、文字列型であることが必要です。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p> <p>例 1 : {cn: "Alice", ou: "Engineering", o: "Acme Inc"}。DN 属性の名前 (cn、o、ou など) は、RDN で定義されているものです (587 ページの RDN オブジェクトを参照)。</p> <p>例 2 : {cn: "Joe Smith", ou: "Engineering", 2.5.4.43: "JS"}。OID の 2.5.4.43 は、「Initials」属性との一致を表します。この DN を定義するサンプルコードを次に示します。</p> <pre>var subjectDN = {cn: "Joe Smith", ou: "Engineering"}; subjectDN["2.5.4.43"] = "JS";</pre> <p>値の型が DirectoryString や IA5String の属性は、この例で示したように指定できます。それ以外の型の値（例えば、GeneralizedTime 型の dateOfBirth など）は、16 進エンコードされたバイナリ文字列として指定する必要があります。</p> <p>様々な属性とその型について詳しくは、RFC 3280 を参照してください。</p>

プロパティ	型	アクセス	説明
url	文字列	R / W	<p>一致する証明書が見つからない場合に、新しい証明書を登録するために使用する URL。</p> <p>このプロパティに、DigitalID ストア Web サービス（サーバベースでの署名を可能にする署名 Web サービスなど）の URL を指定すれば、その Web サービスで提供されている任意の DigitalID で署名できるようになります。</p> <p>この制限が必須であるかオプションであるかは、flags プロパティの該当するフラグ値で示されます。</p>
urlType	文字列	R / W	<p>(Acrobat 8.0) url プロパティで指定されている URL のタイプを示す文字列。Acrobat 8.0 で有効な値は、次のとおりです。</p> <p>「HTML」：URL は HTML の Web サイトを示します。その内容は Web ブラウザで表示されます。flags プロパティの url ビットの値は無視されます。</p> <p>「ASSP」：URL は、サーバベースの署名を可能にする署名 Web サービスを示します。この制約が必須であることが flags プロパティの url ビットで示されている場合は、このサーバで提供されている証明書で署名されている必要があります。</p> <p>この属性が存在しない場合は、HTML サイトを示しているものと見なされます。</p>

シード値 timeStamp 指定子オブジェクト

シード値 timeStamp 指定子オブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
url	文字列	R / W	RFC 3161 に準拠したタイムスタンプを提供するタイムスタンプサーバの URL。
flags	数値	R / W	タイムスタンプが必須である (1) か必須でない (0) を制御するビットフラグ。デフォルトは 0 です。

例 1

未署名の署名フィールドに、MDP、法的な証明、有効な署名理由のシード値を設定します。

```
// 署名 Field オブジェクトを取得
var f = this.getField("mySigField");
f.signatureSetValue({
```

```
        mdp: "defaultAndComments",
        legalAttestations: ["Trust me and be at ease.",
            "You can surely trust the author."],
        reasons: ["This is a reason", "This is a better reason"],
        flags: 8
    })
}
```

例 2

署名ハンドラを PPKMS に設定し、フォーマットを「adbe.pkcs7.sha1」に設定します。

```
var f = this.getField( "mySigField" );

f.signatureSetSeedValue({
    filter: "Adobe.PPKMS",
    subFilter: ["adbe.pkcs7.sha1"],
    flags: 3
});
```

例 3

署名ハンドラを PPKLite に設定し、署名者の証明書の発行者を caCert に設定します。両方のシード値を必須にしているので、いずれかの制限が満たされない場合は、署名が失敗します。

```
var caCert = security.importFromFile("Certificate", "/C/temp/CA.cer");
f.signatureSetSeedValue({
    filter: "Adobe.PPKLite",
    certspec: {
        issuer: [caCert],
        url: "http://www.example.com/enroll.html",
        flags : 2
    },
    flags: 1
});
```

signatureSign

5.0			
-----	--	--	--

指定のセキュリティハンドラでフィールドに署名します。security.[getHandler](#) および securityHandler.[login](#) も参照してください。

注意：このメソッドを実行できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、event オブジェクトを参照してください。

既に署名されている署名フィールドに署名することはできません。resetForm を使用して署名フィールドをクリアしてください。

パラメータ

<code>oSig</code>	署名に使用する <code>SecurityHandler</code> オブジェクトを指定します。指定したハンドラで署名処理がサポートされていない場合は、例外が発生します。セキュリティハンドラの中には、署名を行う前にユーザのログインを必要とするものもあります。 <code>osig</code> を指定しなかった場合は、ユーザ環境設定に基づいてハンドラを選択するか、ハンドラを選択するためのプロンプトを表示します (<code>bUI</code> が <code>true</code> の場合)。
<code>oInfo</code>	(オプション) 署名の書き込み可能なプロパティを設定する <code>SignatureInfo</code> オブジェクト。 signatureInfo も参照してください。
<code>cDIPath</code>	(オプション) デバイスに依存しない、署名後のファイルの保存先パス。指定しなかつた場合、ファイルは元の場所に保存されます。
<code>bUI</code>	(オプション、Acrobat 6.0) 署名時にユーザインターフェイスを表示するかどうかを指定するブーリアン値。 <code>true</code> の場合は、 <code>oInfo</code> と <code>cDIPath</code> が署名ダイアログボックスのデフォルト値として使用されます。 <code>false</code> (デフォルト) の場合は、ユーザインターフェイスを使用せずに署名が行われます。
<code>cLegalAttest</code>	(オプション、Acrobat 6.0) 証明用の署名を作成するときに提供する文字列。 証明用の署名とは、 <code>SignatureInfo</code> オブジェクトの <code>mdp</code> プロパティが <code>allowAll</code> 以外の値を持つ署名です。証明用の署名を作成すると、文書内の法的な警告がスキャンされて、文書に埋め込まれます。検出された法的な警告は、 <code>Doc</code> の <code>getLegalWarnings</code> メソッドで取得できます。警告を埋め込むときに、文書にその警告を適用した理由を附加することができます。

戻り値

署名が正常に行われた場合は `true`。それ以外の場合は `false`。

例 1

PPKLite 署名ハンドラで「Signature」フィールドに署名します。

```
var myEngine = security.getHandler( "Adobe.PPKLite" );
myEngine.login( "dps017", "/c/profile/dps.pfx" );
var f = this.getField("Signature");

// フィールドに署名
f.signatureSign( myEngine,
  { password: "dps017", // パスワードを指定
    location: "San Jose, CA", // ... 後述の注意を参照
    reason: "I am approving this document",
    contactInfo: "dpsmith@example.com",
    appearance: "Fancy" } );
```

注意：この例では、パスワードを指定しています。パスワードを指定する必要があるかどうかは、`PasswordTimeout` が切れているかどうかによります。`PasswordTimeout` は、`securityHandler.setPasswordTimeout` で設定できます。

例 2

証明用の署名フィールドに署名します。

```
var myEngine = security.getHandler( "Adobe.PPKLite" );
myEngine.login( "dps017", "/c/profile/dps.pfx" );

var f = this.getField( "AuthorSigFieldName" );
var s = { reason: "I am the author of this document",
          mdp: "allowNone" };
f.signatureSign({
  oSig: myEngine,
  oInfo: s,
  bUI: false,
  cLegalAttest: "To reduce file size, fonts are not embedded."
});
```

signatureValidate

5.0			
-----	--	--	--

署名フィールドの署名の完全性のステータスを検証して返します。署名に使用された署名ハンドラによっては、コンピュータに大きな負荷がかかり、大量の処理時間がかかることがあります。

注意：このメソッドはいつでも呼び出せますが、パラメータ `oSig` が常に使用できるとは限りません。詳しくは、[security.getHandler](#) を参照してください。

パラメータ

<code>oSig</code>	(オプション) 署名の検証に使用するセキュリティハンドラ。この値は、 <code>SecurityHandler</code> オブジェクトか <code>SignatureParameters</code> オブジェクトです。このハンドラを指定しなかった場合は、署名の <code>handlerName</code> プロパティで指定されているセキュリティハンドラが使用されます。
<code>bUI</code>	(オプション、Acrobat 6.0) <code>true</code> の場合は、検証時に必要に応じて UI が表示されます。検証ハンドラが指定されていない場合は、UI を使用して選択できます。デフォルトは <code>false</code> です。

戻り値

署名の完全性のステータス。有効な値は、次のとおりです。

- 1 — 署名フィールドではありません
- 0 — 署名が空です
- 1 — 未知のステータス
- 2 — 署名が無効です
- 3 — 文書の署名は有効ですが、署名者の ID が検証できません
- 4 — 文書の署名が有効で、署名者の ID も有効です。

[SignatureInfo](#) オブジェクトの `signVisible` プロパティと `statusText` プロパティを参照してください。

SignatureParameters オブジェクト

次のプロパティを含む汎用オブジェクト。各プロパティには、`signatureValidate` での検証に使用するセキュリティハンドラを指定します。

<code>oSecHdlr</code>	署名の検証に使用するセキュリティハンドラオブジェクト。
<code>bAltSecHdlr</code>	<code>true</code> の場合は、ユーザ環境設定に基づいて選択された別のセキュリティハンドラで署名を検証できます。デフォルトは <code>false</code> です。この場合は、署名の <code>handlerName</code> プロパティで返されるセキュリティハンドラが、署名の検証に使用されます。 <code>oSecHdlr</code> が指定されている場合、このパラメータは使用されません。

例

署名を検証してコンソールに表示します。

```
var f = this.getField("Signature1") // 署名フィールドを取得
var status = f.signatureValidate();
var sigInfo = f.signatureInfo();
if ( status < 3 )
    var msg = "Signature not valid! " + sigInfo.statusText;
else
    var msg = "Signature valid! " + sigInfo.statusText;
app.alert(msg);
```

FullScreen

フルスクリーンモード（プレゼンテーションモード）の環境設定やプロパティにアクセスできるインターフェイス。`fullScreen` オブジェクトを取得するには、`app.fs` を使用します。

FullScreen のプロパティ

backgroundColor

5.0			
-----	--	--	--

フルスクリーンモードでの画面の背景色。詳しくは、[191 ページの「カラー配列」](#) を参照してください。

型

カラー配列

アクセス

R / W

例

```
app.fs.backgroundColor = color.ltGray;
```

clickAdvances

5.0			
-----	--	--	--

ページ上の任意の場所をマウスでクリックすると、次のページに移動するかどうかを示します。

型

ブーリアン

アクセス

R / W

cursor

5.0			
-----	--	--	--

フルスクリーンモードでのポインタの動作を示します。cursor オブジェクトには、有効なカーソルの動作がすべて定義されています。

カーソルの動作	キーワード
常に非表示	cursor.hidden
数秒間だけ表示	cursor.delay
常に表示	cursor.visible

型

数値

アクセス

R / W

例

```
app.fs.cursor = cursor.visible;
```

defaultTransition

5.0			
-----	--	--	--

フルスクリーンモードでページを切り替える場合に使用するデフォルトの効果。ビューアでサポートされている有効な効果名のリストを取得するには、`transitions` を使用します。

「効果なし」に設定するには、`app.fs.defaultTransition = "";` とします。

型

文字列

アクセス

R / W

例

文書をプレゼンテーションモードにします。

```
app.fs.defaultTransition = "WipeDown";
app.fs.isFullScreen = true;
```

escapeExits

5.0	P	S	
-----	----------	----------	--

Esc キーでフルスクリーンモードを終了できるかどうかを指定するブーリアン値。

注意：(バージョン 8.0) バッヂイベントとコンソールイベント内でのみ本プロパティに `false` を設定できます。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、`event` オブジェクトを参照してください。

型

ブーリアン

アクセス

R / W

isFullScreen

5.0			
-----	--	--	--

`true` の場合、ビューアは通常の表示モードではなく、フルスクリーンモードになります。フルスクリーンモードは、1 つ以上の文書が開かれて表示されている場合にのみ可能です。

注意：Web ブラウザに表示されている PDF 文書をフルスクリーンモードにすることはできません。

型

ブーリアン

アクセス

R / W

例

```
app.fs.isFullScreen = true;
```

この例では、ビューアがフルスクリーンモードになります。`isFullScreen` が `false` であった場合は、デフォルトの表示モードが設定されます（デフォルトの表示モードとは、フルスクリーンモードになる前の Acrobat アプリケーションのモードのことです）。

loop

5.0	P		
-----	----------	--	--

フルスクリーンモードで（マウスクリック、キーボード、タイム設定によって）次のページを表示するときに、文書の最後のページに達したら最初のページに戻るかどうかを示します。

型

ブーリアン

アクセス

R / W

timeDelay

5.0	P		
-----	---	--	--

フルスクリーンモードで自動的に次のページを表示するまでのデフォルトの秒数。ページの自動切り替えを有効または無効にするには、[useTimer](#) を参照してください。

型

数値

アクセス

R / W

例

フルスクリーンのプロパティを設定してから、フルスクリーンモードにします。

```
app.fs.timeDelay = 5;           // 5 秒間隔
app.fs.useTimer = true;          // 自動的にページをめくる
app.fs.usePageTiming = true;     // ページごとのオーバーライドを許可
app.fs.isFullScreen = true;      // フルスクリーンに切り替える
```

transitions

5.0			
-----	--	--	--

ビューアに実装されている有効な効果名を表す文字列の配列。効果なしに設定するには、次のように defaultTransition を空の文字列に設定します。

```
app.fs.defaultTransition = "";
```

型

配列

アクセス

R

例

現在サポートされている効果名のリストを生成します。

```
console.println([" + app.fs.transitions + "]));
```

usePageTiming

5.0	P		
-----	---	--	--

フルスクリーンモードで自動的にページを切り替えるときに、各ページで設定されている値を使用するかどうかを示します。個別のページの効果のプロパティは、[setPageTransitions](#) を使用して設定します。

型

ブーリアン

アクセス

R / W

useTimer

5.0	P		
-----	---	--	--

フルスクリーンモードで自動的にページを切り替えるかどうかを示します。次のページに切り替えるまでのデフォルトの時間間隔を設定するには、`timeDelay` を使用します。

型

ブーリアン

アクセス

R / W

global

global オブジェクトは、文書間やセッション間でデータを共有したいときに使用できる、静的な JavaScript オブジェクトです。このようなデータは、パーシスタントグローバルデータと呼ばれます。文書間でのグローバルデータの共有や変更の通知は、subscription メカニズムを介して行われます。このメカニズムによってグローバル変数が監視され、その変更が各文書に通知されます。

注意：Acrobat 8 では、グローバル変数へのアクセス方法に変更が加えられています。

環境設定ダイアログボックス (Ctrl+K キー) の「JavaScript」分類に、「グローバルオブジェクトセキュリティポリシーを有効にする」というセキュリティ関連の新しいチェックボックスが追加されました。

- チェックを付けると（デフォルトでチェックが付いています）、グローバル変数を設定するたびに設定元が記憶され、その設定元のみがその変数にアクセスできるようになります。
 - 例えば、ファイルで変数を設定した場合は、そのファイル（変数を設定したときと同じパスにあるファイル）のみがその変数にアクセスできるようになります。
 - また、特定の URL にある文書で変数を設定した場合は、そのホストのみがその変数にアクセスできるようになります。

これらの制限には、重要な例外があります。グローバル変数の定義やアクセスが行えるのは、セキュリティによる制限がないコンテキスト（コンソール、バッチシーケンス、フォルダレベル JavaScript）のみです。セキュリティによる制限がないコンテキストについて詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

- チェックを解除すると、設定元でない文書もその変数にアクセスできるようになります（これは、8.0 より前のバージョンの Acrobat と同じ動作です）。

例については、[480 ページの「グローバルオブジェクトセキュリティポリシー」](#) を参照してください。

グローバルプロパティの作成

グローバルデータを指定するには、global オブジェクトにプロパティを追加します。プロパティの型は、文字列、ブーリアン値、数値のいずれかになります。

例えば、radius という変数を追加して、すべての文書のスクリプトからこの変数にアクセス可能にするには、スクリプトでプロパティを次のように定義します（「グローバルオブジェクトセキュリティポリシーを有効にする」のチェックは解除されているものとします）。

```
global.radius = 8;
```

これで、現在のビューアセッションのすべての文書からグローバル変数 radius を認識できるようになります。A.pdf と B.pdf の 2 つのファイルが開いており、A.pdf でこのグローバル変数を宣言したとします。A.pdf と B.pdf のどちらのファイルでも、global.radius を使用して球の体積を計算できます。

```
var v = (4/3) * Math.PI * Math.pow(global.radius, 3);
```

どちらのファイルでも、2144.66058 という同じ結果が得られます。global.radius の値を変更してスクリプトを再度実行すると、それに応じて v の値も変わります。

グローバルプロパティの削除

global オブジェクトから変数やプロパティを削除するには、`delete` 演算子を使用します。JavaScript の予約語である `delete` について詳しくは、JavaScript 1.5 のマニュアルを参照してください ([28 ページの「関連ドキュメント」](#) を参照)。

例えば、`global.radius` プロパティを削除するには、次のスクリプトを呼び出します。

```
delete global.radius
```

グローバルオブジェクトセキュリティポリシー

グローバルセキュリティポリシーが新たに導入されて、文書からのグローバル変数へのアクセスが制限されるようになりました。詳細や例外については、[479 ページの「global」](#) を参照してください。

例えば、`docA.pdf` という文書で、セキュリティによる制限があるコンテキスト（ボタンの「マウスボタンを放す」アクション）から次のスクリプトを実行します。

```
global.x = 1
global.setPersistent("x", true);
```

`global.x` 変数には、設定元として `docA.pdf` のパスが保存されます。したがって、`docA.pdf` はこの変数にアクセスできます。

```
console.println("global.x = " + global.x);
```

また、`docA.pdf` は、このグローバル変数を任意のコンテキストで設定できます (`global.x = 3` など)。

別のパスにある文書でこのグローバル変数を取得したり設定したりするには、権限レベルが引き上げられた信頼済みコンテキストで取得や設定を行う必要があります。次のスクリプトはフォルダレベルの JavaScript です。

```
myTrustedGetGlobal = app.trustedFunction ( function()
{
    app.beginPriv();
    var y = global.x;
    return y
    app.endPriv();
});
myTrustedSetGlobal = app.trustedFunction ( function(value)
{
    app.beginPriv();
    global.x=value;
    app.endPriv();
});
```

別の `docB.pdf` という文書から `global.x` 変数にアクセスするには、この信頼済み関数を使用します。

```
// 文書 B のマウスボタンを放すアクション
console.println("The value of global.x is " + myTrustedGetGlobal());
```

また、`docB.pdf` からこのグローバル変数を設定することもできます。

```
// docB.pdf から global.x を設定
myTrustedSetGlobal(2);
```

`docB.pdf` から `global.x` を設定すると、設定元が変更されます。したがって、`docA.pdf` から `global.x` に直接アクセスすることはできなくなります。アクセスするには、信頼済み関数を使用する必要があります。

```
// マウスボタンを放すアクションを docA.pdf から実行
console.println("The value of global.x is " + myTrustedGetGlobal());
```

global のメソッド

setPersistent

3.01			
------	--	--	--

指定の変数を、Acrobat のセッションをまたがって保持する（パースタントにする）かどうかを制御します。

パースタントグローバルデータにできるのは、ブーリアン、数値、文字列の変数のみです。

パースタントグローバル変数は、アプリケーションの終了時に、フォルダレベルスクリプトが保存されているユーザフォルダ内の `glob.js` というファイルに保存され、アプリケーションの起動時に再読み込みされます。Acrobat 6.0 以降では、このファイルに対して 2 ~ 4 KB のサイズ制限が行われます。この制限を超えたデータは、すべて切り捨てられます。

パースタントグローバル変数を指定するときは、一定の規則に従って命名してください。例えば、`myCompany_variableName` という形の名前にすることができます。これによって、各文書でパースタントグローバル変数が競合するのを防ぐことができます。

パラメータ

cVariable	パースタントにする変数（グローバルプロパティ）。
bPersist	<code>true</code> の場合、プロパティは Acrobat ビューアのセッションをまたがって保持されます。 <code>false</code> （デフォルト）の場合は、文書をまたがってアクセスすることはできませんが、Acrobat ビューアのセッションをまたがってアクセスすることはできなくなります。

例

`radius` プロパティをパースタントにして、他の文書からアクセスできるようにします。

```
global.radius = 8;                                // radius をグローバルとして宣言
global.setPersistent("radius", true);               // パースタントにする
```

これで、前述の体積計算のコードは、`global.radius` の値を変更しない限り、ビューアセッションをまたがって同じ結果が得られるようになります。

subscribe

5.0			
-----	--	--	--

グローバル変数の値が変更されたら、フィールドを自動的に更新することができます。`cVariable` で指定したプロパティが変更されると、別の文書内であっても、`fCallback` で指定した関数が呼び出されます。1つ のグローバルプロパティに対して、複数の `subscribe` が可能です。

パラメータ

cVariable	グローバルプロパティ。
fCallback	プロパティが変更されたときに呼び出す関数。

コールバック関数の構文は、次のとおりです。

```
function fCallback(newval) {  
    // newval は、subscribe で指定したグローバル変数の  
    // 新しい値。  
    <グローバル変数の新しい値を処理するコード>  
}
```

コールバック関数は、そのスクリプトを登録した文書が閉じられたり、(グローバル) プロパティが削除されたりすると、破棄されます。

例

この例では、「グローバルオブジェクトセキュリティポリシーを有効にする」のチェックが解除されているものとします。setRadius.pdf と calcVolume.pdf の 2 つのファイルが Acrobat または Adobe Reader で開いているとします。

setRadius.pdf には、次のコードを持つボタンが 1 つ含まれています。

```
global.radius = 2;
```

calcVolume.pdf では、subscribe という名前の次のコードが、文書レベルの JavaScript に定義されています。

```
global.subscribe("radius", RadiusChanged);  
function RadiusChanged(x)// コールバック関数  
{  
    var V = (4/3) * Math.PI * Math.pow(x,3);  
    this.getField("MyVolume").value = V;// テキストフィールドに値を設定  
}
```

両方のファイルが開いた状態で、setRadius.pdf のボタンをクリックすると、calcVolume.pdf の「MyVolume」というテキストフィールドの値が、直ちに 33.51032 に更新されます (global.radius = 2 が反映されます)。

HostContainer

このオブジェクトは、PDF 文書と、その文書が含まれているホストコンテナ（HTML ページなど）との間の通信を管理します。文書のホストコンテナは、Doc の hostContainer プロパティで取得できます。

これに対応する Embedded PDF オブジェクトには、コンテナアプリケーションのオブジェクトモデルに対するインターフェイスが用意されています。

HostContainer のプロパティ

messageHandler

7.0.5			
-------	--	--	--

ホストコンテナのスクリプトが postMessage メソッドを呼び出したときに呼び出される通知オブジェクト。このオブジェクトでは、次のプロパティやメソッドを定義できます。

メソッド／プロパティ	説明
onMessage	(オプション) postMessage に応答して呼び出されるメソッド。メッセージの送信は非同期で行われます。このメソッドには、postMessage に渡された配列を含む単一の配列パラメータが渡されます。
onError	(オプション) エラーに応答して呼び出されるメソッド。このメソッドには、Error オブジェクトと、エラーを発生させたメッセージを表す文字列の配列が渡されます。 Error オブジェクトの name プロパティは、次のいずれかに設定されます。 「MessageGeneralError」：一般エラーが発生しました。 「MessageNotAllowedError」：セキュリティ上の理由で操作が失敗しました。 エラーが発生してもこのプロパティが定義されていない場合、エラーは送信されません（メッセージと異なり、エラーはキューに入れられません）。
onDisclose	ホストアプリケーションから文書にメッセージを送信できるかどうかを調べるために呼び出される、必須のメソッド。PDF 文書の作成者は、このメソッドを設定して、メッセージを送信可能なセキュリティ上の条件を定義できます。このメソッドは、Doc / Open イベントで設定します。このメソッドには 2 つのパラメータが渡されます。 cURL — ホストコンテナの場所を示す URL（例えば、<OBJECT> タグが使用されている HTML ページの URL）。 cDocumentURL — 公開されているかどうか確認する PDF 文書の URL。 このメソッドで true を返すと、ホストコンテナがメッセージハンドラにメッセージを送信できるようになります。

メソッド／プロパティ	説明
allowDeliver WhileDocIsModal	モーダル状態にある文書でメッセージやエラーを送信するかどうかを示すブーリアン値。デフォルト (<code>false</code>) では、モーダルダイアログボックスがアプリケーションで表示されている場合、メッセージやエラーはキューに入れられ、 <code>onMessage</code> ハンドラや <code>onError</code> ハンドラには送信されません。モーダル状態であるかどうかは、 <code>app.isModal</code> プロパティで調べることができます。

注意：`onDisclose` では、メソッドを指定する代わりに、`HostContainerDisclosurePolicy.SameOriginPolicy` という値を指定できます。これを指定すると、ホストコンテナと文書が同じ場所から取得された場合に、文書が公開されます。同じ場所であるかどうかは、スキーム、サーバ、ポートによって判断されます。文書が公開されるためには、スキームが `http`、`https`、`ftp` のいずれかであることが必要です。

これらのメソッドを呼び出した場合、`this` オブジェクトは、そのメソッドを呼び出している `messageHandler` のインスタンスになります。

`messageHandler` には、`on` 以外の文字で始まるメソッドやプロパティを追加できます。

`messageHandler` に `null` を設定した場合や、`onMessage` メソッドが存在しないオブジェクトを設定した場合は、このプロパティに有効な `messageHandler` インスタンスが設定されるまで、`postMessage` で送信したメッセージはキューに入れられます。

メッセージはポストされた順番に送信され、エラーは発生した順番に送信されます。ただし、メッセージとエラーの送信順序に特定の関係はありません。

ハンドラメソッドで発生した例外は廃棄されます。`onDisclose` で例外が発生した場合は、関数が失敗したものとして処理されます。メッセージやエラーは、`onMessage`、`onError`、`onDisclose` の各ハンドラにある間は送信されません。

型

オブジェクト

アクセス

R / W

例

ホストコンテナの一般的なメッセージハンドラ。

```
this.hostContainer.messageHandler =
{
    onMessage: function(aMessage)
    {
        for(var i = 0; i < aMessage.length; i++)
            console.println("Recv'd Msg[ " + i + "] : " + aMessage[i]);
    },
    onError: function(error, aMessage){ },
    onDisclose: HostContainerDisclosurePolicy.SameOriginPolicy
};
```

HostContainer のメソッド

postMessage

7.0.5			
-------	--	--	--

PDF 文書のホストコンテナのメッセージハンドラに、非同期でメッセージを送信します。このメッセージを送信するには、`messageHandler` プロパティを設定して、ホストコンテナ（HTML ページの `<OBJECT>` 要素）に通知されるように登録しておく必要があります。このメッセージは、`messageHandler` の `onMessage` メソッドに渡されます。

メッセージは、送信されるまでキューに入れられます。キュー内のメッセージ数が最大数を超えた場合は、キュー内のいくつかのメッセージが送信されるまで、パラメータエラーが発生します。

パラメータ

aMessage messageHandler プロパティの `onMessage` メソッドに渡す 1 つ以上の文字列の配列。

例

```
var aMessage = ["MyMessageName", "Message Body"];  
this.hostContainer.postMessage(aMessage);
```

Icon

この汎用 JavaScript オブジェクトは、Doc の `icons` プロパティに保存されている Form XObject の外観を不透明に描写したものです。button タイプの Field オブジェクトで使用されます。`icon` オブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
<code>name</code>	文字列	R	アイコンの名前。文書レベルの名前付きアイコンツリーに含まれているアイコンは、名前を持ちます。

Icon Stream

この汎用 JavaScript オブジェクトは、アイコンストリームを表します。`app.addToolBar` や `collab.addStateModel` で使用されます。このオブジェクトのプロパティは、次のとおりです。

プロパティ	説明
<code>read(nBytes)</code>	読み取るバイト数を受け取って、16進エンコードされた文字列を返す関数。このデータは、1ピクセル 32ビットの ARGB（4つの8ビットチャンネルをチャンネルごとにインターリーブする方式）で表されたアイコンです。アイコンに複数のレイヤーがある場合は、最上位のレイヤーのピクセル、次のレイヤーのピクセル、という順序でピクセルを返します。
<code>width</code>	アイコンの幅（ピクセル単位）。
<code>height</code>	アイコンの高さ（ピクセル単位）。

`util.iconStreamFromIcon` メソッドを使用すれば、Icon オブジェクトを Icon Stream オブジェクトに変換できます。

identity

5.0			
-----	--	--	--

アプリケーションの現在のユーザを識別するための静的なオブジェクトです。

注意：identity オブジェクトのプロパティにアクセスできるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。これは、ユーザのプライバシーを保護するためです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

identity のプロパティ

corporation

環境設定の「ユーザ情報」に入力されている会社名。

型

文字列

アクセス

R / W

email

環境設定の「ユーザ情報」に入力されている電子メールアドレス。

型

文字列

アクセス

R / W

loginName

オペレーティングシステムに登録されているログイン名。

型

文字列

アクセス

R

name

環境設定の「ユーザ情報」に入力されているユーザの氏名（「姓」と「名」）。

型

文字列

アクセス

R / W

例

次の例は、コンソールまたはフォルダレベルの適切な JavaScript で実行できます。

```
console.println("Your name is " + identity.name);
console.println("Your e-mail is " + identity.email);
```

Index

5.0			
-----	--	--	--

Catalog によって生成されたインデックスを表すオブジェクト。このオブジェクトを作成することはできません。これは、search オブジェクトや catalog オブジェクトの様々なメソッドから取得します。このオブジェクトを使用して、Catalog による様々なインデックス処理を実行できます。インデックスのステータスは、search オブジェクトを使用して調べることができます。

Index のプロパティ

available

このインデックスを使用して選択や検索が行えるかどうかを示します。ネットワークがダウンしていたり、CD-ROM が挿入されていなかったり、インデックスの管理者がインデックスの保守作業をしているなどの理由で、インデックスを使用できないことがあります。

型

ブーリアン

アクセス

R

name

インデックス管理者がインデックス作成時に指定したインデックスの名前。

search.[indexes](#) を参照してください。このプロパティは、検索エンジンが現在アクセスしている Index オブジェクトの配列を返します。

型

文字列

アクセス

R

例

この例では、すべてのインデックスを列挙して、それらの名前をコンソールに書き込みます。

```
for (var i = 0; i < search.indexes.length; i++) {  
    console.println("Index[" + i + "] = " + search.indexes[i].name);  
}
```

path

デバイスに依存する、インデックスが存在するパス。パスの構文について詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

型

文字列

アクセス

R

selected

このインデックスを検索を利用するかどうかを示します。`true` の場合は、クエリの一部としてこのインデックスが検索されます。`false` の場合は検索されません。このプロパティを設定または解除することは、インデックスの選択ダイアログボックスでインデックスにチェックを付けるまたは解除することと同じです。

型

ブーリアン

アクセス

R / W

Index のメソッド

build

6.0			
-----	--	--	--

Catalog プラグインを使用して、この Index オブジェクトに対応するインデックスを作成します。このメソッドで新規インデックスの作成を行うことはできません。

インデックスは、インデックスファイルと同じ場所に作成されます。インデックスが既に存在する場合は、含めるディレクトリが再度スキャンされて変更が確認され、インデックスが更新されます。インデックスが存在しない場合は、ユーザインターフェイスを通じて新しいインデックスを定義および作成できます。

インデックスの作成は、Catalog がアイドル状態の場合はすぐに開始されます。それ以外の場合は、Catalog によってキューに入れられます。

注意：(Acrobat 7.0) このメソッドを実行できるのは、バッチャイ벤트または콘솔이벤트のみ입니다。詳しく는 [32 페이지의 「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)를 참조해주세요. JavaScript 이벤트에 대해서는 [event](#) 오브젝트를 참조해주세요.

パラメータ

cExpr	(オプション) インデックスの作成処理が完了した後に評価する式。デフォルトでは指定されていません。詳しくは、『PDF Reference』バージョン 1.7 を参照してください。
bRebuildAll	(オプション) <code>true</code> の場合は、完全な作り直しが行われます。インデックスは、最初に削除されてから作成されます。デフォルトは <code>false</code> です。

戻り値

`CatalogJob` オブジェクト。これを使用して、ジョブパラメータやステータスを確認することができます。

例

```
/* インデックスを作成 */
if (typeof catalog != "undefined") {
    var idx = catalog.getIndex("/c/mydocuments/index.pdx");
    var job = idx.build("Done()", true);
    console.println("Status : ", job.status);
}
```

Link

このオブジェクトを使用すれば、リンクプロパティの設定や取得を行ったり、リンクの JavaScript アクションを設定することができます。

Link オブジェクトは、Doc の addLink メソッドや getLinks メソッドから取得できます（[removeLinks](#) も参照してください）。

Link のプロパティ

borderColor

6.0	D		X
-----	----------	--	---

Link オブジェクトの境界線の色。カラー配列の定義や、このプロパティでの色の使用法については、[191 ページの「カラー配列」](#) を参照してください。

型

配列

アクセス

R / W

borderWidth

6.0	D		X
-----	----------	--	---

Link オブジェクトの境界線の幅。

型

整数

アクセス

R / W

highlightMode

6.0	D		X
-----	----------	--	---

リンクのアクティブな領域内でマウスボタンを押したときに使用される視覚効果。有効な値は、次のとおりです。

- None
- Invert (デフォルト)
- Outline
- Push

型

文字列

アクセス

R / W

rect

6.0	D		X
-----	---	--	---

ページ上のリンクの矩形を表す 4 つの数値の配列。これらの数値は、回転ユーザースペースにおける矩形の、左上隅の x 座標、左上隅の y 座標、右下隅の x 座標、右下隅の y 座標を表します。

型

配列

アクセス

R / W

Link のメソッド

setAction

6.0	D		X
-----	---	--	---

Link オブジェクトの MouseUp トリガに、指定の JavaScript アクションを設定します。

注意：このメソッドを実行すると、このリンクで既に定義されているアクションが上書きされます。

パラメータ

cScript 使用する JavaScript アクション。

Marker

Marker オブジェクトは、メディアクリップ内の特定の時間やフレーム番号を識別する、名前付きの場所です。これは、オーディオ CD のトラックや DVD のチャプタに似ています。マーカーの定義は、メディアクリップ自体によって行われます。

Marker オブジェクトは `Markers.get` メソッドから取得できます。

Marker のプロパティ

frame

6.0			
-----	--	--	--

フレーム番号。0 はメディアの先頭を表します。ほとんどのプレーヤーで、マーカーは `frame` または `time` のいずれかの値を持ちます。両方持つことはありません。

型

数値

アクセス

R

index

6.0			
-----	--	--	--

このマーカーに割り当てられているインデックス番号。マーカーには、0 から始まる連続したインデックス番号が付けられます。この番号は、メディア内でのマーカーの順序とは異なる場合があります。

型

数値

アクセス

R

name

6.0			
-----	--	--	--

このマーカーの名前。メディアクリップ内の各マーカーには、一意の名前が付けられます。

型

文字列

アクセス

R

例

マーカーをインデックスで取得し、そのマーカーの名前をコンソールに出力します。

```
// player は MediaPlayer オブジェクトと仮定
var markers = player.markers;
// インデックス 2 のマーカーを取得
var markers = g.get( { index: 2 } );
console.println( "The marker with index of " + markers.index
+ ", has a name of " + index.name );
```

time

6.0			
-----	--	--	--

時間（秒単位）。0 はメディアの先頭を表します。ほとんどのプレーヤーで、マーカーは frame または time のいずれかの値を持ちます。両方持つことはありません。

型

数値

アクセス

R

例

指定の名前のマーカーを取得し、メディアの先頭からそのマーカーまでの時間（秒単位）を出力します。

```
// player は MediaPlayer オブジェクトと仮定
var markers = player.markers;
// 「Chapter 1」というマーカーを取得
var markers = g.get( { name: "Chapter 1" } );
console.println( "The named marker ¥"Chapter 1¥", occurs at time "
+ markers.time );
```

Markers

MediaPlayer の `markers` プロパティは、現在プレーヤーに読み込まれているメディアクリップ内にあるすべてのマークターを表す `Markers` オブジェクトです。マークターは、メディアクリップ内の特定の時間やフレーム番号を識別する、名前付きの場所です。これは、オーディオ CD のトラックや DVD のチャプタに似ています。マークターの定義は、メディアクリップによって行われます。

コンストラクタは `app.media.Markers` です。

Markers のプロパティ

player

6.0			
-----	--	--	--

この `Markers` オブジェクトが属している `MediaPlayer` オブジェクト。

型

`MediaPlayer` オブジェクト

アクセス

R

Markers のメソッド

get

6.0			
-----	--	--	--

名前、インデックス番号、時間（秒単位）、フレーム番号などでマークターを検索し、そのマークターを表す `Marker` オブジェクトを返します。オブジェクトパラメータには、名前、インデックス、時間、フレームのいずれかのプロパティを含める必要があります。マークター名の文字列を直接渡すこともできます。

時間またはフレームを渡した場合は、それと同じ位置かその前にある、最も近いマークターが返されます。メディア内のすべてのマークターより前の時間またはフレームを渡した場合は、`null` が返されます。

パラメータ

マークターの名前、インデックス番号、時間（秒単位）またはフレーム番号を表す、オブジェクトまたは文字列。オブジェクトパラメータには、名前、インデックス、時間、フレームのいずれかのプロパティを含める必要があります。マークター名の文字列を直接渡すこともできます。

戻り値

Marker オブジェクトまたは null。

マーカーには、0 から始まる連続したインデックス番号が付けられます。この番号は、時間やフレームの順序とは異なる場合があります。また、Windows Media Player でのマーカー番号とも異なります。メディアクリップ内の有効なマーカーをすべて見つけるには、MediaPlayer.markers.get() を {index: 0} から始めてループで呼び出し、get() が null を返すまで番号をインクリメントします。

例 1

メディアクリップ上のマーカーの数をカウントします。

```
var index, i = 0;
// player は MediaPlayer オブジェクトと仮定
var m = player.markers;
while ( (index = m.get( { index: i } ) ) != null ) i++;
console.println("There are " + i + " markers.");
```

例 2

名前、インデックス、時間、フレームでマーカーを取得します。

```
// マーカーを名前で取得 (2 種類の方法)
var marker = player.markers.get( "My Marker" );
var marker = player.markers.get( { name: "My Marker" } );
// マーカーをインデックスで取得
var marker = player.markers.get( { index: 1 } );
// マーカーを時間で取得
var marker = player.markers.get( { time: 17.5 } );
// マーカーをフレームで取得
var marker = player.markers.get( { frame: 43 } );
```

MediaOffset

MediaOffset は、MediaClip 内の位置（時間またはフレームカウント）を表します。この位置は、絶対位置（メディアの先頭からの相対位置）で表すことも、名前付きマーカーからの相対位置で表すこともできます。

MediaOffset オブジェクトでは、次に示すプロパティを指定します。また、数値だけを指定することもできます。その場合は、`{time: number}` として解釈されます。

メディアフォーマットの中には、時間ベースのもの（QuickTime など）とフレームベースのもの（Flash など）があります。MediaOffset で時間またはフレームを指定するときは、使用するメディアフォーマットで対応しているものを指定する必要があります。時間とフレームを両方とも指定した場合は、定義が行われません。不正な指定を行うと、指定が無視されたり、JavaScript の例外が発生したりします。

MediaOffset オブジェクトは、`MediaPlayer.seek`、`MediaPlayer.where`、`MediaSettings.endAt`、`MediaSettings.startAt` で使用されます。

MediaOffset のプロパティ

frame

6.0			
-----	--	--	--

フレーム番号。`marker` プロパティも指定されている場合は、そのマーカーからの相対フレーム数（正、負またはゼロ）を表します。それ以外の場合は、メディアの先頭からの相対位置になります。負の値は指定できません。メディアの先頭は `{frame: 0}` で表されます。

型

数値

アクセス

R / W

marker

6.0			
-----	--	--	--

メディア内の特定のマーカーの名前。

型

文字列

アクセス

R / W

time

6.0			
-----	--	--	--

時間（秒単位）または `Infinity`（無限）。`marker` プロパティも指定されている場合は、そのマーカーから相対時間（負の値と `Infinity` は不可）を表します。それ以外の場合は、メディアの先頭からの相対時間になります。負の値は指定できません。メディアの先頭は、オフセット `{ time: 0 }` で表されます。

型

数値

アクセス

R / W

例

絶対オフセットと相対オフセットの例を示します。

```
{ time: 5.4 } // メディアの先頭から 5.4 秒のオフセット
{ marker: "Chapter 1", time: 17 } // 「Chapter 1」の 17 秒後
```

これらのオフセットは、`MediaPlayer.seek` メソッドで使用できます。

```
// player は MediaPlayer オブジェクトと仮定
player.seek({ time: 5.4 });
player.seek({ marker: "Chapter 1", time: 17 });
```

MediaPlayer

MediaPlayer オブジェクトは、QuickTime や Windows Media Player などの、マルチメディアプレーヤーのインスタンスを表します。このオブジェクトの `settings` プロパティや `events` プロパティを使用して、JavaScript コードでプレーヤーを操作したり、プレーヤーで発生したイベントを処理したりすることができます。MediaPlayer は、PDF ファイルの一部ではありません。これは、必要に応じてメモリ内に作成される一時オブジェクトです。

MediaPlayer のプロパティ

annot

6.0			
-----	--	--	--

MediaPlayer に関する付けられている画面注釈への参照。このプロパティは、画面注釈が関連付けられている MediaPlayer オブジェクトにのみ存在します。このプロパティは、`app.media.addStockEvents` によって設定されるか、または `addStockEvents` を間接的に呼び出すメソッド（`app.media.openPlayer` など）によって設定されます。

型

`ScreenAnnot` オブジェクト

アクセス

R / W

defaultSize

6.0			
-----	--	--	--

MediaPlayer の MediaClip の幅と高さを表す、読み取り専用のオブジェクト。

{ width: number, height: number }

メディアプレーヤーがこの値を提供できない場合、このプロパティは `undefined` になります。

型

オブジェクト

アクセス

R

doc

6.0			
-----	--	--	--

MediaPlayer を所有している Doc への参照。

型

オブジェクト

アクセス

R

events

6.0			
-----	--	--	--

MediaPlayer に登録されている EventListener を含む Events オブジェクト。詳しくは、[Events オブジェクト](#)を参照してください。

型

Events オブジェクト

アクセス

R / W

例

メディアプレーヤーを作成し、そのプレーヤーのイベントを変更します。このスクリプトは、関連するレンディションを持つレンディションアクションとして実行します。

```
var events = new app.media.Events;
var player = app.media.createPlayer();
player.events.add({
    onReady: function() { console.println("The player is ready"); }
});
player.open();
```

hasFocus

6.0			
-----	--	--	--

ブーリアン値。メディアプレーヤーが開いており、キーボードフォーカスを持っている場合は true。

型

ブーリアン

アクセス

R

id

6.0			
-----	--	--	--

このプレーヤーで使用されているプレーヤーソフトウェアのプレーヤー ID。プレーヤーが開いていない場合は、`undefined` になります。このプレーヤー ID は、このプレーヤーを実装しているメディアプレーヤーソフトウェアの `PlayerInfo.id` と同じ値です。

型

ブーリアン

アクセス

R

例

プレーヤー ID をコンソールに出力します。

```
// args が定義されているものと仮定
var player = app.media.openPlayer( args )
console.println("player.id = " + player.id);
// コンソールには、例えば次のように出力される
player.id = vnd.adobe.swname:ADBE_MCI
```

innerRect

6.0			
-----	--	--	--

プレーヤーの内側の矩形を表す配列。JavaScript の矩形を表す通常の配列と同じように、座標は [左, 上, 右, 下] の順になります。この矩形には、プレーヤーの周囲にある付属物（ウィンドウのタイトルなど）は含まれません。ただし、プレーヤーにコントローラが存在する場合は、そのコントローラも含まれます。プレーヤーが開いていない場合は、`undefined` になります。

ドッキングメディアプレーヤーの場合、この矩形はデバイススペースで表され、読み取り専用になります（設定しようとすると例外が発生します）。ドッキングプレーヤーのサイズを変更するには、`triggerGetRect` を使用します。フローティングメディアプレーヤーの場合、この矩形は画面座標で表され、書き込みが可能です。ただし、ユーザのセキュリティ設定によって、設定した値が上書きされることがあります。例えば、フローティングメディアプレーヤーを画面の外に移動しようとするとき、画面上に戻されることがあります。これによって例外が発生することはありません。このプロパティに書き込んだ後で読み取れば、値が上書きされたかどうかを確認できます。

[outerRect](#) も参照してください。

型

配列

アクセス

R / W

isOpen

6.0			
-----	--	--	--

ブーリアン値。メディアプレーヤーが現在開いている場合は `true`。プレーヤーを開いたり閉じたりするには、`MediaPlayer.open` や `MediaPlayer.close` を使用します。

型

ブーリアン

アクセス

R

isPlaying

6.0			
-----	--	--	--

ブーリアン値。メディアが現在再生中の場合は `true`。プレーヤーが開いていない場合や、メディアが一時停止、停止、早送り、巻き戻しなどの、その他の状態の場合は、`false`。

型

ブーリアン

アクセス

R

markers

6.0			
-----	--	--	--

現在のメディアで使用可能なすべてのマーカーの集合。

このプロパティについて詳しくは、[Markers](#) オブジェクトを参照してください。

型

Markers オブジェクト

アクセス

R

例

使用例については、[510 ページの「例 2」](#) を参照してください。

outerRect

6.0			
-----	--	--	--

プレーヤーの外側の矩形を表す配列。Acrobat JavaScript の矩形を表す通常の配列と同じように、座標は [左, 上, 右, 下] の順になります。この矩形には、プレーヤーの周囲にある付属物（プレーヤーのコントローラや、ウィンドウのタイトルなど）も含まれます。プレーヤーが開いていない場合は、`undefined` になります。

ドッキングメディアプレーヤーの場合、この矩形はデバイススペースで表され、読み取り専用になります。設定しようとすると例外が発生します。ドッキングプレーヤーのサイズを変更するには、`MediaPlayer.triggerGetRect` を使用します。フローティングメディアプレーヤーの場合、この矩形は画面座標で表され、書き込みが可能です。ただし、ユーザのセキュリティ設定によって、設定した値が上書きされることがあります。例えば、フローティングメディアプレーヤーを画面の外に移動しようとすると、画面上に戻されることがあります。これによって例外が発生することはありません。このプロパティに書き込んだ後で読み取れば、値が上書きされたかどうかを確認できます。

[innerRect](#) も参照してください。

型

配列

アクセス

R / W

page

6.0			
-----	--	--	--

ドッキングメディアプレーヤーが表示されるページ番号。ドッキングされていないプレーヤーの場合は `undefined` になります。ドッキングメディアプレーヤーは、`page` プロパティを変更することで、別のページに移動することができます。その場合は、`GetRect` イベント ([onGetRect](#) を参照) が発生します。

型

数値

アクセス

R / W

例

ページ 1 (0 から数えます) でメディアクリップを再生します。ページ 1 のメディアプレーヤーの配置は、ページ 0 の画面注釈と同じです。

```
var player = app.media.openPlayer({
    rendition: this.media.getRendition( "myClip" ),
    annot: this.media.getAnnot({ nPage:0, cAnnotTitle:"myScreen" }),
    settings: { windowType: app.media.windowType.docked }
});
player.page = 1;
```

この例のバリエーションについては、[onGetRect](#) および [triggerGetRect](#) を参照してください。

settings

6.0			
-----	--	--	--

MediaPlayer の作成に使用される設定がすべて含まれています。完全なリストについては、[MediaSettings](#) オブジェクトを参照してください。

注意：Acrobat 6.0 では、プレーヤーの作成後に MediaPlayer.settings のプロパティを変更しても、効果はありません。今後のリリースでは、これらの設定が即座に反映されるように変更される可能性があります。現在のリリースと今後のリリースで互換性を保つために、プレーヤーが開いているときに settings のプロパティを変更しないようにしてください。

型

MediaSettings オブジェクト

アクセス

R / W

uiSize

6.0			
-----	--	--	--

プレーヤーのコントローラのサイズ（プレーヤーの各エッジ）を表す配列。順序は、ウィンドウの矩形と同じ [左, 上, 右, 下] です。これらの値は、通常、正の値またはゼロです。これらの値には、ウィンドウの付属物（タイトルバーなど）は含まれません。

このプロパティは、Ready イベントが発生するまで使用できません ([onReady](#) および [afterReady](#) を参照)。ほとんどの MediaPlayer のプロパティとは異なり、on イベントメソッド (onReady など) で読み取ることができます。

型

配列

アクセス

R

例

プレーヤーの `uiSize` を取得します。このコードは、レンディションアクションイベントとして実行します。

```
var args = {
    events: {
        onReady: function () {
            console.println("uiSize = " + player.uiSize );
        }
    }
};
var player = app.media.openPlayer(args);
```

visible

6.0			
-----	--	--	--

プレーヤーが表示されているかどうかを制御するブーリアン値。`MediaPlayer.settings.visible` とは異なり、このプロパティは即座に反映されます。プレーヤーが開いていない場合、このプロパティを読み取ると `undefined` が返され、これを設定すると例外が発生します。

このプロパティを設定すると、イベントが発生することがあります。例えば、プレーヤーが表示されており、フォーカスを持っている場合、非表示にすると `Blur` イベントが発生します。

型

ブーリアン

アクセス

R / W

例

ビデオクリップのオーディオのみを再生します。

```
// args は定義済みと仮定
var player = app.media.openPlayer(args);
player.visible = false;
```

MediaPlayer のメソッド

close

6.0			
-----	--	--	--

メディアプレーヤーが開いている場合は、閉じます。プレーヤーが閉じている場合は、何も行いません（エラーにはなりません）。

eReason パラメータには、app.media.closeReason の列挙値を指定します。この値は、close メソッドで発生する Close イベント ([onClose](#) および [afterClose](#) を参照) の event.media.closeReason プロパティに渡されます。

プレーヤーがキーボードフォーカスを持っている場合は、Close イベントの前に Blur イベント (onBlur / afterBlur) が発生します。メディアプレーヤーによっては、Status (onStatus / afterStatus) や Stop (onStop / afterStop) などの、他のイベントが発生することもあります。

パラメータ

eReason eReason は、app.media.closeReason の列挙値です。

open

6.0			
-----	--	--	--

MediaPlayer.settings の指定に従ってメディアプレーヤーを開こうとします。プレーヤーが既に開いている場合は、例外が発生します。以前にプレーヤーが開かれて閉じられた場合は、open を呼び出すことで、プレーヤーを再度開くことができます。この場合は、以前と同じ JavaScript オブジェクトが使用されますが、メディアプレーヤーは新しいインスタンスが開かれます。この新しいプレーヤーでは、古いプレーヤーの再生位置などは保持されません。

ドッキングプレーヤーの場合は、プレーヤーが開かれると GetRect イベント (onGetRect) が発生します。

MediaPlayer.settings.autoPlay が true (デフォルト) の場合は、再生が開始し、Play イベント (onPlay / afterPlay) が発生します。

ユーザの設定によっては、open メソッドにより、セキュリティプロンプトダイアログボックスが表示されることがあります。また、他のメディアプレーヤーや画面注釈などのオブジェクトでイベントが発生することもあります。例えば、別のメディアプレーヤーにキーボードフォーカスがある場合は、そのプレーヤーで Blur イベント (onBlur / afterBlur) が発生します。

bAllowSecurityUI が false の場合は、open メソッドによってセキュリティプロンプトが表示されることはありませんが、障害コードが返されます。

フローティングウィンドウのメディアプレーヤーの場合は、ユーザの設定に基づいて追加のセキュリティチェックが行われます。例えば、フローティングプレーヤーのすべてのウィンドウでタイトルバーを表示するように、ユーザが指定している場合があります。ユーザが許可していないオプションが MediaPlayer.settings.floating に含まれている場合の処理は、bAllowFloatOptionsFallback によって決まります。これが false の場合は、再生が許可されず、エラーコードが返されます。これが true の場合は、ユーザのセキュリティ設定に合うように、MediaPlayer.settings.floating のオプションが適宜変更されます。open は、この変更された設定を使用して実行されます。

戻り値は单一のオブジェクトです。現在のところ、このオブジェクトには `code` という 1 つのプロパティが含まれます。これは、`app.media.openCode` で列挙されている結果コードです。今後のバージョンの Acrobat では、このオブジェクトにプロパティが追加されたり、コード値が追加されたりする可能性があります。そのような値も適切に処理できるようにしてください。

パラメータ

<code>bAllowSecurityUI</code>	(オプション) デフォルトは <code>true</code> です。このパラメータに関する前述の説明を参照してください。
<code>bAllowFloatOptionsFallback</code>	(オプション) デフォルトは <code>true</code> です。このパラメータに関する前述の説明を参照してください。

戻り値

`code` プロパティを持つオブジェクト

例

使用例については、[163 ページの「例 1」](#) を参照してください。

pause

6.0			
-----	--	--	--

現在のメディアの再生を一時停止し、Pause イベント (`onPause` / `afterPause`) を発生させます。Pause イベントは、`pause` の呼び出し中またはその後に発生します（プレーヤーによって異なります）。

メディアが一時停止または停止されている場合や、再生が開始されていないか完了済みの場合は、`pause` メソッドを実行しても何も行われません。メディアプレーヤーやメディアフォーマットの中には、`pause` をサポートしていないものもあります。特に、ほとんどのストリーミングフォーマットでは `pause` をサポートしていません。その場合は、プレーヤーで例外が発生するか、`pause` が無視されます。

例

使用例については、[510 ページの「例 2」](#) を参照してください。

play

6.0			
-----	--	--	--

現在のメディアの再生を開始し、Play イベント (`onPlay` / `afterPlay`) を発生させます。Play イベントは、`play` の呼び出し中またはその後に発生します（プレーヤーによって異なります）。

メディアが既に再生中の場合は、再生が続行され、イベントは発生しません。一時停止、巻き戻しまたは早送り中の場合は、現在の位置から再生が再開されます。メディアの先頭または最後で停止している場合は、最初から再生が開始されます。

例

使用例については、[510 ページの「例 2」](#) を参照してください。

seek

6.0			
-----	--	--	--

現在のメディアの再生位置を、`oMediaOffset` パラメータの `MediaOffset` オブジェクトで指定した位置に設定します。

メディアが再生中の場合は、新しい位置から再生が続行されます。メディアが一時停止している場合は、新しい位置に移動して、そこで一時停止します。メディアが停止している場合の結果は、プレーヤーによって異なります。

シークエラーの処理は、メディアプレーヤーによって異なります。エラーが無視される場合もあれば、JavaScript 例外が発生する場合もあります。

多くのメディアプレーヤーでは、シークの完了時に `Seek` イベント (`onSeek` / `afterSeek`) が発生します。

シーク処理は、`seek` メソッドの実行中またはその後で行われます（プレーヤーによって異なります）。シーク処理の完了前に `seek` が戻り、シークの完了前に別のプレーヤーメソッドを呼び出した場合の結果は、プレーヤーによって異なります。

パラメータ

`oMediaOffset`

`MediaOffset` オブジェクト。このプロパティで、設定する再生位置を指定します。

例 1

```
// メディアクリップを巻き戻す
player.seek({ time: 0 });

// 「First」というマーカーから再生を開始
player.seek({ marker: "First" });

// 「One」というマーカーの 5 秒後から再生を開始
player.seek({ marker: "One", time: 5 });
```

例 2

次のスクリプトは、指定された名前の引用句をランダムに再生します。このメディアは、マーカーとスクリプトがサポートされているオーディオクリップ (.wma) であり、指定された名前の引用句が収録されています。`afterReady` リスナは、各引用句の先頭にあるマーカーの数をカウントしています。引用句の末尾には、コマンドスクリプトが埋め込まれています。`afterScript` リスナはそれらのコマンドを監視し、「`pause`」コマンドである場合はプレーヤーを一時停止します。

```
var nMarkers=0;
var events = new app.media.Events;
events.add({
```

```
// このオーディオクリップ内の引用句の数をカウントし、nMarkers として保存
afterReady: function()
{
    var g = player.markers;
    while ( (index = g.get( { index: nMarkers } ) ) != null )
        nMarkers++;
},
// 各引用句の末尾にはスクリプトがある。pause コマンドである場合は
// プレーヤーを一時停止する。
afterScript: function( e ) {
    if ( e.media.command == "pause" ) player.pause();
}
);
var player = app.media.openPlayer({
rendition: this.media.getRendition( "myQuotes" ),
settings: { autoPlay: false },
events: events
});
// 引用句をランダムに選択
function randomQuote() {
    var randomMarker, randomMarkerName;
    console.println("nMarkers = " + nMarkers);
    // 1 ~ nMarkers までの整数をランダムに選択
    randomMarker = Math.floor(Math.random() * 100) % ( nMarkers ) + 1;
    // どの引用句を再生しているかを示す
    this.getField("Quote").value = "Playing quote " + randomMarker;
    // マーカーの名前は「quote 1」、「quote 2」、「quote 3」など
    randomMarkerName = "quote " + randomMarker;
    // randomMarkerName で指定される名前のマーカーを表示
    player.seek( { marker: randomMarkerName } );
    player.play();
}
```

アクションは、次のような「マウスボタンを放す」アクションで開始します。

```
try { randomQuote() } catch(e) {}
```

setFocus

6.0			
-----	--	--	--

キーボードフォーカスをメディアプレーヤーに設定し、Focus イベント (onFocus / afterFocus) を発生させます。別のプレーヤーや PDF オブジェクトにフォーカスがある場合は、そのオブジェクトで Blur イベント (onBlur / afterBlur) が発生します。メディアプレーヤーに既にフォーカスがある場合は、何も行われません。プレーヤーが開いていないか表示されていない場合は、例外が発生します。

例

使用例については、[163 ページの「例 1」](#) を参照してください。

stop

6.0			
-----	--	--	--

現在のメディアの再生を停止し（再生中または一時停止中の場合）、Stop イベント（onStop / afterStop）を発生させます。Stop イベントは、stop メソッドの実行中またはその後に発生します（プレーヤーによって異なります）。メディアが再生中でも一時停止中でもない場合は、何も行われません。

プレーヤーが開いていない場合は、例外が発生します。

再生を停止すると、メディアの先頭または最後に再生位置が設定されます（プレーヤーによって異なります）。この後に MediaPlayer.play が呼び出された場合、再生はメディアの先頭から開始します。

triggerGetRect

6.0			
-----	--	--	--

GetRect イベント（[onGetRect](#) を参照）を発生させて、ドッキングメディアプレーヤーのサイズを変更します。

例

この例は、onGetRect の例に似ています。ページ 0 には一連の（サムネールサイズの）ScreenAnnot があります。これは、標準的なレンディションアクションまたは「マウスボタンを放す」JavaScript アクションです。アクションが実行されると、メディアアクリップのサイズが変更され、再生されます。

```
var rendition = this.media.getRendition("Clip1");
var annot = this.media.getAnnot({ nPage:0,cAnnotTitle:"ScreenClip1" });
var player = app.media.openPlayer({
    rendition: rendition,
    annot: annot,
    settings: { windowType: app.media.windowType.docked },
    events: {
        onGetRect: function (e) {
            var width = e.media.rect[2] - e.media.rect[0];
            var height = e.media.rect[3] - e.media.rect[1];
            width *= 3; // 幅と高さを 3 倍にする
            height *= 3;
            e.media.rect[0] = 36; // 左端と上端を
            e.media.rect[1] = 36; // 左上隅へ移動
            e.media.rect[2] = e.media.rect[0]+width;
            e.media.rect[3] = e.media.rect[1]+height;
            return e.media.rect; // これを返す
        }
    }
});
player.triggerGetRect(); // onGetRec イベントを発生させる
```

where

6.0			
-----	--	--	--

MediaOffset オブジェクトの現在のメディアの再生位置を報告します。このオブジェクトには、time プロパティまたは frame プロパティのいずれかが含まれています（メディアプレーヤーやメディアタイプによって異なります）。

プレーヤーが開いていないか、プレーヤーが where をサポートしていない場合は、例外が発生します。

戻り値

MediaOffset オブジェクト

例

再生位置（秒単位）を取得します。

```
// このコードでは、time を返す where() がプレーヤーでサポートされているものと仮定。  
var where = player.where();  
var seconds = where.time;
```

チャプタ（マーカー）を取得。

```
var marker = player.markers.get({ time: seconds });  
var name = marker ? marker.name : "no marker";
```

MediaReject

MediaReject は、Rendition.select の呼び出しで拒否されたレンディションに関する情報を提供します。これには、元のレンディションへの参照と、拒否された理由が含まれています。select によって返される MediaSelection オブジェクトで、MediaSelection.rejects を実行すれば、MediaReject オブジェクトの配列を取得できます。

MediaReject のプロパティ

rendition

6.0			
-----	--	--	--

select の呼び出しで拒否されたレンディションへの参照。

型

Rendition オブジェクト

アクセス

R

例

拒否されたレンディションのリストを取得します。このスクリプトは、レンディションアクションとして実行します。

```
selection = event.action.rendition.select(true);
for ( var i=0; i<selection.rejects.length; i++)
    console.println("Rejected Renditions: "
        + selection.rejects[i].rendition.uiName);

// 最初に使用可能なレンディションを再生
console.println( "Preparing to play " + selection.rendition.uiName);
var settings = selection.rendition.getPlaySettings();
var args = {
    rendition: selection.rendition,
    annot: this.media.getAnnot({ nPage: 0, cAnnotTitle: "myScreen" }),
    settings: settings
};
app.media.openPlayer(args);
```

MediaSelection

Rendition.select メソッドは、MediaSelection オブジェクトを返します。このオブジェクトを使用して、再生用の MediaSettings オブジェクトを作成できます。

MediaSelection のプロパティ

selectContext

6.0			
-----	--	--	--

この値を使用すれば、ループの中で Rendition.select を繰り返し呼び出すことができます。取得した選択を独自のテストコードで検査していけば、目的に合った選択を抽出できます。

型

オブジェクト

アクセス

R

例

selectContext を使用した一般的なスクリプト

```
function MyTestSelection( selection )
{
    // この関数で、選択に対して独自のテストを行う。採用する場合は
    // true を返す。却下して次の選択をテストする場合は false を返す。
}
function MyGetSelection( rendition )
{
    var selection;
    for( selection = rendition.select(); selection;
        selection = rendition.select
            ({ oContext: selection.selectContext })) {
        if( MyTestSelection( selection ) )
            break;
    }
    return selection;
}
```

players

6.0			
-----	--	--	--

MediaSelection.rendition の再生に使用可能なメディアプレーヤーを識別する文字列の配列。再生可能なレンディションが見つからない場合、players プロパティと rendition プロパティは null になります。

型

文字列の配列

アクセス

R

例

選択されたレンディションを再生できるプレーヤーのリストを取得します。次のコードは、レンディションアクションとして実行します。

```
var selection = event.action.rendition.select();
for ( var o in selection.players )
    console.println( selection.players[o].id );
```

rejects

6.0			
-----	--	--	--

MediaReject オブジェクトの配列。これらのオブジェクトは、この MediaSelection を返した Rendition.select の呼び出しで拒否されたレンディションです。詳しくは、[MediaReject](#) オブジェクトを参照してください。

型

MediaReject オブジェクトの配列

アクセス

R

例

[514 ページの「例」](#) を参照してください。

rendition

6.0			
-----	--	--	--

選択されたレンディション。再生可能なものが無い場合は `null`。

型

`Rendition` オブジェクト

アクセス

R

例

選択されたレンディションの名前を取得します。このスクリプトは、レンディションアクションイベントから実行します。

```
var selection = event.action.rendition.select();
console.println( "Preparing to play " + selection.rendition.uiName);
```

MediaSettings

MediaSettings オブジェクトには、MediaPlayer を作成して開くために必要な設定が含まれています。このオブジェクトは、MediaPlayer オブジェクトの settings プロパティの値です。これらの設定の多くにはデフォルト値がありますが、開いているプレーヤーのタイプや他の設定によっては、設定が必要なものもあります。詳しくは、MediaSettings の各プロパティの説明を参照してください。

Acrobat や各種のメディアプレーヤーは、これらの設定を使用しようとしますが、すべての設定が使用されるとは限りません（例えば、palindrome 設定を使用するプレーヤーはほとんどありません）。

MediaSettings のプロパティ

autoPlay

6.0			
-----	--	--	--

プレーヤーが開かれたら、メディアクリップの再生を自動的に開始するかどうかを指定します。autoPlay を `false` に設定した場合は、`MediaPlayer.play` を使用して再生を開始します。デフォルト値は `true` です。

型

ブーリアン

アクセス

R / W

例

[afterReady](#) および [players](#) の例も参照してください。

baseURL

6.0			
-----	--	--	--

メディアクリップで使用される相対 URL を解決するためのベース URL（Web ページを開く場合などに使用されます）。デフォルト値はありません。baseURL を指定しなかった場合の相対 URL の解釈は、メディアプレーヤーによって異なります。ほとんどの場合は機能しません。

型

文字列

アクセス

R / W

bgColor

6.0			
-----	--	--	--

メディアプレーヤーウィンドウの背景色。この配列には、Acrobat JavaScript でサポートされている任意のカラー配列形式を使用できます。

bgColor を指定しなかった場合のデフォルト値は、ウィンドウのタイプによって異なります。

ドッキング — 白

フローティング — オペレーティングシステムのコントロールパネルで指定されているウィンドウの背景色

フルスクリーン — Acrobat のユーザ環境設定で指定されているフルスクリーンモードの背景色

型

カラー配列

アクセス

R / W

例

```
// 赤の背景
settings.bgColor = [ "RGB", 1, 0, 0 ];
```

bgOpacity

6.0			
-----	--	--	--

メディアプレーヤーウィンドウの背景の不透明度。有効な値の範囲は、0.0（完全に透明）～1.0（完全に不透明）です。デフォルト値は 1.0 です。

型

数値

アクセス

R / W

data

6.0			
-----	--	--	--

MediaData オブジェクトとも呼ばれます。メディアプレーヤーは、このオブジェクトを使用して、(外部または PDF 埋め込みの) メディアクリップデータを読み取ることができます。このオブジェクトのコンテンツを JavaScript から直接使用することはできません。

このオブジェクトは、`app.media.getAltTextData` や `app.media.getURLData` から取得するか、`Rendition.getPlaySettings` から間接的に取得します。data オブジェクトは、レンディションの文書にバインドされている場合があるので、文書が閉じていると使用できないことがあります。

型

オブジェクト

アクセス

R / W

例

`app.media.getURLData` の例を参照してください。

duration

6.0			
-----	--	--	--

再生する時間 (秒単位)。指定しなかった場合のデフォルト値は、メディア全体の長さか、`startAt` ポイントと `endAt` ポイントの間隔 (いずれかのポイントが指定されている場合) になります。

`duration` には、メディア全体の長さや、`startAt` ポイントと `endAt` ポイントの間隔より長い時間を指定することもできます。その場合は、メディアの最後または `endAt` ポイントまで再生した後、指定の時間が経過するまでその位置で一時停止します。

型

数値

アクセス

R / W

例

フローティングウィンドウ内のメディアを無限に再生します。レンディションの UI 上での再生場所は、フローティングウィンドウです。次のコードは、フォームボタンから実行します。プレーヤーがメディアの最後に達しても、フローティングウィンドウは開いたままになります。フローティングウィンドウが重ならないように、プレーヤーを閉じてから再度開いています。

このスクリプトをレンディションアクションから実行する場合は、UI を介してレンディションを指定できるので、プレーヤーを閉じる必要はありません。

```
var rendition = this.media.getRendition("Clip");
if ( player && player.isOpen )
    try { player.close(app.media.closeReason.done); } catch(e) {};
var player = app.media.openPlayer({
    rendition: rendition,
    settings: { duration: Infinity }
});
```

endAt

6.0			
-----	--	--	--

再生を終了する時間またはフレーム。この値は、絶対時間、絶対フレーム、マーカー名、マーカーと時間、マーカーとフレームのいずれかになります（MediaOffset オブジェクトを参照）。再生は、指定の時間またはフレームで終了するか、停止可能な位置のうち指定のポイントに最も近い位置で終了します。endAt を指定しなかった場合のデフォルト値は、メディアの最後です。

[startAt](#) も参照してください。

型

MediaOffset オブジェクト

アクセス

R / W

例

次のスクリプトでは、メディアの 3 秒から 8 秒の位置までオーディオクリップを再生します。

```
var player = app.media.openPlayer({
    rendition: this.media.getRendition( "myAudio" ),
    doc: this,
    settings: {
        startAt: 3,
        endAt: 8
    }
});
```

floating

6.0			
-----	--	--	--

フローティングウィンドウの場所とスタイルを定義するプロパティ（次の表を参照）を含むオブジェクト。

`MediaSettings.windowType` に `app.media.windowType.floating` という値が指定されていない場合、このオブジェクトは無視されます。

指定しなかった `floating` 設定については、すべてデフォルトが使用されます。

プロパティ	型	説明
<code>align</code>	数値	<code>over</code> プロパティで指定したウィンドウに対する、フローティング ウィンドウの相対的な位置を指定します。 <code>align</code> の値は、 <code>app.media.align</code> の値の 1 つです。
<code>over</code>	数値	どのウィンドウでフローティングウィンドウを整列させるかを指定します。 <code>over</code> の値は、 <code>app.media.over</code> の値の 1 つです。
<code>canResize</code>	数値	フローティングウィンドウのサイズをユーザが変更できるかどうかを指定します。 <code>canResize</code> の値は、 <code>app.media.canResize</code> の値の 1 つです。
<code>hasClose</code>	ブーリアン	<code>true</code> の場合、フローティングウィンドウにウィンドウを閉じるコントロールボタンが表示されます。
<code>hasTitle</code>	ブーリアン	<code>true</code> の場合、タイトルバーにタイトルが表示されます。
<code>title</code>	文字列	<code>hasTitle</code> が <code>true</code> の場合、このタイトルが表示されます。
<code>ifOffScreen</code>	数値	フローティングウィンドウの全体または一部が画面の外に出た場合のアクションを指定します。 <code>ifOffScreen</code> の値は、 <code>app.media.ifOffScreen</code> の値の 1 つです。
<code>rect</code>	4 つの数値の配列	フローティングウィンドウの場所とサイズを表す、画面座標の配列。 <code>width</code> と <code>height</code> を指定しない場合は必須です。
<code>width</code>	数値	フローティングウィンドウの幅。 <code>rect</code> を指定しない場合は必須です。
<code>height</code>	数値	フローティングウィンドウの高さ。 <code>rect</code> を指定しない場合は必須です。

型

オブジェクト

アクセス

R / W

例

フローティングウィンドウ内でメディアクリップを再生します。

```
var rendition = this.media.getRendition( "myClip" );
var floating = {
    align: app.media.align.topCenter,
    over: app.media.over.appWindow,
    canResize: app.media.canResize.no,
    hasClose: true,
    hasTitle: true,
    title: rendition.altText,
    ifOffScreen: app.media.ifOffScreen.forceOnScreen,
    width: 400,
    height: 300
};
var player = app.media.openPlayer({
    rendition: rendition,
    settings: {
        windowType: app.media.windowType.floating,
        floating: floating
    }
});
```

layout

6.0			
-----	--	--	--

ウィンドウに合わせてコンテンツのサイズを変更するかどうかと、その方法を定義します。この値は、`app.media.layout` の列挙値の 1 つです。デフォルト値は、メディアプレーヤーによって異なります。

型

数値

アクセス

R / W

monitor

6.0			
-----	--	--	--

フルスクリーンモードで再生する場合に使用するディスプレイモニタを指定します。これは、`Monitor` オブジェクトか `Monitors` オブジェクトです。配列の場合は、最初の要素（`Monitor` オブジェクト）が使用されます。

型

Monitor オブジェクトまたは Monitors オブジェクト

注意：再生に使用される `rect` プロパティは、`MediaSettings.monitor.rect` (Monitor オブジェクトの場合) または `MediaSettings.monitors[0].rect` (Monitors オブジェクトの場合) のみです。

`monitor` プロパティと `monitorType` プロパティの関係については、次の [monitorType](#) の説明を参照してください。

アクセス

R / W

例

フォームボタンからメディアクリップをフルスクリーンモードで再生します。

```
var player = app.media.openPlayer({
    rendition: this.media.getRendition("Clip"),
    settings: {
        monitor: app.monitors.primary(),
        windowType: app.media.windowType.fullScreen,
    }
});
```

注意：環境設定の「マルチメディアの信頼性」分類にある「信頼性管理オプション」の設定で、フルスクリーンモードでの再生が許可されている必要があります。

monitorType

6.0			
-----	--	--	--

フローティングウィンドウやフルスクリーンウィンドウでの再生に使用するモニタのタイプを表します。これは、`app.media.monitorType` の値の 1 つです。

`monitor` プロパティと `monitorType` プロパティの違いは、次のとおりです。

- `monitor` は、矩形を定義して、現在のシステム上の特定のモニタを指定します。
- `monitorType` は、プライマリ、セカンダリ、最適な色深度などの属性に基づいて、モニタの一般的カテゴリを指定します。

JavaScript を使用しない PDF ファイルでは、特定のモニタを指定することはできませんが、モニタのタイプは指定できます。`monitorType` を指定して `app.media.createPlayer` や `app.media.openPlayer` を呼び出す場合は、システムで使用可能なモニタのリストを取得し、`monitorType` を使用して再生用のモニタを 1 つ選択します。このモニタの矩形は、`MediaPlayer.open` が呼び出されてモニタが選択されたときに使用されます。

型

数値

アクセス

R / W

例

最適な色深度のモニタを使用して、フルスクリーンモードでメディアクリップを再生します。

```
var player = app.media.openPlayer({
    rendition: this.media.getRendition("Clip"),
    settings: {
        monitorType: app.media.monitorType.bestColor,
        windowType: app.media.windowType.fullScreen,
    }
});
```

page

6.0			
-----	--	--	--

ドッキングメディアプレーヤーをドッキングするページの番号。その他のタイプのメディアプレーヤーの場合、このプロパティは無視されます。

[MediaPlayer.page](#) も参照してください。

型

数値

アクセス

R / W

palindrome

6.0			
-----	--	--	--

このプロパティが `true` の場合は、メディアが通常どおり 1 回再生された後、先頭に向かって逆向きに再生されます。`repeat` が指定されている場合、この正再生／逆再生は何回も繰り返されます。正再生／逆再生のペアが完了するたびに、1 回の繰り返しとしてカウントされます。

デフォルト値は `false` です。

型

ブーリアン

アクセス

R / W

注意：ほとんどのメディアプレーヤーでは、正逆再生がサポートされておらず、この設定は無視されます。

例

正逆再生をサポートしている QuickTime を使用してメディアクリップを再生します。

```
var playerList = app.media.getPlayers().select({ id: '/quicktime/i' });
var settings = { players: playerList, palindrome: true };
var player = app.media.openPlayer({ settings: settings });
```

このコードは、関連するレンディションを持つレンディションアクションイベントで実行します。

players

6.0			
-----	--	--	--

このレンディションの再生に使用できるメディアプレーヤーを表すオブジェクトの配列。通常、JavaScript コードでこの配列に直接アクセスすることはできません。この配列は、Rendition.select から app.media.createPlayer の settings オブジェクトに渡します。

型

プレーヤーまたは文字列の配列

アクセス

R / W

例

このレンディションの再生に使用可能なプレーヤーをリストします。このスクリプトは、関連するレンディションを持つレンディションアクションとして実行します。

```
var player = app.media.openPlayer({ settings: {autoPlay: false} });
console.println("players: " + player.settings.players.toSource() );

// コンソールへの出力例
players: [{id:"vnd.adobe.swname:ADBE_MCI", rank:0},
{id:"vnd.adobe.swname:AAPL_QuickTime", rank:0},
{id:"vnd.adobe.swname:RNWK_RealPlayer", rank:0},
{id:"vnd.adobe.swname:MSFT_WindowsMediaPlayer", rank:0}]
```

rate

6.0			
-----	--	--	--

再生速度を指定する数値。デフォルト値は 1 です。これは通常の再生を意味します。それ以外の値は、通常の速度を基準とした速度です。例えば、.5 は半分の速度、2 は 2 倍の速度、-1 は逆方向で通常の速度です。

多くのプレーヤーやメディアタイプでは、サポートされている速度の値が制限されており、サポートされている再生速度に最も近い値が選択されます。

型

数値

アクセス

R / W

例

メディアクリップを 2 倍速で再生します。このスクリプトは、レンディションアクションとして実行します。

```
var player = app.media.createPlayer();
player.settings.rate = 2;
player.open();
```

repeat

6.0			
-----	--	--	--

メディアの再生を自動的に繰り返す回数。デフォルト値の 1 では、メディアは 1 回再生されます。

多くのプレーヤーでは、整数値のみがサポートされていますが、非整数値（1.5 など）がサポートされているものもあります。Infinity という値を設定すると、メディアクリップは無限に再生し続けます。

デフォルト値は 1 です。

型

数値

アクセス

R / W

例

レンディションアクションから、メディアクリップを無限に再生し続けます。

```
var player = app.media.openPlayer({settings: {repeat: Infinity}});
```

showUI

6.0			
-----	--	--	--

メディアプレーヤーのコントロールを表示するか非表示にするかを指定するブーリアン値。

デフォルト値は `false` です。

型

ブーリアン

アクセス

R / W

例

メディアプレーヤーのコントロールを表示します。このスクリプトは、レンディションアクションとして実行します。

```
var player = app.media.createPlayer();
player.settings.showUI = true;
player.open();
```

または

```
app.media.openPlayer( {settings: {showUI: true}} );
```

startAt

6.0			
-----	--	--	--

再生を開始する時間またはフレーム。この値は、絶対時間、絶対フレーム、マーカー名、マーカーと時間、マーカーとフレームのいずれかになります（`MediaOffset` を参照）。再生は、指定の時間またはフレームから開始するか、停止可能な位置のうち指定のポイントに最も近い位置から開始します。`startAt` を指定しなかつた場合のデフォルト値は、メディアの先頭です。

[endAt](#) も参照してください。

型

`MediaOffset` オブジェクト

アクセス

R / W

例

[endAt](#) の例を参照してください。

visible

6.0			
-----	--	--	--

プレーヤーを表示するかどうかを指定するブーリアン値。

デフォルト値は `true` です。

型

ブーリアン

アクセス

R / W

例

ドッキングされたメディアクリップで、オーディオのみが再生されるようにします。このスクリプトは、レンディションアクションとして実行します。

```
var args = {
    settings: {
        visible: false,
        windowType: app.media.windowType.docked
    }
};
app.media.openPlayer( args );
```

`MediaPlayer.visible` も参照してください。

volume

6.0			
-----	--	--	--

再生音量を指定します。値 0 はミュート、値 100 は通常の（フル）音量、その間の値は中間の音量です。将来のメディアプレーヤーでは、通常の音量より大きいことを意味する 100 を超える値を使用できるようになる可能性がありますが、現在は使用できません。

デフォルト値は 100 です。

型

数値

アクセス

R / W

windowType

6.0			
-----	--	--	--

MediaPlayer を作成するウィンドウのタイプを定義します。この値は、app.media.windowType の列挙値の 1 つです。

低レベル関数である doc.media.newPlayer を使用する場合、windowType のデフォルト値は app.media.windowType.docked です。

app.media オブジェクトの高レベル関数である createPlayer や openPlayer を使用する場合、デフォルト値は次のようにになります。

- `annot` を指定した場合 ([PlayerArgs オブジェクト](#) の説明を参照)、デフォルトは `app.media.windowType.docked` です。
- `settings.floating` オブジェクトを指定した場合 ([PlayerArgs オブジェクト](#) の説明を参照)、デフォルトは `app.media.windowType.floating` です。
- それ以外の場合、デフォルトは未定義です。

型

数値

アクセス

R / W

例

様々なタイプのウィンドウでメディアプレーヤーを作成します。このスクリプトはレンディションアクションとして実行するので、レンディションの仕様を選択する必要はありません。

```
// 関連付けられている ScreenAnnot で再生される、ドッキングプレーヤー
app.media.openPlayer({
    settings: { windowType: app.media.windowType.docked }
});
// フルスクリーンモードで再生。monitor および monitorType も参照
app.media.openPlayer({
    settings: { windowType: app.media.windowType.fullScreen }
});
// フローティングウィンドウでメディアクリップを表示。floating プロパティも参照
var args = {
    settings: {
        windowType: app.media.windowType.floating,
        floating: {
            title: "A. C. Robat",
            width: 352,
            height: 240,
        }
    }
};
app.media.openPlayer( args );
```

Monitor

Monitor オブジェクトは、個々のディスプレイモニタを表します。Monitor オブジェクトは、システムに接続されているすべてのモニタの配列を返す `app.monitors` から取得できます。`app.monitors` は Monitors オブジェクトなので、Monitors オブジェクトのメソッドを使用して、様々な基準に基づいてマルチモニタシステムからモニタを選択したりフィルタしたりすることができます。詳しくは、[Monitors オブジェクト](#)を参照してください。

Monitor オブジェクトや Monitors オブジェクトは、`MediaSettings` の `monitor` プロパティで使用されます。

Monitor のプロパティ

colorDepth

6.0			
-----	--	--	--

モニタの色深度（1ピクセル当たりのビット数）。

型

数値

アクセス

R

例

プライマリモニタを取得し、その色深度を確認します。プライマリモニタの選択には、`Monitors.primary` メソッドを使用します。

```
var monitors = app.monitors.primary();
console.println( "Color depth of primary monitor is "
+ monitors[0].colorDepth );
```

isPrimary

6.0			
-----	--	--	--

ブーリアン値。プライマリモニタの場合は `true`、それ以外のモニタの場合は `false`。

型

ブーリアン

アクセス

R

例

最大幅のモニタを取得し、それがプライマリモニタであるかどうか確認します。

```
var monitors = app.monitors.widest();
var isIsNot = (monitors[0].isPrimary) ? "is" : "is not";
console.println("The widest monitor "+isIsNot+" the primary monitor.");
```

rect

6.0			
-----	--	--	--

仮想デスクトップ座標における、モニタの境界を表す矩形。

- 仮想デスクトップの原点は、プライマリモニタの左上隅にあります。したがって、プライマリモニタの境界は常に [0, 0, 右, 下] の形式になります。
- セカンダリモニタの境界配列には、プライマリモニタからの相対位置に応じて、正の値や負の値が含まれことがあります。

型

矩形

アクセス

R

workRect

6.0			
-----	--	--	--

仮想デスクトップ座標における、モニタのワークスペースの境界を表す矩形。この座標について詳しくは、[rect](#) を参照してください。

ワークスペースとは、アプリケーションで通常使用されるモニタ領域のことです。ドッキングされたツールバーや、タスクバーなどのアイテムは含まれません。例えば、Windows を単一の 800×600 ディスプレイで実行している場合、rect は [0, 0, 800, 600] になります。標準の Windows タスクバーを画面の下部に 30 ピクセルの高さで常に表示している場合、workRect は [0, 0, 800, 570] になります。

型

矩形

アクセス

R

Monitors

Monitors オブジェクトは、単一のモニタを表す Monitor オブジェクトの配列です。これは読み取り専用です。各要素には、通常の配列表記や、`length` プロパティを使用してアクセスできます。

`app.monitors` プロパティは、ユーザのシステムに接続されている各モニタを含む Monitors オブジェクトを返します。JavaScript コードでこの配列をループして、使用可能なモニタに関する情報を取得し、フルスクринやポップアップのメディアプレーヤーを表示するモニタを選択できます。

注意： `app.monitors` によって返される Monitors オブジェクトは未ソートです（モニタがリストされる順序は定義されていません）。

Monitors オブジェクトのフィルタメソッドを使用すれば、様々な基準に基づいて 1 つ以上のモニタを選択できます。PDF ファイルフォーマットで提供されているモニタ選択オプションはすべて、これらのフィルタメソッドとして実装されています。

どのフィルタメソッドを実行しても、元の Monitors オブジェクトは変更されません。これらのメソッドは、通常、1 つ以上の Monitor オブジェクトを含む新しい Monitors オブジェクトを返します。1 つのモニタのみがフィルタ基準を満たしている場合、返される Monitors オブジェクトにはそのモニタが含まれます。複数のモニタがフィルタ基準を同等に満たしている場合（例えば、`bestColor` メソッドで、最大の色深度を持つモニタが複数ある場合）、返されるオブジェクトにはそれらのモニタがすべて含まれます。

一部のフィルタメソッドには、最低要件を指定するオプションパラメータがあります。このパラメータが指定されており、最低要件を満たすモニタが存在しない場合は、空の Monitors オブジェクトを返します。それ以外の場合は、元の Monitors オブジェクトが空でない限り、常に 1 つ以上のモニタを含むオブジェクトを返します。

フィルタメソッドでの高さ、幅、領域の指定は、常にピクセル単位で行います。

例

```
var monitors = app.monitors;
for ( var i = 0; i < monitors.length; i++)
    console.println("monitors["+i+"] .colorDepth = "+monitors[i].colorDepth);
```

`Monitors.length` には、Monitors オブジェクトの要素の数が含まれています。`app.monitors` で返される Monitors オブジェクトの場合、これはユーザのシステムにあるモニタの数を表します。フィルタメソッドで返される Monitors オブジェクトの場合は、それより少なくなる可能性があります。

Monitors のメソッド

bestColor

6.0			
-----	--	--	--

最大の色深度を持つモニタを含む、Monitors オブジェクトのコピーを返します。

`nMinColor` が指定されており、最大の色深度が `nMinColor` よりも小さい場合は、空の Monitors 配列を返します。

パラメータ

nMinColor (オプション) モニタに必要な最小の色深度。

戻り値

Monitors オブジェクト

例

```
var monitors = app.monitors.bestColor(32);
if (monitors.length == 0 )
    console.println("Cannot find the required monitor.");
else
    console.println("Found at least one monitor.");
```

bestFit

6.0			
-----	--	--	--

少なくとも指定の nWidth と nHeight (ピクセル単位) を持つ最小のモニタのみを含む、Monitors オブジェクトのコピーを返します。

パラメータ

nWidth モニタの最小限の幅

nHeight モニタの最小限の高さ

bRequire (オプション) 指定の最小幅および高さを満たすモニタが存在しない場合の処理を指定します。true の場合、このメソッドは空の Monitors 配列を返します。false または省略されている場合は、最大のモニタを含む Monitors 配列を返します。

戻り値

Monitors オブジェクト

desktop

6.0			
-----	--	--	--

仮想デスクトップ全体を表す单一の Monitor オブジェクトを含む、新しい Monitors オブジェクトを作成します。rect プロパティは、元の Monitors オブジェクトの各 rect の和集合です。workRect プロパティは、元の Monitors オブジェクトの各 workRect の和集合です。colorDepth は、元の Monitors オブジェクトの最小の colorDepth 値です。

戻り値

Monitors オブジェクト

注意： desktop メソッドは、通常、app.monitors で返される Monitors オブジェクトで直接呼び出します。フィルタメソッドによってフィルタ済みの Monitors オブジェクトで desktop メソッドを実行した場合は、モニタのそのサブセットに対して、前述と同じ計算が行われます。

document

6.0			
-----	--	--	--

Doc の doc パラメータで指定されている文書が最も広く表示されているモニタを含む、Monitors オブジェクトのコピーを返します。

元の Monitors オブジェクトにリストされているどのモニタにもその文書が表示されていない場合は、bRequire が true であれば空の Monitors 配列を返します。bRequire が false であるか省略されていれば、元の配列から任意に選択したモニタを 1 つ以上含む Monitors 配列を返します。

パラメータ

doc 文書の Doc。

bRequire (オプション) ブーリアン値。前述の説明を参照してください。

戻り値

Monitors オブジェクト

filter

6.0			
-----	--	--	--

リスト内の各モニタをランキング関数でフィルタした結果を含む、Monitors オブジェクトのコピーを返します。ランキング関数は Monitor パラメータを取り、数値ランクを返します。filter からの戻り値は、ランクが最も高いモニタ（単一のモニタか、同ランクが存在する場合は複数のモニタ）を含む Monitors 配列です。

パラメータ

fnRanker Monitor パラメータを取り、数値ランクを返す（ランキング）関数。

nMinRank (オプション) nMinRank が未定義の場合は、常に元のリストのモニタを 1 つ以上返します（元のリストが空でない場合）。nMinRank が指定されており、ランキング関数の戻り値が nMinRank 以上であるモニタが存在しない場合は、空の Monitors 配列を返します。

戻り値

Monitors オブジェクト

注意： Monitors の他のフィルタ関数の多くは、`filter` の呼び出しとして実装されています。

例

このスクリプトは、`Monitors.bestColor(minColor)` の実装です。最大の色深度のモニタを含む Monitors オブジェクトを返します。`minColor` が指定されており、最大の色深度が `minColor` よりも小さい場合は、空の Monitors 配列を返します。

```
bestColor: function( minColor )
{
    return this.filter(
        function( m ) { return m.colorDepth; }, minColor );
}
```

largest

6.0			
-----	--	--	--

最大の領域（ピクセル単位）を持つモニタを含む、Monitors オブジェクトのコピーを返します。

パラメータ

`nMinArea` (オプション) オプションのパラメータ `nMinArea` (数値) が指定されており、最大の領域がその値よりも小さい場合、`largest()` は空の Monitors 配列を返します。

戻り値

Monitors オブジェクト

leastOverlap

6.0			
-----	--	--	--

`rect` パラメータで指定した矩形と重なる範囲が最も狭いモニタを含む、Monitors オブジェクトのコピーを返します。

パラメータ

`rect` 画面座標における、矩形を表す 4 つの数値の配列。

`maxOverlapArea` (オプション) `maxOverlapArea` が指定されている場合は、矩形と重なる範囲がそのパラメータで指定した領域以上であるモニタのみが Monitors 配列に含まれます。そのようなモニタが存在しない場合は、空の Monitors 配列を返します。

戻り値

Monitors オブジェクト

mostOverlap

6.0			
-----	--	--	--

rect パラメータで指定した矩形と重なる範囲が最も広いモニタを含む、Monitors オブジェクトのコピーを返します。

minOverlapArea が指定されており、矩形と重なる範囲がそのパラメータで指定した領域以上であるモニタが存在しない場合は、空の Monitors 配列を返します。minOverlapArea が省略されている場合は、元の配列から任意に選択したモニタを 1 つ以上含む Monitors 配列を返します。

パラメータ

rect 画面座標における、矩形を表す 4 つの数値の配列。

minOverlapArea (オプション) ブーリアン値。前述の説明を参照してください。

戻り値

Monitors オブジェクト

nonDocument

6.0			
-----	--	--	--

対象文書がまったく表示されていないモニタか、対象文書が最も狭く表示されているモニタを含む、Monitors オブジェクトのコピーを返します。

パラメータ

doc 対象文書の Doc。

bRequire (オプション) bRequire は、対象文書がまったく表示されていないモニタが存在しない場合の処理を指定するブーリアン値です。true の場合、nonDocument は空の Monitors 配列を返します。false または省略されている場合、nonDocument は元の Monitors 配列から任意に選択したモニタを 1 つ以上含む Monitors 配列を返します。

戻り値

Monitors オブジェクト

primary

6.0			
-----	--	--	--

セカンダリモニタがすべて除外された、プライマリモニタのみを含む（元のリストに存在していた場合）`Monitors` オブジェクトのコピーを返します。

プライマリモニタが元のリストに存在していなかった場合は、元のリストから任意に選択したモニタを 1 つ以上含む `Monitors` 配列を返します。

戻り値

`Monitors` オブジェクト

例

プライマリモニタを取得し、その色深度を確認します。

```
var monitors = app.monitors.primary();
// Monitors オブジェクトの各要素は Monitor オブジェクト
// このコードでは monitor.colorDepth を使用する
console.println( "Color depth of primary monitor is "
+ monitors[0].colorDepth );
```

secondary

6.0			
-----	--	--	--

プライマリモニタを除外した、セカンダリモニタのみを含む `Monitors` オブジェクトのコピーを返します。

元の `Monitors` オブジェクトにプライマリモニタのみが含まれ、セカンダリモニタが含まれていない場合は、元のリストを返します。

戻り値

`Monitors` オブジェクト

select

6.0			
-----	--	--	--

`nMonitor` でフィルタした結果を含む、`Monitors` オブジェクトのコピーを返します。`nMonitor` は、PDF で使用されるモニタ選択値であり、`app.media.monitorType` の列挙値の 1 つです。

`nMonitor` が `app.media.monitorType.document` または `app.media.monitorType.nonDocument` である場合、`doc` は必須です。`nMonitor` が他の値である場合、`doc` は無視されます。

これらの選択値は、`Monitors` の各フィルタメソッドに直接対応します。`select` では、対応するフィルタメソッドを呼び出して複数のモニタが返されたときには、ほとんどの場合、`primary` による再フィルタが行われます。

パラメータ

nMonitor	モニタのタイプ。 <code>app.media.monitorType</code> の列挙値の 1 つ。
doc	Doc。このパラメータは、 <code>nMonitor</code> が <code>app.media.monitorType.document</code> または <code>app.media.monitorType.nonDocument</code> の場合は必須であり、それ以外の場合は無視されます。

戻り値

`Monitors` オブジェクト。

例

次のどちらの方法でも、同じモニタがフィルタされます。

```
settings.monitor =  
  app.monitors().select( app.media.monitorType.document, doc );  
settings.monitor = app.monitors().document(doc).primary();
```

tallest

6.0			
-----	--	--	--

最大の高さ（ピクセル単位）を持つモニタを含む、`Monitors` オブジェクトのコピーを返します。

パラメータ

nMinHeight	（オプション） <code>nMinHeight</code> が指定されており、それ以上の高さを持つモニタが存在しない場合は、空の <code>Monitors</code> 配列を返します。
------------	---

戻り値

`Monitors` オブジェクト

widest

6.0			
-----	--	--	--

最大の幅（ピクセル単位）を持つモニタを含む、`Monitors` オブジェクトのコピーを返します。

パラメータ

nMinWidth	(オプション) nMinWidth が指定されており、それ以上の幅を持つモニタが存在しない場合は、空の Monitors 配列を返します。
-----------	---

戻り値

Monitors オブジェクト

Net

Net オブジェクトを使用すれば、様々なプロトコルでネットワークサービスを検出してアクセスできます。

ネットワークサービス（SOAP）はグローバル名前空間で定義されていましたが、これは Net オブジェクトの名前空間にまとめられました。従来のオブジェクトは、下位互換性のために残されていますが、新しいサービスでは新しい名前空間を使用してください。

Net のプロパティ

SOAP

8.0			
-----	--	--	--

Net.SOAP オブジェクトを使用すれば、SOAP メッセージングプロトコルを使用したネットワークサービスと通信が行えます。Net.SOAP オブジェクトには、1 つのプロパティと 3 つのメソッドがあります。詳しくは、参照先のページを参照してください。

名前	簡単な説明	ページ
Net.SOAP のプロパティ		
wireDump	true の場合は、同期 SOAP リクエストを行うと、XML のリクエストやレスポンスが JavaScript コンソールに表示されます。これは、SOAP の問題をデバッグする場合に役立ちます。	654 ページ
Net.SOAP のメソッド		
connect	WSDL 文書の URL を、Web サービスから呼び出し可能なメソッドを持つ JavaScript オブジェクトに変換します。	654 ページ
request	SOAP HTTP エンドポイントに対してリモートプロシージャ呼び出し (RPC) を開始するか、XML メッセージを送信します。このメソッドは、エンドポイントの応答を待つ (同期処理) か、通知オブジェクトのメソッドを呼び出します (非同期処理)。	661 ページ
response	応答を待たずに、SOAP HTTP エンドポイントに対してリモートプロシージャ呼び出し (RPC) を開始するか、XML メッセージを送信します。	669 ページ

型

オブジェクト

アクセス

R

Discovery

8.0			
-----	--	--	--

Discovery オブジェクトを使用すれば、ネットワークサービスを動的に検出できます。Net.Discovery オブジェクトには、2つのメソッドがあります。詳しくは、参照先のページを参照してください。

Net.Discovery のメソッド	簡単な説明	ページ
queryServices	自分自身をパブリッシュしたネットワークサービスを、DNS Service Discovery (DNS-SD) を使用して見つけます。	657 ページ
resolveService	接続を確立するために、ネットワークアドレスおよびポートにサービス名をバインドできます。	659 ページ

型

オブジェクト

アクセス

R

HTTP

8.0			
-----	--	--	--

Net.HTTP オブジェクトを使用すれば、HTTP を使用した Web サービスにアクセスできます。このオブジェクトでは、Net.SOAP オブジェクトよりも多くのサービスがサポートされています。WebDAV や ATOM 出版プロトコルなどの、HTTP を使用したあらゆるメッセージングプロトコルは、HTTP オブジェクトを使用して処理することができます。

HTTP オブジェクトのメソッドについて詳しくは、[545 ページ](#)の [Net.HTTP](#) を参照してください。

型

オブジェクト

アクセス

R

Net のメソッド

Net オブジェクトには、3 つのメソッドがあります。詳しくは、参照先のページを参照してください。

Net のメソッド	簡単な説明	ページ
streamDecode	指定のエンコードタイプでストリームオブジェクトをデコードします。	670 ページ
streamDigest	指定のエンコードタイプでストリームオブジェクトのダイジェストを生成します。	671 ページ
streamEncode	ストリームオブジェクトをエンコードします。	671 ページ

Net.HTTP

Net.HTTP オブジェクトを使用すれば、HTTP を使用した Web サービスにアクセスできます。このオブジェクトでは、Net.SOAP オブジェクトよりも多くのサービスがサポートされています。WebDAV や ATOM 出版プロトコルなどの、HTTP を使用したあらゆるメッセージングプロトコルは、HTTP オブジェクトを使用して処理することができます。

Net.HTTP のメソッド

request

8.0			
-----	--	--	--

request メソッドは、HTTP Web サービスを非同期で呼び出します。このメソッドでは、Net.SOAP.request (SOAP.request) メソッドよりも多くの Web サービスがサポートされており、HTTP のメソッド、ヘッダ、本文を詳細に制御できます。WebDAV や Atom 出版プロトコルなどの複雑なネットワークプロトコルも、このメソッドで作成できます。

注意：このメソッドは、文書のコンテキストの外部（フォルダレベルの JavaScript など）でのみ実行できます。

パラメータ

cVerb 使用する HTTP の動詞を指定する文字列。動詞は、次のいずれかである必要があります。

メッセージングプロトコル	パラメータの値
HTTP (RFC 1945、2616)	GET、POST、PUT、DELETE、OPTIONS、HEAD
WebDAV (RFC 2518)	PROPFIND、PROPPATCH、MKCOL、LOCK、UNLOCK、COPY、MOVE、ACL、SEARCH
WebDAV Versioning Extensions (RFC 3253)	CHECKIN、CHECKOUT、UNCHECKOUT、MERGE、VERSION-CONTROL、REPORT、KWORKSPACE、UPDATE、LABEL、MERGE、MKACTIVITY
CalDAV Calendaring Extensions	MKCALENDAR

cURL 接続する Web サービスのネットワークエンドポイントの場所を指定する文字列。URL のスキームは、HTTP または HTTPS であることが必要です。

aHeaders (オプション) リクエストとともに送信する HTTP ヘッダを指定する配列。一部のヘッダ (Content-length など) は、明示的に設定することはできません。配列内の各要素は、ヘッダの情報を記述するオブジェクトで、次のプロパティがあります。

プロパティ	説明
name	使用するヘッダの名前。有効な HTTP ヘッダ名であることが必要です。
value	HTTP ヘッダの値。有効な HTTP ヘッダの値を表す文字列である必要があります。

oRequest (オプション) メソッドのリクエストの本文を指定したストリームオブジェクト。メッセージ本文のエンコーディングは、送信前には実行されません。

oHandler (オプション) サービスからのレスポンスを処理するオブジェクト。`oHandler` オブジェクトでは、次のパラメータを取る `response` メソッドを定義します。

パラメータ	説明
<code>oRequest</code>	レスポンスの本文を表すストリームオブジェクト（成功した場合）。
<code>cURL</code>	このリクエストの呼び出しに使用されたリクエスト URL。
<code>oException</code>	例外オブジェクト（メソッドが失敗した場合）。この例外オブジェクトのプロパティは、次のとおりです。 <code>error</code> — 返された HTTP ステータスコード。 <code>msg</code> — 失敗に関連するエラーメッセージ。
<code>aHeaders</code>	サーバから返されたレスポンスヘッダの配列。この形式については、 <code>Net.HTTP.request</code> メソッドの <code>aHeaders</code> パラメータを参照してください。

`oAuthenticate` (オプション) HTTP 認証の処理方法や使用する証明書を指定するオブジェクト。デフォルトでは、BASIC 認証モードと DIGEST 認証モードの HTTP 認証を処理するため、ユーザインターフェイスが表示されます。`oAuthenticate` オブジェクトに設定できるプロパティは、次のとおりです。

プロパティ	説明
<code>Username</code>	認証に使用するユーザ名を含む文字列。
<code>Password</code>	認証に使用する資格証明を含む文字列。
<code>UsePlatformAuth</code>	プラットフォーム認証を使用することを示すブーリアン値。プラットフォーム認証が有効な場合は、 <code>Username</code> と <code>Password</code> が無視されて、基盤となるプラットフォームのネットワークコードが使用されます。この場合、認証 UI がユーザに表示されるか、または現在ログインしているユーザの資格証明が使用されます（両方行われることもあります）。デフォルトは <code>false</code> です。サポートされているのは Windows プラットフォームのみです。

例外

URL が無効であるか、文書の送信元と同じでない場合は、例外が発生します。

例

指定の URL にコレクションを作成します。

```
CreateWebDAVCollection = function(cURL)
{
    var params =
    {
        cVerb: "MKCOL",
        cURL: cURL,
        oHandler:
        {
            response: function(msg, uri, e)
            {
                // HTTP 405 - 既に存在するコレクションを作成しようとした
                if(e != undefined && e.error != 405) {
                    app.alert("Failed to MKCOL: "+ e);
                } else app.alert("Created collection");
            }
        }
    };
    Net.HTTP.request(params);
} // CreateWebDAVCollection
```

OCG

OCG オブジェクトは、PDF ファイル内のオプショナルコンテンツグループ（Optional Content Group）を表します。ファイル内のコンテンツは、1 つ以上のオプショナルコンテンツグループに関連付けることができます。1 つ以上の OCG に属するコンテンツをオプショナルコンテンツと呼びます。このコンテンツが表示されるかどうかは、属している OCG の状態（オンまたはオフ）によって決まります。例えば、1 つの OCG に属しているオプショナルコンテンツは、その OCG がオンの場合には表示され、オフの場合には非表示になります。複数の OCG や様々な表示マッピングを使用すれば、より高度に表示を制御することができます。

PDF 文書の OCG オブジェクトの配列を取得するには、Doc の `getOCGs` メソッドを使用します。

Doc の `addWatermarkFromFile` メソッドや `addWatermarkFromText` メソッドは、OCG に透かしを追加します。

オプショナルコンテンツグループについて詳しくは、『PDF Reference』バージョン 1.7 を参照してください。

OCG のプロパティ

constants

7.0			
-----	--	--	--

OCG オブジェクトの各インスタンスは、このプロパティを継承します。このプロパティは、様々な定数値を保持するためのラッパー オブジェクトです。

intents オブジェクト

OCG のインテント配列には、任意の文字列を含めることができます。ただし、Acrobat で認識されるのは次の文字列のみです。

プロパティ	説明
<code>design</code>	OCG オブジェクトのデザインインテントを指定します。
<code>view</code>	OCG オブジェクトのビューインテントを指定します。

states オブジェクト

`states` オブジェクトは、OCG の初期状態を設定するために使用されます（[`initState`](#) を参照）。

プロパティ	説明
<code>on</code>	OCG のオン状態を指定します。
<code>off</code>	OCG のオフ状態を指定します。

initState

7.0	D		
-----	----------	--	--

このプロパティは、OCG をデフォルトでオンにするかオフにするかを示します。有効な値については、[states オブジェクト](#)を参照してください。

型

ブーリアン

アクセス

R / W (Adobe Reader : R のみ)

例

OCG の初期状態をオフに設定します。

```
var ocgs = this.getOCGs();  
ocgs[0].initState.constants.states.off;
```

locked

7.0	D		
-----	----------	--	--

このプロパティは、OCG をロックするかどうかを示します。OCG がロックされている場合、そのオン／オフ状態を UI を通じて切り替えることはできません。

型

ブーリアン

アクセス

R / W (Adobe Reader : R のみ)

name

6.0	D		
-----	----------	--	--

UI に表示するテキスト文字列。これを使用して OCG を識別することは可能ですが、一意の文字列とは限らない点に注意してください。

注意： name は、Acrobat 6.0 では読み取り専用ですが、Acrobat 7.0 では読み取り／書き込みが可能です。

型

文字列

アクセス

R / W (Adobe Reader : R のみ)

例

すべての透かし OCG を切り替える関数。

```
function ToggleWatermark(doc)
{
    var ocgArray = doc.getOCGs();
    for (var i=0; i < ocgArray.length; i++) {
        if (ocgArray[i].name == "Watermark") {
            ocgArray[i].state = !ocgArray[i].state;
        }
    }
}
```

state

6.0			
-----	--	--	--

OCG の現在のオン／オフ状態を表します。

OCG の状態を変更しても、文書にその情報が追加されたりしません。文書を再度読み込むと、OCG は初期状態に戻ります。

型

ブーリアン

アクセス

R / W

例

指定した文書内のすべての OCG をオンにします。

```
function TurnOnOCGsForDoc(doc)
{
    var ocgArray = doc.getOCGs();
    for (var i=0; i < ocgArray.length; i++)
        ocgArray[i].state = true;
}
```

OCG のメソッド

getIntent

7.0			
-----	--	--	--

この OCG のインテント配列を返します。

OCG がコンテンツの表示／非表示に影響を与えるのは、この配列に `constants.intents.view` というインテントが含まれている場合のみです。

[setIntent](#) および [intents オブジェクト](#) も参照してください。

戻り値

文字列の配列。戻り値の種類については、「`constants.intents`」を参照してください。

setAction

6.0			X
-----	--	--	---

OCG の状態が変化するたびに評価する JavaScript 式を登録します。

OCG のアクションを設定しても、文書にその情報が追加されたりしません。したがって、アクションの割り当ては、文書が初めて開かれるときに、文書レベルのスクリプトで実行するか、ユーザインターフェイスを通じて動的に実行する必要があります。

注意：このメソッドを実行すると、この OCG で既に定義されているアクションが上書きされます。

パラメータ

cExpr	OCG の状態が変化した後に評価する式。
-------	----------------------

例

OCG にビープ音を割り当てる関数を定義してから、OCG の配列を `getOCGs` メソッドで取得し、最初の OCG にビープ音のアクションを割り当てます。

```
/* 指定の OCG が変更されたらビープ音を鳴らす */
function BeepOnChange(ocg)
{
    ocg.setAction("app.beep() ");
}
var ocgs = this.getOCGs();
BeepOnChange(ocgs[0]);
```

setIntent

7.0	D		X
-----	---	--	---

この OCG のインテント配列を設定します。OCG がコンテンツの表示／非表示に影響を与えるのは、この配列に `constants.intents.view` が含まれている場合のみです。有効な値については、[intents オブジェクト](#) を参照してください。

[getIntent](#) および [intents オブジェクト](#) も参照してください。

パラメータ

aIntentArray OCG のインテント配列として使用する文字列の配列。

例

文書内のすべての OCG のインテントを、ビューとデザインの両方に設定します。

```
var ocgs = this.getOCGs();
for (i=0; i < ocgs.length; i++) {
    ocgs[i].setIntent( [ocgs[i].constants.intents.view,
    ocgs[i].constants.intents.design] );
}
```

PlayerInfo

PlayerInfo オブジェクトは、メディアの再生に使用可能なメディアプレーヤーを表します。
app.media.getPlayers メソッドは、PlayerInfo オブジェクトのコレクションである PlayerInfoList オブジェクトを返します。

PlayerInfo のプロパティ

id

6.0			
-----	--	--	--

メディアプレーヤープラグインや、関連付けられているメディアプレーヤーを表します。この文字列はローカライズされていないので、ユーザへの表示には適していません。この文字列は、MediaPlayer オブジェクトのインスタンスを生成するときに MediaPlayer.settings.players 配列で使用できます。また、プレーヤーを開いた後は MediaPlayer.id プロパティでアクセスできます。

型

文字列

アクセス

R

例

「video/mpeg」を再生するすべてのメディアプレーヤーのプレーヤー情報をリストします。

```
var playerInfoList = app.media.getPlayers("video/mpeg");

for ( var i=0; i < playerInfoList.length; i++) {
    console.println("id: " + playerInfoList[i].id)
    console.println("name: " + playerInfoList[i].name)
    console.println("version: " + playerInfoList[i].version)
}
```

mimeTypes

6.0			
-----	--	--	--

このメディアプレーヤーでサポートされている MIME タイプを表す文字列の配列。

型

文字列の配列

アクセス

R

例

```
var qtinfo = app.media.getPlayers().select({id: /quicktime/i })[0];  
console.println( qtinfo.mimeTypes );
```

name

6.0			
-----	--	--	--

メディアプレーヤーの名前。この文字列は、app.language の現在の言語に従ってローカライズされています。リストボックスなどでの表示には適していますが、JavaScript コードで直接比較するのには適していません。

型

文字列

アクセス

R

version

6.0			
-----	--	--	--

メディアプレーヤーのバージョン番号を含む文字列。多くの場合、ユーザのシステムにインストールされているメインのメディアプレーヤーのバージョン番号を表します。この文字列は、ドット区切りの 10 進数です (7.4.030.1170 など)。

型

文字列

アクセス

R

PlayerInfo のメソッド

canPlay

6.0			
-----	--	--	--

メディアプレーヤーを再生に使用できるかどうかを確認します。このメソッドでは、ユーザのセキュリティ設定を考慮するかどうかを指定できます。

`bRejectPlayerPrompt` パラメータを `true` にすると、プレーヤーの使用時にセキュリティプロンプトが表示される場合は `false` が返されます。このパラメータを `false` にすると、セキュリティプロンプトが表示されるかどうかは考慮されず、再生が許可される場合は `true` が返されます（このメソッドを実行してもセキュリティプロンプトは表示されません。このプロンプトは、メディアプレーヤーのインスタンスを生成するときに表示されます）。

パラメータ

oDoc	Doc
<code>bRejectPlayerPrompt</code>	ブーリアン値。デフォルトは <code>false</code> です。 <code>true</code> にすると、プレーヤーの使用時にセキュリティプロンプトが表示される場合は <code>false</code> が返されます。 <code>false</code> にすると、セキュリティプロンプトが表示されるかどうかは考慮されず、再生が許可される場合は <code>true</code> が返されます

戻り値

ブーリアン

canUseData

6.0			
-----	--	--	--

`oData` パラメータで指定されたデータをプレーヤーが再生に使用できるかどうかを確認します。データを再生に使用できる場合は `true` を、それ以外の場合は `false` を返します。

パラメータ

<code>oData</code>	MediaData オブジェクト（このオブジェクトについては、 <code>MediaSettings.data</code> を参照）。このオブジェクトは、 <code>app.media.getAltTextData</code> や <code>app.media.getURLData</code> から取得するか、 <code>Rendition.getPlaySettings</code> から間接的に取得します。
--------------------	--

戻り値

ブーリアン

honors

7.0			
-----	--	--	--

args パラメータにリストされているすべての設定、メソッド、イベントに対応しているかどうかを、プレーヤープラグインに問い合わせます。結果は正確でないこともあります、メディアプレーヤーを開かずには取得できる情報としては最善のものになります。例えば、args.URL を渡した場合、スキーム (<http://>など) は確認されますが、URL を実際に開けるかどうかは確認されません。

注意：互換性：honors がサポートされているのは、Acrobat 7.0 以降のみです。Acrobat SDK には、Acrobat 6.0 と Acrobat 7.0 の両方で実行できる JavaScript ソースコードが用意されており、PDF にコピーして利用できます。このコードを Acrobat 6.0 で実行した場合は、そのコードで実装されているアルゴリズムでメディアプレーヤーの機能が検証されます。Acrobat 7.0 以降で実行した場合は、honors が呼び出されます。詳しくは、[playerHonors 関数](#)を参照してください。

honors と HonorsArgs オブジェクト ([557 ページ](#)を参照) は、PDF の MH (「Must Honor」) エントリに似ています。このエントリの一部は、マルチメディアレンディションのレンディション設定ダイアログボックの「再生要件」タブで設定できます。honors メソッドを使用すると、再生要件を満たすプレーヤーを PDF ファイルで静的に選択するのではなく、JavaScript から動的に選択できます。

パラメータ

args	テストする HonorsArgs オブジェクト。HonorsArgs オブジェクトは、app.media.openPlayer メソッドのパラメータとして使用する PlayerArgs オブジェクトによく似ています。実際、PlayerArgs オブジェクトは HonorsArgs として使用できます。HonorsArgs には、honors のみで使用できる追加のオプションが用意されています。
------	--

戻り値

ブーリアン値。プレーヤープラグインが args オブジェクトのすべての内容に対応している場合は true。

例

特定の機能をサポートしているプレーヤーでメディアクリップを再生します。

```
function playWithRequirements( args )
{
    var plugins = app.media.getPlayers( args.mimeType )
    if( plugins )
    {
        for (var plugin in plugins)
        {
            if( plugin.honors(args) )
            {
                args.players = [plugin];
                return app.media.openPlayer( args );
            }
        }
    }
}
```

指定の URL にある AVI ファイルを再生します。使用できるプレーヤーは、`autoPlay` をオフにでき、`pause` メソッドがサポートされており、`marker` と `time` でオフセットを指定できる `seek` メソッドと `startAt` 設定がサポートされており、`Ready` イベントと `Close` イベントがサポートされているプレーヤーです。

```
playWithRequirements({
  mimeType: 'video/avi',
  URL: 'http://www.example.com/bar.avi',
  settings: {
    {
      autoPlay: false,
      startAt: { marker: 'test', time: 1 },
    },
  methods: {
    {
      pause: [],
      seek[ { marker: 'test', time: 1 } ],
    },
  events:
  {
    afterReady: doAfterReady( e ),
    onClose: doOnClose( e ),
  },
}),
});
```

HonorsArgs オブジェクト

`HonorsArgs` オブジェクトは、`PlayerInfo` オブジェクトの `honors` メソッドや `playerHonors` 関数で使用する設定、メソッド、イベントをリストしたオブジェクトです。ここでは `PlayerInfo.honors` を使用して両方を説明します。

`PlayersArgs` オブジェクト (`app.media.openPlayer` で使用するオブジェクト) が既に存在している場合は、それを `HonorsArgs` オブジェクトとして使用することができます。また、`PlayerInfo.honors` 専用の `HonorsArgs` オブジェクトを作成することもできます。

同じオブジェクトを使用しても、`app.media.openPlayer` と `PlayerInfo.honors` で不明な `args` の処理が異なることに注意してください。`app.media.openPlayer` では、不明な設定やイベントは無視されますが、`PlayerInfo.honors` では、認識できない設定、メソッド、イベントがあると `false` が返されます。

例えば、`app.media.openPlayer` では、`{ settings: { upsideDown: true } }` というオブジェクトを使用することができます。「`upsideDown`」という設定は存在しませんが、この関数では無視されます。一方、`PlayerInfo.honors` でのこのオブジェクトを使用すると、不明な設定が含まれているので `false` が返されます。

`HonorsArgs` オブジェクトで指定できるすべてのプロパティを次に示します。これは JavaScript のオブジェクトリテラルに似ていますが、プロパティ値を指定する場所には、各プロパティのタイプまたは説明が示されています。

```
args =
{
  mimeType: string,
  URL: string,
  settings:
  {
    autoPlay: boolean,
    baseURL: string,
    bgColor: Acrobat color array,
```

```
        duration: number,
        endAt: MediaOffset,
        layout: number,
        palindrome: boolean,
        rate: number,
        repeat: number,
        showUI: boolean,
        startAt: MediaOffset,
        visible: boolean,
        volume: number,
    },
    methods:
    {
        pause: [],
        play: [],
        seek: [ MediaOffset ],
        stop: [],
        where: [],
    },
    events:
    {
        Done: anything, onDone: anything, afterDone: anything,
        Error: anything, onError: anything, afterError: anything,
        Escape: anything, onEscape: anything, afterEscape: anything,
        Pause: anything, onPause: anything, afterPause: anything,
        Play: anything, onPlay: anything, afterPlay: anything,
        Ready: anything, onReady: anything, afterReady: anything,
        Script: anything, onScript: anything, afterScript: anything,
        Seek: anything, onSeek: anything, afterSeek: anything,
        Status: anything, onStatus: anything, afterStatus: anything,
        Stop: anything, onStop: anything, afterStop: anything,
    },
}
}
```

このリストの補足事項を次に示します。

- `mimeType` 、 `URL` 、 `settings` の各プロパティは、 `PlayerArgs` の対応するプロパティと同じです。
 `mimeType` プロパティは必須です。 `honors` メソッドでは、 `URL` のファイル拡張子に基づく MIME タイプの判別は行われません。 `URL` では、正しい URL 形式であれば、実在の URL でも架空の URL でも指定できます。これらのプロパティについては、 [`MediaSettings`](#) オブジェクトを参照してください。
- `methods` プロパティには、指定の MIME タイプでプレーヤーがサポートしている必要がある `MediaPlayer` メソッドをリストします。各メソッドの値には、そのメソッドに渡す引数を配列として指定します。Acrobat 7.0 で、引数のあるプレーヤーメソッドは `seek` のみです。このメソッドには、 `MediaOffset` という引数を 1 つ設定できます。これらのプロパティについては、 [`MediaPlayer`](#) オブジェクトを参照してください。

同じオブジェクトを `PlayerArgs` および `HonorsArgs` として使用する場合は、 `methods` プロパティが含まれていても問題ありません。このプロパティは `PlayerArgs` にはありませんが、不明なプロパティは `PlayerArgs` では無視されます。

- `events` プロパティには、プレーヤーがサポートしている必要があるイベントをリストします。前述のリストにも示されているように、 `on` が付いたイベント、 `after` が付いたイベント、何も付いていないイベントのいずれも指定できます。3 つとも意味は同じです。特定の `on` イベントがプレーヤーでサポートされていれば、それに対応する `after` イベントも常にサポートされています（どのプレーヤーでも `after`

イベントは同じ方法で生成されるからです)。これらのプロパティについては、[EventListener](#) オブジェクトを参照してください。

HonorsArgs の anything は、文字どおり、値が単なるプレースホルダであることを示します。したがって、PlayerArgs で次のような events オブジェクトが指定されていても、HonorsArgs として使用できます。

```
events:  
{  
    afterReady: doAfterReady( e ),  
    onClose: doOnClose( e ),  
},
```

HonorsArgs を作成している場合は、次のように簡単に書くこともできます。

```
events: { Ready: true, Close: true },
```

playerHonors 関数

この関数は、JavaScript のソースコードとして提供されており、PDF ファイルに文書レベルのスクリプトとしてコピーできます。この関数は、Acrobat 7.0 の honors メソッドと同じテストを、Acrobat 6.0 でも実行できるように拡張したものです。

Acrobat 6.0 で playerHonors を実行すると、そのコードで実装されているアルゴリズムで、Acrobat 6.0 のメディアプレーヤーの機能が検証されます。

Acrobat 7.0 以降で playerHonors を実行した場合は、honors が呼び出されます。

パラメータ

doc	Doc。
info	PlayerInfo オブジェクト。
args	テストする HonorsArgs オブジェクト。

戻り値

ブーリアン値。プレーヤープラグインが args オブジェクトのすべての内容に対応している場合は true。

例

次の例は、PlayerInfo オブジェクトの honors メソッドで示した例と同じですが、playerHonors 関数を使用しています。この例は、playerHonors のソースコードが PDF にコピーされていれば、Acrobat 6.0 と 7.0 の両方で動作します。

```
function playWithRequirements( args ) {  
    var plugins = app.media.getPlayers( 'video/avi' )  
    if( plugins ) {  
        for (var plugin in plugins) {  
            if( playerHonors( doc, plugin, args ) ) {  
                args.players = [plugin];  
                return app.media.openPlayer( args );  
            }  
        }  
    }  
}
```

PlayerInfoList

このオブジェクトは、`PlayerInfo` オブジェクトの配列に相当します。各要素（`PlayerInfo` オブジェクト）には、通常の配列表記を使用してアクセスできます。要素数は、`length` プロパティから取得できます。

このオブジェクトは、`app.media.getPlayers` メソッドによって返されます。

`app.media.createPlayer` を使用してメディアプレーヤーを作成するときは、このメソッドに渡す `PlayerArgs` オブジェクトの `settings.players` プロパティに、`PlayerInfoList` を含めることができます。作成されるプレーヤーは、リストに含まれているプレーヤーのいずれかになります。

PlayerInfoList のメソッド

select

6.0			
-----	--	--	--

選択条件に一致するプレーヤーのみを含む `PlayerInfoList` のコピーを返します。一致するプレーヤーがない場合は、空の配列が返されます。

パラメータ

object	(オプション) <code>id</code> 、 <code>name</code> 、 <code>version</code> の任意のプロパティを含むオブジェクト。各プロパティの値には、文字列か正規表現を使用できます。指定したプロパティにすべて一致するプレーヤーのみが選択されます。省略したプロパティは、選択条件として使用されません。
--------	---

戻り値

`PlayerInfoList` オブジェクト

例 1

QuickTime を使用してメディアクリップを表示します。

```
var playerList = app.media.getPlayers().select({ id: /quicktime/i });
// QuickTime では palindrome がサポートされているので使用できる。
var settings = { players: playerList, palindrome: true };
var player = app.media.openPlayer({ settings: settings });
```

例 2

プレーヤー ID にパターンマッチを使用して Flash プレーヤーを選択します。

```
var player = app.media.createPlayer();
player.settings.players = app.media.getPlayers().select({ id:/flash/i});
player.open();
```

PlugIn

5.0			
-----	--	--	--

このオブジェクトを使用すると、このオブジェクトが表すプラグインに関する情報にアクセスできます。PlugIn オブジェクトは、`app.plugins` を使用して取得します。

PlugIn のプロパティ

certified

`true` の場合、プラグインはアドビシステムズ社によって承認されています。承認済みプラグインとは、アプリケーションや文書のセキュリティに違反しないことがアドビシステムズ社によって確認されたプラグインのことです。ユーザは、承認済みプラグインのみをロードするようにビューアを設定することもできます。

型

ブーリアン

アクセス

R

例

承認されていないプラグインの数を取得します。

```
var j=0; aPlugins = app.plugins;
for (var i=0; i < aPlugins.length; i++)
    if (!aPlugins[i].certified) j++;
console.println("Report: There are "+j+" uncertified plug-ins loaded.");
```

loaded

`true` の場合、プラグインはロードされています。

型

ブーリアン

アクセス

R

name

プラグインの名前。

型

文字列

アクセス

R

例

プラグインの数を取得して、コンソールに表示します。

```
// PlugIn オブジェクトの配列を取得
var aPlugins = app.plugins;
// プラグインの数を取得
var nPlugins = aPlugins.length;
// すべてのプラグインの名前を列挙
for (var i = 0; i < nPlugins; i++)
    console.println("Plugin #" + i + " is " + aPlugins[i].name);
```

path

デバイスに依存しない、プラグインのパス。

型

文字列

アクセス

R

version

プラグインのバージョン番号。バージョン番号の整数部分はメジャーバージョン、小数部分はマイナーバージョンおよびアップデートバージョンを示します。例えば、5.11 は、プラグインのメジャーバージョンが 5、マイナーバージョンが 1、アップデートバージョンが 1 であることを示します。

型

数値

アクセス

R

PrintParams

印刷パラメータを制御する汎用オブジェクト。このパラメータは、JavaScript を使用して印刷されるすべての文書に影響を与えます。このオブジェクトを変更しても、ユーザ環境設定が変更されたり、文書が永続的に変更されたりすることはありません。

Acrobat 6.0 では、Doc の `print` メソッドの引数として PrintParams オブジェクトが使用されます。PrintParams オブジェクトは、Doc の `getPrintParams` メソッドから取得できます。また、このオブジェクトを変更することで印刷条件を設定できます。

PrintParams のプロパティの多くは、値として整数の定数を取ります。この定数には、`constants` プロパティを使用してアクセスできます。例えば、次のようにになります。

```
// デフォルトのプリンタの printParams オブジェクトを取得
var pp = this.getPrintParams();
// いくつかのプロパティを設定
pp.interactive = pp.constants.interactionLevel.automatic;
pp.colorOverride = pp.colorOverrides.mono;
// 印刷
this.print(pp);
```

`constants` のプロパティはすべて整数であり、読み取り専用です。

PrintParams のプロパティ

binaryOK

6.0			
-----	--	--	--

プリンタへのバイナリチャンネルがサポートされている場合は `true`。デフォルトは `true` です。

型

ブーリアン

アクセス

R / W

bitmapDPI

6.0			X
-----	--	--	---

ビットマップの生成や透明のラスタライズを行う場合に使用する、1インチ当たりのドット数 (DPI)。有効な範囲は 1 ~ 9600 です。文書を保護するために最大印刷解像度が指定されている場合は、いずれか小さい方の値が使用されます。デフォルトは 300 です。無効な値を設定した場合は、300 として処理されます。

[gradientDPI](#) も参照してください。

型

整数

アクセス

R / W

booklet

8.0			
-----	--	--	--

小冊子の印刷パラメータを設定するために使用するオブジェクト。booklet オブジェクトのプロパティを、次の表に示します。

プロパティ	型	アクセス	説明
binding	整数	R / W	binding プロパティは、小冊子を印刷するときの用紙の綴じ方の方向と順序を指定します。binding の値を設定するには、後述する constants の bookletBindings オブジェクトを使用します。デフォルトは Left です。
duplexMode	整数	R / W	duplexMode プロパティは、小冊子を印刷するときの両面印刷モードを指定します。duplexMode プロパティの値を設定するには、後述する constants の duplexMode オブジェクトを使用します。デフォルトは BothSides です。出力デバイスで自動両面印刷機能がサポートされていない場合は、表側と裏側の 2 回に分けて印刷することで、手動で両面印刷を行うことができます。
subsetFrom	整数	R / W	<p>subsetFrom は、小冊子の何枚目から印刷するかを指定します。このオプションによって、小冊子全体の一部のシートのみを印刷することができます。印刷対象のページ範囲を指定するオプションとは異なります。</p> <p>有効なシート番号の開始値は 0 です。これは最初のシートを表します。デフォルト値は 0 です。</p> <p>-1 という値は、最後のシートを表します。一般的にいうと、-N という負の整数は、最後のシートの (N-1) 枚前のシートを表します。例えば、-2 は最後のシートの 1 枚前のシートを、-3 は最後のシートの 2 枚前のシートを表します。</p> <p>後述の、小冊子のサブセットに関する注意を参照してください。</p>
subsetTo	整数	R / W	<p>subsetTo は、小冊子の何枚目まで印刷するかを指定します。このオプションによって、小冊子全体の一部のシートのみを印刷することができます。印刷対象のページ範囲を指定するオプションとは異なります。</p> <p>有効なシート番号の開始値は 0 です。-1 という値は、最後のシートを表します。デフォルト値は -1 です。</p> <p>-N という負の整数は、最後のシートの (N-1) 枚前のシートを表します。例えば、-2 は最後のシートの 1 枚前のシートを、-3 は最後のシートの 2 枚前のシートを表します。</p> <p>後述の、小冊子のサブセットに関する注意を参照してください。</p>

booklet.binding の値を設定するには、bookletBindings オブジェクトのプロパティを使用します。

constants.bookletBindings オブジェクト

プロパティ	説明
Left	左綴じ（西洋式に左から右に読む場合）。用紙の短辺を綴じます。
Right	右綴じ（右から左に読む場合や、日本式の縦書きの場合）。用紙の短辺を綴じます。
LeftTall	左綴じ（西洋式に左から右に読む場合）。用紙の長辺を綴じ、縦長のページを作成します。
RightTall	右綴じ（右から左に読む場合や、日本式の縦書きの場合）。用紙の長辺を綴じ、縦長のページを作成します。

`booklet.duplexMode` の値を設定するには、`bookletDuplexMode` オブジェクトのプロパティを使用します。

constants.bookletDuplexMode オブジェクト

プロパティ	説明
BothSides	自動的に用紙の両面に印刷します。選択したプリンタが、自動両面印刷をサポートしている必要があります。
FrontSideOnly	用紙の表側に表示されるページのみをすべて印刷します。印刷されたページを再び挿入して、 <code>BacksideOnly</code> モードを指定して再び印刷することで、手動で両面印刷を行うことができます。
BackSideOnly	用紙の裏側に表示されるページのみをすべて印刷します。

型

オブジェクト

アクセス

R / W

注意： (`subsetFrom` と `subsetTo` を使用した小冊子のサブセット) 印刷ページ範囲の設定 (`PrintParams` オブジェクトの `firstPage` プロパティと `lastPage` プロパティ) を変更しても、小冊子のサブセットと同じ結果を得るのは困難です。小冊子の出力結果は、印刷ページ範囲の設定によって変わることになります。例えば、8 ページの小冊子の最初のシート（両面）では、ページ番号の順序は [8, 1, 2, 7] になりますが、16 ページの小冊子の最初のシートでは、ページ番号の順序は [16, 1, 2, 15] になります。`subsetFrom` プロパティと `subsetTo` プロパティは、小冊子の特定のページを別のデバイスやメディアで印刷する場合に便利です。例えば、表紙と表紙裏はカラーで印刷し、他のページは白黒で印刷する場合があります。また、表紙ページを厚手のカラー用紙で印刷する場合もあります。

例 1 (binding)

右綴じの小冊子印刷を設定し、印刷します。

```
var pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.booklet;
pp.booklet.binding = pp.constants.bookletBindings.Right;
this.print(pp);
```

例 2 (duplexMode)

小冊子の両面のうち、表側の片面のみを印刷します。

```
pp.pageHandling = pp.constants.handling.booklet;
pp.booklet.duplexMode = pp.constants.bookletDuplexMode.FrontSideOnly;
this.print(pp);
```

例 3 (subsetFrom と subsetTo)

2枚目のシートのみを印刷するように小冊子印刷を設定し、印刷します。

```
var pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.booklet;
pp.booklet.subsetFrom = 1;
pp.booklet.subsetTo = 1;
this.print(pp);
```

2枚目のシートから最後のシートまでの小冊子印刷を設定します。

```
pp.booklet.subsetFrom = 1;
pp.booklet.subsetTo = -1;
```

最後のシートのみを印刷するように小冊子印刷を設定します。

```
pp.booklet.subsetFrom = -1;
pp.booklet.subsetTo = -1;
```

最後の2枚のシートを印刷するように小冊子印刷を設定します。

```
pp.booklet.subsetFrom = -2;
pp.booklet.subsetTo = -1;
```

最後の2枚を除くすべてのページを印刷するように小冊子印刷を設定します。

```
pp.booklet.subsetFrom = 0;
pp.booklet.subsetTo = -3;
```

colorOverride

6.0			
-----	--	--	--

カラーの設定を変更するかどうかを指定します。値には、`constants.colorOverrides` オブジェクトのプロパティを使用します。無効な値を設定した場合は、デフォルト値である `auto` として処理されます。

注意：このプロパティがサポートされているのは、Windows プラットフォームのみです。

colorOverrides 定数オブジェクト

プロパティ	説明
auto	カラーの設定を自動的に変更します。
gray	カラーをグレースケールにします。
mono	カラーをモノクロにします。

型

整数の定数

アクセス

R / W

例

```
var pp = this.getPrintParams();
pp.colorOverride = pp.constants.colorOverrides.mono;
this.print(pp);
```

colorProfile

6.0			X
-----	--	--	---

使用するカラープロファイル。使用可能なカラースペースのリストは、`printColorProfiles` から取得できます。デフォルトは、「Printer/PostScript Color Management」です。

型

文字列

アクセス

R / W

constants

6.0			
-----	--	--	--

PrintParams オブジェクトのインスタンスはこのプロパティを継承します。このプロパティは、様々な定数値が保持されているラッパーです。定数値はすべて整数であり、読み取り専用です。この定数値は、PrintParams オブジェクトの他のプロパティでオプション値として使用します。次の表に、各定数オブジェクトとそれを使用するプロパティを示します。

定数オブジェクト	この定数値を使用するプロパティ
bookletBindings	booklet
bookletDuplexMode	booklet
colorOverrides	colorOverride
fontPolicies	fontPolicy
handling	pageHandling
interactionLevel	interactive
nUpPageOrders	nUpPageOrder
printContents	printContent
flagValues	flags
rasterFlagValues	rasterFlags
subsets	pageSubset
tileMarks	tileMark
usages	usePrinterCRD useT1Conversion

型

オブジェクト

アクセス

R

downloadFarEastFonts

6.0			
-----	--	--	--

true (デフォルト) の場合、必要に応じて CJK フォントをプリンタに送信します。プリンタに CJK フォントが存在しているにも関わらず、これらのフォントが必要であると間違って通知される場合は、false に設定します。

型

ブーリアン

アクセス

R / W

fileName

6.0			
-----	--	--	--

印刷ジョブを、プリンタではなくファイルに出力する場合に使用するファイル名（デバイスに依存しないパス）。このパスには、現在の文書の位置に対する相対パスも指定できます。ファイルに出力する場合は、対話レベル ([interactive](#) を参照) を `full` に設定しても `automatic` として処理されます。

デフォルト値は空の文字列（ファイル名なし）です。

注意： ファイルに出力すると、Postscript や GDI コマンドなどの、プリンタに適したファイルが生成されます。

`printerName` が空の文字列で、`fileName` が指定されている場合は、現在の文書が PostScript ファイルとして保存されます。

型

文字列

アクセス

R / W

例

```
var pp = this.getPrintParams();
pp.fileName = "/c/print/myDoc.prn";
this.print(pp);
```

例 2

現在の文書を PostScript ファイルとして保存します。

```
var pp = this.getPrintParams();
pp.fileName = "/c/temp/myDoc.ps";
pp.printerName = "";
this.print(pp);
```

firstPage

6.0			
-----	--	--	--

印刷範囲の先頭のページ番号。番号は 0 から数えます。ページ番号ラベルに関係なく、文書の最初のページは常に 0 です。デフォルト値は 0 です。有効なページ範囲以外の値を指定した場合は、0 を指定したものと見なされます。

[lastPage](#) も参照してください。

型

整数

アクセス

R / W

例

```
var pp = this.getPrintParams();
pp.firstPage = 0;
pp.lastPage = 9;
this.print(pp);
```

flags

6.0			
-----	--	--	--

印刷を制御するフラグのビットフィールド。このフラグを設定またはクリアするには、`constants.flagValues` オブジェクトを使用してビット演算を行います。

ゼロ個以上のフラグを設定できます。サポートされていないフラグは無視されます。フラグのデフォルトには、印刷ダイアログボックスの設定値が使用されます。

constants.flagValues オブジェクト

`flagValues` オブジェクトのプロパティを次の表に示します。特に明記されていない限り、これらのプロパティは Adobe Reader では使用できません。

プロパティ	説明
<code>applyOverPrint</code>	印刷時にオーバープリントプレビューを実行します。オーバープリントがネイティブでサポートされている場合はオフにします。
<code>applySoftProofSettings</code>	カラーマネジメントを実行する前に、ソフトプルーフの設定を使用します。
<code>applyWorkingColorSpaces</code>	印刷時に作業用のカラースペースを適用します。
<code>emitHalftones</code>	文書で指定されているハーフトーンを送出します。

プロパティ	説明
emitPostScriptXObjects	(PostScript 専用) 出力に PostScript XObject の内容を含めます。
emitFormsAsPSForms	Form XObject を PS 形式に変換します。デフォルトはオフです。
maxJP2KRes	最適解像度でなく、JPEG2000 画像の最大解像度を使用します。
setPageSize	setPageSize を使用可能にします。PDF ページサイズに基づいて給紙トレイを選択します。Adobe Reader で使用できます。
suppressBG	文書の墨版合成を送出しません。
suppressCenter	ページを中央に配置しません。Adobe Reader で使用できます。
suppressCJKFontSubst	プリンタの CJK フォントの置き換えを使用しません。 kAVEmitFontAllFonts が使用されている場合は、このプロパティは適用されません。
suppressCropClip	トリミング領域ページクリップを送出しません。Adobe Reader で使用できます。
suppressRotate	ページを回転しません。Adobe Reader で使用できます。
suppressTransfer	文書のトランスマニア関数を送出しません。
suppressUCR	文書のアンダーカラーリムーバル (UCR) を送出しません。
useTrapAnnots	印刷設定が「文書のみ」である場合も、TrapNet 注釈とトンボ類の注釈を印刷します。
usePrintersMarks	印刷設定が「文書のみ」である場合も、トンボ類の注釈を印刷します。 Acrobat Professional でのみ使用できます。

型

整数

アクセス

R / W

例 1

詳細設定ダイアログボックスの「出力オプション」セクションにある「出力プレビュー設定を適用」チェックボックスにチェックを付けます。

```
pp = getPrintParams();
fv = pp.constants.flagValues;
// または pp.flags |= fv.applySoftProofSettings;;
pp.flags = pp.flags | fv.applySoftProofSettings;
this.print(pp);
```

例 2

印刷ダイアログボックスの「自動回転と中央配置」のチェックを解除します（デフォルトではチェックが付いています）。

```
pp = getPrintParams();
fv = pp.constants.flagValues;
pp.flags |= (fv.suppressCenter | fv.suppressRotate);
this.print(pp);
```

例 3

詳細設定ダイアログボックスの「PostScript オプション」セクションにある「アンダーカラーリムーバル(UCR) と墨板合成を送出」チェックボックスにチェックを付けます。

```
pp = getPrintParams();
fv = pp.constants.flagValues;
pp.flags &= ~(fv.suppressBG | fv.suppressUCR)
this.print(pp)
```

fontPolicy

6.0			
-----	--	--	--

フォントポリシーを設定します。fontPolicy プロパティの値を設定するには、constants.fontPolicies オブジェクトを使用します。デフォルトは pageRange です。

型

整数

アクセス

R / W

constants.fontPolicies オブジェクト

プロパティ	説明
everyPage	各ページの印刷前に必要なフォントを送出し、各ページの印刷後にすべてのフォントを解放します。この場合、印刷ジョブのデータは最も大きく、処理は最も遅くなりますが、プリントに必要なメモリサイズは最小になります。
jobStart	必要なフォントを印刷ジョブの開始時に送出し、印刷ジョブの終了時にそれらのフォントを解放します。この場合、印刷ジョブのデータは最も小さく、処理は最も速くなりますが、プリントに必要なメモリサイズは最大になります。
pageRange	(デフォルト) 該当するフォントを最初に使用するページの印刷前にそのフォントを送出し、そのフォントを最後に使用するページの印刷後にそのフォントを解放します。この場合は、印刷ジョブのデータが最も小さく、処理が最も速くなる上、使用するメモリサイズも抑えられます。ただし、この印刷ジョブでは、ページの順序を変更することはできないので、文書をそのまま印刷する必要があります。

注意：pageRange は、速度とメモリサイズのバランスが取れた方法です。ただし、後で PostScript ページの順序をプログラムで変更する場合は使用できません。

gradientDPI

6.0			X
-----	--	--	---

グラデーションをラスタライズするときに使用する dpi 値。この値は、視覚的な影響が少ない領域に適用されるので、通常は `bitmapDPI` よりも小さな値に設定できます。この値は 1 ~ 9600 に設定する必要があります。無効な値を設定した場合は、150 として処理されます。文書を保護するために最大印刷解像度が指定されている場合は、いずれか小さい方の値が使用されます。デフォルト値は 150 です。

型

整数

アクセス

R / W

interactive

6.0			
-----	--	--	--

ユーザが印刷ジョブを設定する際の対話レベルを指定します。このプロパティの値を設定するには、`constants.interactionLevel` オブジェクトを使用します。デフォルトは `full` です。

注意：(Acrobat 7.0) ユーザの操作を要求せずに、自動的な出力を行えるのは、バッチャイベント、コンソールイベント、メニューイベントのみです。自動的な出力をを行うには、`Doc` の `print` メソッドを呼び出すときに `bUI` を `false` に設定するか、次のように `interactive` プロパティを `silent` に設定します。

```
var pp = this.getPrintParams();
pp.interactive = pp.constants.interactionLevel.silent;
```

バッチャイベント、コンソールイベント、メニューイベント以外では、`bUI` や `interactive` の値は無視され、出力ダイアログボックスが常に表示されます。

[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)も参照してください。

型

整数

アクセス

R / W

constants.interactionLevel オブジェクト

プロパティ	説明
automatic	印刷ダイアログボックスは表示されません。印刷中は、キャンセルボタンの付いた進捗状況ダイアログボックスが表示され、印刷が完了すると自動的に消えます。
full	印刷設定を変更できる印刷ダイアログボックスが表示されます。処理を続行するには、「OK」を押す必要があります。印刷中は、キャンセルボタンの付いた進捗状況ダイアログボックスが表示され、印刷が完了すると自動的に消えます。
silent	印刷ダイアログボックスは表示されません。キャンセルボタンの付いた進捗状況ダイアログボックスも表示されません。エラーメッセージも表示されません。

例

```
var pp = this.getPrintParams();
pp.interactive = pp.constants.interactionLevel.automatic;
pp.printerName = "Adobe PDF";
this.print(pp);
```

lastPage

6.0			
-----	--	--	--

印刷範囲の最後のページ番号。番号は 0 から数えます。ページ番号ラベルに関係なく、文書の最初のページは常に 0 です。firstPage より小さい値や、有効なページ範囲にない値を指定した場合は、デフォルト値が使用されます。デフォルト値は、文書のページ数から 1 を引いた値です。

例については、[firstPage](#) を参照してください。

型

整数

アクセス

R / W

nUpAutoRotate

7.0			
-----	--	--	--

ブーリアン値。true の場合は、1 枚に複数ページを印刷するときに、用紙の領域に合わせて各ページが自動的に回転されます。デフォルトは false ですが、nUpAutoRotate は印刷設定に従って設定されます。

1 枚に複数ページを印刷するには、pageHandling を nUp に設定します。

型

ブーリアン

アクセス

R / W

nUpNumPagesH

7.0			
-----	--	--	--

1枚に複数ページを印刷するときに水平方向にレイアウトするページ数。デフォルトは2ですが、nUpNumPagesHは印刷設定に従って設定されます。

1枚に複数ページを印刷するには、pageHandlingをnUpに設定します。

型

整数

アクセス

R / W

例

1枚に複数ページを印刷するためのパラメータを設定して、文書を印刷します。

```
pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.nUp;
pp.nUpPageOrders = pp.constants.nUpPageOrders.Vertical;
pp.nUpNumPagesH = 3;
pp.nUpNumPagesV = 3;
pp.nUpPageBorder=true;
pp.nUpAutoRotate=true;
this.print(pp);
```

nUpNumPagesV

7.0			
-----	--	--	--

1枚に複数ページを印刷するときに垂直方向にレイアウトするページ数。デフォルトは2ですが、nUpNumPagesVは印刷設定に従って設定されます。

1枚に複数ページを印刷するには、pageHandlingをnUpに設定します。

例については、[nUpNumPagesH](#)を参照してください。

型

整数

アクセス

R / W

nUpPageBorder

7.0			
-----	--	--	--

ブーリアン値。true の場合は、1枚に複数ページを印刷するときに、各ページの周りにページの境界線が印刷されます。デフォルトは false ですが、nUpPageBorder は印刷設定に従って設定されます。

1枚に複数ページを印刷するには、pageHandling を nUp に設定します。

例については、[nUpNumPagesH](#) を参照してください。

型

ブーリアン

アクセス

R / W

nUpPageOrder

7.0			
-----	--	--	--

nUpPageOrder プロパティは、1枚に複数ページを印刷するときに複数のページを用紙にレイアウトする方法を指定します。nUpPageOrder プロパティの値を設定するには、constants.nUpPageOrders オブジェクトを使用します。デフォルトは Horizontal ですが、nUpPageOrder は印刷設定に従って設定されます。

1枚に複数ページを印刷するには、pageHandling を nUp に設定します。

型

整数

アクセス

R / W

constants.nUpPageOrders オブジェクト

プロパティ	説明
Horizontal	左から右、上から下にページが配置されます。
HorizontalReversed	右から左、上から下にページが配置されます。
Vertical	上から下、左から右にページが配置されます。

例

1枚に複数ページを印刷するためのパラメータを設定して、文書を印刷します。

```
pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.nUp;
pp.nUpPageOrders = pp.constants.nUpPageOrders.Horizontal;
pp.nUpNumPagesH = 2;
pp.nUpNumPagesV = 2;
pp.nUpPageBorder=true;
this.print(pp);
```

pageHandling

6.0			
-----	--	--	--

`constants.handling` オブジェクトの 7 つの値のいずれかを取ります。無効な値を設定した場合は、`shrink` として処理されます。デフォルトは `shrink` です。

型

整数

アクセス

R / W

constants.handling オブジェクト

プロパティ	Reader	説明
none		拡大または縮小は行われません。
fit		プリンタの用紙に合わせて、ページが拡大または縮小されます。
shrink		小さなページはそのまま印刷され、大きなページはプリンタの用紙に合わせて縮小されます。
tileAll	×	すべてのページがタイリングの設定に従って印刷されます。このプロパティを設定し、 <code>tileScale</code> プロパティに 1 より大きい値を設定すれば、標準サイズのページからポスターを作成することができます。

プロパティ	Reader	説明
tileLarge	×	小さなページまたは標準ページは元のサイズのまま印刷され、大きなページは複数の用紙に分割して印刷されます。
nUp		(バージョン 7.0) ページの倍率を変更して、1枚の用紙に複数ページを印刷します。 1枚に複数ページを印刷する場合の関連プロパティとしては、 nUpAutoRotate 、 nUpNumPagesH 、 nUpNumPagesV 、 nUpPageBorder 、 nUpPageOrder があります。
booklet		(バージョン 8.0) 2つ折りにしたときに正しい順序でページが並ぶように、1枚の用紙に複数ページを印刷します。小冊子の印刷パラメータを設定するには、 booklet オブジェクトのプロパティを使用します。

例 1

```
var pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.shrink;
this.print(pp);
```

例 2

1枚に複数ページを印刷するためのパラメータを設定して、文書を印刷します。

```
pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.nUp;
pp.nUpPageOrders = pp.constants.nUpPageOrders.Horizontal;
pp.nUpNumPagesH = 2;
pp.nUpNumPagesV = 2;
pp.nUpPageBorder=true;
this.print(pp);
```

例 3 (バージョン 8.0)

文書を小冊子として印刷します。すべてデフォルトの設定を使用します。

```
var pp = this.getPrintParams();
pp.pageHandling = pp.constants.handling.booklet;
this.print(pp);
```

pageSubset

6.0			
-----	--	--	--

印刷するページを偶数、奇数、すべてから選択します。pageSubset の値を設定するには、[constants.subsets](#) オブジェクトを使用します。デフォルトは all です。

型

整数

アクセス

R / W

constants.subsets オブジェクト

プロパティ	説明
all	ページ範囲のすべてのページを印刷します。
even	偶数ページのみを印刷します。ページラベルは無視され、文書に 1 ~ n (ページ数) の番号が付いているものとして処理されます。
odd	奇数ページのみを印刷します。

例

```
var pp = this.getPrintParams();
pp.pageSubset = pp.constants.subsets.even;
this.print(pp);
```

printAsImage

6.0			
-----	--	--	--

true に設定すると、各ページがビットマップとして送信されます。処理速度は低速になり、仕上がりも粗くなります。プリンタの PostScript インタプリタの問題を回避することができます。ビットマップの解像度を変更するには、bitmapDPI を設定します。対話レベル ([interactive](#) を参照) が full の場合は、印刷ダイアログボックスで指定した printAsImage 用の設定が使用されます。デフォルトは false です。

型

ブーリアン

アクセス

R / W

printContent

6.0			
-----	--	--	--

印刷ジョブの内容を設定します。printContent プロパティの値を設定するには、constants.printContents オブジェクトを使用します。デフォルトは doc です。

型

整数

アクセス

R / W

constants.printContents オブジェクト

プロパティ	説明
doc	文書のコンテンツを印刷します。注釈は印刷しません。
docAndComments	文書のコンテンツと注釈を印刷します。
formFieldsOnly	フォームフィールドの内容のみを印刷します。フォームフィールドが既に印刷された用紙に、値のみを印刷する場合に便利です。

例

フォームフィールドの内容のみを silent で印刷するようにプリンタを設定します。

```
var pp = this.getPrintParams();
pp.interactive = pp.constants.interactionLevel.silent;
pp.printContent = pp.constants.printContents.formFieldsOnly;
this.print(pp);
```

printerName

6.0			
-----	--	--	--

出力先プリンタの名前。このプロパティは、Windows でのみ使用できます。現在、Macintosh では、このプロパティを使用して出力先プリンタを設定することはできません。

デフォルトでは、printerName はデフォルトプリンタの名前に設定されています。printerName を空の文字列に設定すると、デフォルトプリンタが使用されます。printerName が空の文字列で、fileName が空の文字列でない場合は、現在の文書が PostScript ファイルとして保存されます。次の[例 2](#) を参照してください。

app.printerNames も参照してください。

型

文字列

アクセス

R / W

例 1

```
var pp = this.getPrintParams();
pp.printerName = "hp officejet d series";
this.print(pp);
```

例 2

現在の文書を PostScript ファイルとして保存します。

```
var pp = this.getPrintParams();  
pp.fileName = "/c/temp/myDoc.ps";  
pp.printerName = "";  
this.print(pp);
```

psLevel

6.0			
-----	--	--	--

PostScript プリンタに送出される PostScript のレベル。レベル 0 の場合は、プリンタの PostScript レベルを使用します。レベル 1 はサポートされていません。psLevel の現在の有効値は、0、2、3 です。プリンタで PostScript レベル 1 のみがサポートされている場合は、printAsImage が true に設定されます。無効な値を設定した場合は、3 として処理されます。psLevel のデフォルト値は 3 です。

型

整数

アクセス

R / W

rasterFlags

6.0			X
-----	--	--	---

フラグのビットフィールド。これらのフラグを設定またはクリアするには、constants.rasterFlagValues オブジェクトを使用してビット演算を行います。印刷ダイアログボックスの設定値がデフォルトになります。

型

整数

アクセス

R / W

constants.rasterFlagValues オブジェクト

次の表に、rasterFlagValues オブジェクトのプロパティを示します。いずれのプロパティも、Adobe Reader では使用できません。

プロパティ	説明
textToOutline	アウトラインに変換されたテキストは、太くなることがあります（特に小さなフォントの場合）アートワーク内に半透明なテキストが含まれる場合、統合中にテキストがアウトラインに変換されることがあります、アートワークに含まれないテキストと太さが異なるように見えることがあります。この場合、このオプションをオンになると、すべてのテキストの外観が崩ります。
strokesToOutline	アウトラインに変換されたストロークは、太くなることがあります（特に細いストロークの場合）アートワーク内に半透明のストロークが含まれる場合、統合中にストロークがアウトラインに変換されることがあります、アートワークに含まれないストロークと太さが異なるように見えることがあります。この場合、このオプションをオンになると、すべてのストロークの外観が崩ります。
allowComplexClip	このオプションを使用すると、ベクトルアートワークとラスタライズされたアートワークの境界がオブジェクトのパスに沿って配置されます。これにより、オブジェクトの一部を統合し、それ以外の部分をベクトル形式で残した場合に生じる継ぎ目が少なくなります。ただし、これにより、パスが複雑になりすぎて、プリントで処理できなくなることがあります。
preserveOverprint	色分解出力を行う場合で、文書にオーバープリント指定のオブジェクトが含まれているときは、このオプションを選択します。このオプションを選択すると、通常、透過的でないオブジェクトのオーバープリントが維持されるので、パフォーマンスが向上します。コンポジット印刷の場合、このオプションは無効です。このオプションをオフになると、透過的であるかどうかに関係なくすべてのオーバープリントが統合されるので、より一貫した出力が生成されます。

例 1

アドバンスト／印刷工程メニュー内の分割・統合プレビューダイアログボックスの「透明の分割・統合プリセットオプション」で「すべてのテキストをアウトラインに変換」にチェックを付けたプリセットを設定します。

```
pp = getPrintParams();
rf = pp.constants.rasterFlagValues;
pp.rasterFlags |= rf.textToOutline;
this.print(pp);
```

例 2

アドバンスト／印刷工程メニュー内の分割・統合プレビューダイアログボックスの「透明の分割・統合プリセットオプション」で「複雑な領域をクリップ」のチェックを解除したプリセット（デフォルトでチェックが付いています）を設定します。

```
pp = getPrintParams();
rf = pp.constants.rasterFlagValues;
pp.rasterFlags = pp.rasterFlags & ~rf.allowComplexClip;
// または pp.rasterFlags &= ~rf.allowComplexClip;
this.print(pp);
```

reversePages

6.0			
-----	--	--	--

ページを逆順で（最後のページから最初のページに向かって）印刷する場合は、`true` に設定します。デフォルト値は `false` です。

型

ブーリアン

アクセス

R / W

tileLabel

6.0			X
-----	--	--	---

タイル形式で出力する各ページにラベルを付けます。ラベルが設定されたページには、行、列、ファイル名、印刷日が印刷されます。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

tileMark

6.0			X
-----	--	--	---

タイルマークは、ページを分割する場所と重ね合わせる位置を示します。この値を設定するには、`constants.tileMarks` オブジェクトを使用します。無効な値を設定した場合は、`none` として処理されます。デフォルトは `none` です。

型

整数

アクセス

R / W

constants.tileMarks オブジェクト

プロパティ	説明
none	タイルマークなし
west	西洋式のタイルマーク
east	日本式のタイルマーク

tileOverlap

6.0			X
-----	--	--	---

タイル印刷したページを重ね合わせる部分のポイント数。0 ~ 144 の値を指定する必要があります。無効な値を指定した場合は、0 として処理されます。デフォルト値は 0 です。

型

整数

アクセス

R / W

tileScale

6.0			X
-----	--	--	---

タイル印刷するページの倍率。タイル印刷しないページは、この値の影響を受けません。デフォルトは拡大なし (1.0) です。大きな値を指定すると、印刷サイズが大きくなります (2.0 の場合は 2 倍に拡大し、0.5 の場合は半分に縮小します)。tileScale には、0.01 ~ 99.99 の値を指定する必要があります。無効な値を指定した場合は、1.0 として処理されます。デフォルト値は 1.0 です。

型

数値

アクセス

R / W

transparencyLevel

6.0			X
-----	--	--	---

高度な描画処理を維持する処理レベルを表す 1 ~ 100 の整数値。値が 1 の場合、画像は完全にラスタライズされるので品質は低下しますが、速度は向上します。値が 100 の場合、画像は可能な限りラスタライズされずに維持されますが、印刷速度が低下することがあります。無効な値を設定した場合は、75 が使用されます。ラスタライズには、`bitmapDPI` および `gradientDPI` 値が使用されます。デフォルト値は 75 です

型

整数

アクセス

R / W

usePrinterCRD

6.0			
-----	--	--	--

3 つの値のいずれかを取ります。この値を設定するには、`constants.usages` オブジェクトを使用します。[useT1Conversion](#) も参照してください。2 つのプロパティは同じ値を使用しますが、解釈は異なります。

型

整数

アクセス

R / W

constants.usages オブジェクト

プロパティ	説明
auto	プリンタのカラーレンダリングディクショナリ (CRD) を使用するかどうかを自動判別します。Acrobat には、不正な CRD が含まれているプリンタのリストが保持されています。無効な値を設定した場合は、 <code>auto</code> として処理されます。デフォルトは <code>auto</code> です。
use	プリンタの CRD を使用します。
noUse	プリンタの CRD を使用しません。

useT1Conversion

6.0			
-----	--	--	--

3つの値のいずれかを取ります。useT1Conversion プロパティの値を設定するには、constants.usages オブジェクトを使用します。[usePrinterCRD](#) も参照してください。2つのプロパティは同じ値を使用しますが、解釈は異なります。

注意：このプロパティがサポートされているのは、Windows プラットフォームのみです。

型

整数

アクセス

R / W

このプロパティは、usages オブジェクト ([585 ページを参照](#)) の値を使用します。

プロパティ	説明
auto	Type 1 フォントから、より効率的なフォント形式 (TrueType など) への変換を禁止するかどうかを、自動判別します。Acrobat には、これらのフォントに関して問題のあるプリンタのリストが保持されています。 無効な値を設定した場合は、auto として処理されます。デフォルトは auto です。
use	プリンタの代替フォント形式に問題があることが判明している場合でも、Type 1 フォントの変換を許可します。
noUse	Type 1 フォントをより効率的な形式に変換しません。

RDN

相対識別名（Relative Distinguished Name）を表す汎用オブジェクト。このオブジェクトは、`securityHandler.newUser`、`certificate.issuerDN` および `subjectCN` プロパティで使用されます。

このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	バージョン	説明
businessCategory	文字列	R	8.0	ビジネスカテゴリー
c	文字列	R	5.0	国または地域。ISO 3166 規格に準拠した、2つの大文字からなる文字列を指定する必要があります（「US」など）。
cn	文字列	R	5.0	一般名（「John Smith」など）
countryOfCitizenship	文字列	R	8.0	国籍
countryOfResidence	文字列	R	8.0	居住国
dateOfBirth	文字列	R	8.0	生年月日（newUser ではサポートされません）
dc	文字列	R	8.0	ドメインコンポーネント
dnQualifier	文字列	R	8.0	識別名の修飾子
e	文字列	R	6.0	電子メールアドレス（「jsmith@example.com」など）
gender	文字列	R	8.0	性別
generationQualifier	文字列	R	8.0	世代の識別子
givenName	文字列	R	8.0	名
initials	文字列	R	8.0	イニシャル
l	文字列	R	8.0	地域または県
name	文字列	R	8.0	氏名
nameAtBirth	文字列	R	8.0	出生時の氏名
o	文字列	R	5.0	組織名（「Adobe Systems Incorporated」など）
ou	文字列	R	5.0	組織単位（「Acrobat エンジニアリング部」など）
placeOfBirth	文字列	R	8.0	出生地
postalAddress	文字列	R	8.0	郵便の宛先（newUser ではサポートされません）
postalCode	文字列	R	8.0	郵便番号

プロパティ	型	アクセス	バージョン	説明
pseudonym	文字列	R	8.0	ペンネーム
serialNumber	文字列	R	8.0	シリアル番号
sn	文字列	R	8.0	姓
st	文字列	R	8.0	州
street	文字列	R	8.0	番地
title	文字列	R	8.0	肩書

ReadStream

ReadStream オブジェクトは、データのストリームを表すオブジェクトリテラルです。このオブジェクトには、データストリームを読み取るためのメソッドが用意されています。

メソッド	パラメータ	戻り値	説明
read	nBytes	文字列	read メソッドは、読み取るバイト数を引数に取り、ストリームのデータを 16 進エンコードした文字列を返します。read メソッドを実行するとストリームが破壊されます。ストリームの終わりでは長さが 0 の文字列を返します。

Rendition

Rendition には、埋め込みメディアデータ（またはメディアデータの URL）や再生設定など、メディアアップの再生に必要な情報が格納されています。このオブジェクトは、Acrobat のオーサリングユーザインターフェイスのレンディション設定に相当します。

Rendition は、MediaRendition や MediaSelector のベースタイプです。Rendition オブジェクトを引数に取る関数は、MediaRendition や MediaSelector も引数に取ることができます。この節で説明するプロパティやメソッドは、MediaRendition や MediaSelector でも使用可能です。MediaRendition と MediaSelector を区別するには、`type` プロパティを使用します。

Rendition のプロパティ

altText

6.0			
-----	--	--	--

レンディションの代替テキスト文字列（代替テキストが指定されていなければ空の文字列）。このプロパティを使用できるのは、rendition の `type` が `app.media.renditionType.media` (MediaRendition) の場合のみです。

型

文字列

アクセス

R

例

レンディションの `altText` を取得します。

```
this.media.getRendition("myClip").altText;
```

`app.media.getAltTextSettings` の例を参照してください。

doc

6.0			
-----	--	--	--

Rendition が含まれている文書への参照。

型

Doc

アクセス

R

fileName

6.0			
-----	--	--	--

メディアが埋め込まれている場合、このプロパティは空の文字列を返します。それ以外の場合は、メディアのファイル名または URL を返します。このプロパティを使用できるのは、rendition の type が app.media.renditionType.media の場合のみです。

型

文字列

アクセス

R

type

6.0			
-----	--	--	--

レンディションのタイプを示す app.media.renditionType の値。

現在、MediaRendition と RenditionList の 2 つのタイプがあります。

- Rendition.type が app.media.renditionType.media の場合、Rendition は MediaRendition です。MediaRendition とは、マルチメディアのプロパティダイアログボックスの「設定」タブに表示される個々のレンディションのことです。
- Rendition.type が app.media.renditionType.selector の場合、Rendition は RenditionList です。RenditionList とは、MediaRendition の配列のことです。このリストは、マルチメディアのプロパティダイアログボックスの「設定」タブに表示されています。

今後のバージョンの Acrobat では、renditionType の値が追加される可能性があります。前述の 2 種類の app.media.renditionType 値のみを想定して JavaScript コードを作成するのは避けてください。

型

数値

アクセス

R

uiName

6.0			
-----	--	--	--

PDF ファイルのディクショナリの N エントリにある Rendition の名前。

型

文字列

アクセス

R

例

次のコードは、レンディションアクションとして実行します。

```
console.println("Preparing to play ¥""  
+ event.action.rendition.uiName + "¥""");
```

event.action.rendition については、[event](#) オブジェクトを参照してください。

Rendition のメソッド

getPlaySettings

6.0			
-----	--	--	--

MediaPlayer オブジェクトの作成に使用できる MediaSettings オブジェクトを作成して返します。

このメソッドを使用できるのは、MediaRendition の場合のみです。

パラメータ

bGetData (オプション) ブーリアン値。true の場合、MediaSettings オブジェクトの data プロパティには、MediaData が含まれます (MediaSettings.data を参照)。

戻り値

MediaSettings オブジェクト

注意： app.media.getAltTextSettings は getPlaySettings(false) を呼び出して、代替テキストを表示するための正しい設定を取得します。

MediaSettings オブジェクトのプロパティは、次のとおりです。

autoPlay
baseURL (rendition で指定された場合)

```
bgColor
bgOpacity
data (bGetData が true の場合 )
duration
endAt
layout
monitorType
palindrome
showUI
rate
repeat
startAt
visible
volume
windowType
```

現在のバージョンの Acrobat では、これらすべてのプロパティが MediaSettings オブジェクトに存在します（前述の注意に示した例外あり）。startAt などで値が指定されていない場合は、null が設定されます。ただし、この仕様は今後変更されて、実際に値を持つプロパティのみを返して、他のプロパティにはデフォルトが使用されるようになる可能性があります。

例

```
// この Rendition の MediaSettings を取得する
var settings = myRendition.getPlaySettings();
if( settings.startAt !== null ) // このような厳密な比較は避ける
...
if( settings.startAt ) // これは OK
...
```

使用例については、`app.media.getAltTextSettings` および `app.media.openPlayer` を参照してください。

select

6.0			
-----	--	--	--

MediaRendition または RenditionSelector を再生するメディアプレーヤーを選択します。Rendition が RenditionSelector の場合、`select` はその中に格納されているすべての MediaRendition を調べて、最適のものを選択します（RenditionSelector と MediaRendition については、[type](#) を参照）。

戻り値は MediaSelection オブジェクトです。このオブジェクトを使用して MediaSettings オブジェクトを作成し、この MediaSettings オブジェクトを使用して MediaPlayer オブジェクトを作成できます。

パラメータ

bWantRejects	(オプション) bWantRejects が true の場合は、戻り値である MediaSelection の rejects プロパティに、選択処理で拒否されたメディアプレーヤーに関する情報が格納されます。
oContext	(オプション) oContext には、前回の Rendition.select 呼び出しで返された MediaSelection の selectContext 値を設定します。 このパラメータを使用すれば、ループの中で Rendition.select を繰り返し呼び出すことができます。取得したメディアプレーヤーを独自のテストコードで検査していけば、目的に合ったメディアプレーヤーを抽出できます。

戻り値

MediaSelection オブジェクト

例 1

この Rendition で使用可能な MediaSelection を取得します。

```
var selection = rendition.select();
```

例 2

選択されたレンディションの名前を取得します。このスクリプトは、レンディションアクションイベントから実行します。

```
var selection = event.action.rendition.select();
console.println( "Preparing to play " + selection.rendition.uiName);
```

testCriteria

6.0			
-----	--	--	--

PDF ファイルで指定されている条件（最小帯域幅など）に基づいて Rendition をテストし、Rendition がすべての条件を満たしているかどうかを示すブーリアン値を返します。

戻り値

ブーリアン

Report

Report オブジェクトを使用すると、レポートに適した PDF 文書を生成することができます。Report オブジェクトを作成するには、次のように Report コンストラクタを使用します。

```
var rep = new Report();
```

このオブジェクトのプロパティやメソッドを使用して、レポートの作成やフォーマットが行えます。

Report のプロパティ

absIndent

5.0			X
-----	--	--	---

インデントレベルを絶対位置で制御します。インデントのオーバーフローを避けるために、なるべく `indent` や `outdent` メソッドを使用するようにしてください。

ページ中央を超えるインデントを指定した場合は、ページ中央が有効なインデント位置になります。インデントが大きすぎる場合は、`divide` の線が特別な形状になります。

型

数値

アクセス

R / W

color

5.0			X
-----	--	--	---

レポートのテキストや区切り線の色を制御します。

レポートにテキストを書き込むには `writeText` を使用し、区切り線（水平線）を書き込むには `divide` を使用します。

型

カラー

アクセス

R / W

例

```
var rep = new Report();
rep.size = 1.2;
rep.color = color.blue;
rep.writeText("Hello World!");
```

size

5.0			X
-----	--	--	---

`writeText` で作成するテキストのサイズを制御する乗数。テキストサイズは、定義済みスタイルのデフォルトサイズに `size` プロパティを掛けたものになります。

型

数値

アクセス

R / W

例

レポートを作成し、「Hello World!」と書き込みます。

```
var rep = new Report();
rep.size = 1.2;
rep.writeText("Hello World!");
```

style

6.0			X
-----	--	--	---

このプロパティは、`writeText` で作成するテキストのフォントスタイルを制御します。`style` の有効な値は、次のとおりです。

DefaultNoteText
NoteTitle

型

文字列

アクセス

R / W

例

新しいレポートを作成し、フォントサイズを設定し、短いテキストを書き込んで、レポートを開きます。

```
var rep = new Report();
rep.size = 1.2;
rep.style = "DefaultNoteText";
rep.writeText("Hello World!");
rep.open("My Report");
```

Report のメソッド

breakPage

5.0			X
-----	--	--	---

現在のページを終了し、新しいページを開始します。

divide

5.0			X
-----	--	--	---

ページを横切る水平方向の線を、現在の行に指定の線幅で作成します。線は現在のインデント位置からバウンディングボックスの右端まで引かれます。インデントがバウンディングボックスの中央を越えている場合は、特別な形状の線が表示されます。

パラメータ

nWidth (オプション) 水平線の線幅。

indent

5.0			X
-----	--	--	---

現在のインデント位置を nPoints またはデフォルトの量だけ右に移動します。ページ中央を超えるインデントを指定した場合は、ページ中央が有効なインデント位置になります。インデントが大きすぎる場合は、divide の線が特別な形状になります。

使用例については、[writeText](#) を参照してください。

パラメータ

nPoints (オプション) インデントの位置を右に移動するポイント数。

mail

5.0			X
-----	--	--	---

レポートの生成を終了し、レポートをメールで送信します。

[mailGetAddrs](#)、app.[mailMsg](#)、Doc の [mailForm](#) メソッド、FDF オブジェクトの [mail](#) メソッドも参照してください。

パラメータ

bUI (オプション) ユーザインターフェイスを表示するかどうかを指定します。true (デフォルト) の場合は、メールーの新規メッセージウィンドウがユーザに表示され、他のパラメータ値が初期値として使用されます。false の場合、cTo パラメータが必須で、その他のパラメータはすべてオプションです。

注意：(Acrobat 7.0) セキュリティによる制限があるコンテキストでこのメソッドを実行した場合、bUI パラメータは無視され、デフォルトの true の動作になります。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

cTo (オプション) セミコロンで区切られた、メッセージの受信者のリスト。

cCc (オプション) セミコロンで区切られた、メッセージの CC 受信者のリスト。

cBcc (オプション) セミコロンで区切られた、メッセージの BCC 受信者のリスト。

cSubject (オプション) メッセージの件名。長さの制限は 64 KB です。

cMsg (オプション) メッセージの内容。長さの制限は 64 KB です。

open

5.0			X
-----	--	--	---

レポートの生成を終了して Acrobat で開き、Doc を返します。この Doc を使用して、レポートに追加処理を行うことができます。

パラメータ

cTitle レポートのタイトル。

戻り値

Doc。

例

レポートを開いて Doc を取得し、文書のプロパティを設定します。

```
var docRep = rep.open("myreport.pdf");
docRep.info.Title = "End of the month report: August 2000";
docRep.info.Subject = "Summary of comments at the August meeting";
```

詳しい例については、[writeText](#) を参照してください。

outdent

5.0			X
-----	--	--	---

インデントと逆に、現在のインデント位置を `nPoints` またはデフォルトの量だけ左に移動します。

使用例については、[writeText](#) を参照してください。

パラメータ

`nPoints` (オプション) インデントの位置を左に移動するポイント数。

Report

5.0			X
-----	--	--	---

コンストラクタ。指定のメディアボックスとバウンディングボックス（値の単位はポイント、つまり 1/72 インチ）を使用して、新規の `Report` オブジェクトを作成します。デフォルトでは、メディアボックスは 8.5 × 11 インチです。バウンディングボックスは、メディアボックスのすべてのサイドから 0.5 インチだけインデントした大きさになります。

パラメータ

`aMedia` (オプション) メディアボックスのサイズ。

`aBBox` (オプション) バウンディングボックスのサイズ。

戻り値

`Report` オブジェクト。

save

5.0		S	X
-----	--	---	---

レポートの生成を終了し、指定のパスにレポートを保存します。

注意：このメソッドを実行できるのは、バッティイベントまたはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

パラメータ

cDIPath	デバイスに依存しないパス。
cFS	(オプション) ファイルシステム。cFS に有効な値は、「CHTTP」のみです。指定した場合は、cDIPath パラメータに URL を指定する必要があります。Web サーバが WebDAV をサポートする場合にのみ、このパラメータは意味を持ちます。

例 1

```
rep.save("/c/myReports/myreport.pdf");
```

例 2

```
rep.save({
  cDIPath: "http://www.example.com/reports/WebDAV/myreport.pdf",
  cFS: "CHTTP"
});
```

writeText

5.0			X
-----	--	--	---

テキストのブロックをレポートに書き込みます。この書き込みは、新しい行の現在のインデント位置から開始されます。Roman、CJK、WGL4 テキストは正しくワードラップされます。

パラメータ

String	使用するテキストブロック。
--------	---------------

例

```
// 文書内の注釈を取得し、作成者で並べ替える
this.syncAnnotScan();
annots = this.getAnnots({nSortBy: ANSB_Author});

// 新しいレポートを開く
var rep = new Report();

rep.size = 1.2;
rep.color = color.blue;

if (annots) {
  rep.writeText("Summary of Comments: By Author");
  rep.color = color.black;
  rep.writeText(" ");
  rep.writeText("Number of Comments: " + annots.length);
  rep.writeText(" ");

  var msg = "¥200 page %s: ¥"%s¥";
  var theAuthor = annots[0].author;
```

```
rep.writeText(theAuthor);
rep.indent(20);
for (var i=0; i < annots.length; i++) {
    if (theAuthor != annots[i].author) {
        theAuthor = annots[i].author;
        rep.writeText(" ");
        rep.outdent(20);
        rep.writeText(theAuthor);
        rep.indent(20);
    }
    rep.writeText(
        util.printf(msg, 1 + annots[i].page, annots[i].contents));
}
} else {
    var msg = "No annotations found in this document, %s.";
    rep.writeText(util.printf(msg, this.documentFileName));
}
// レポートを開く
var docRep = rep.open("myreport.pdf");
docRep.info.Title = "End of the month report: August 2006";
docRep.info.Subject = "Summary of comments at the August meeting";
```

Row

この汎用 JavaScript オブジェクトには、行内のすべての列のデータが含まれています。このオブジェクトは、`Statement` オブジェクトの [getRow](#) メソッドによって返されます。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
<code>columnArray</code>	配列	R	<code>Column</code> オブジェクトの配列。 この <code>row</code> オブジェクトの作成と、 <code>Statement</code> オブジェクトの getColumnArray メソッドの呼び出しを同一の <code>Statement</code> オブジェクトで同時に行った場合、このプロパティは <code>Statement</code> オブジェクトの getColumnArray メソッドから返される配列と同じになります。
列のプロパティ	任意	R	クエリで選択された列ごとに、プロパティが定義されます。 プロパティの値は、その列のデータに該当します。

ScreenAnnot

ScreenAnnot オブジェクトは、画面注釈を表します。画面注釈は、ディスプレイ画面上に表示される、PDF 文書内の矩形領域です。ScreenAnnot には、マルチメディアを再生するためのレンディションやレンディションアクションが関連付けられている場合があります。

ScreenAnnot のプロパティ

altText

6.0			
-----	--	--	--

注釈の代替テキスト文字列（代替テキストが指定されていなければ空の文字列）。

型

文字列

アクセス

R

例

注釈を取得して、その altText をデバッグコンソールに表示します。

```
var annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
console.println( "annot.altText = " + annot.altText );
```

alwaysShowFocus

6.0			
-----	--	--	--

通常、画面注釈にキーボードフォーカスがあるかどうかは、フォーカスを示す矩形の表示／非表示で示されます。このプロパティが `true` の場合は、フォーカスがなくても、フォーカスを示す矩形が表示されます。この設定は、ドッキング表示されたメディアを再生するときに、実際にはメディアプレーヤーにキーボードフォーカスがあっても、注釈のフォーカスを示す矩形を表示し続ける場合に使用します。

このプロパティは PDF ファイルに保存されません。変更した場合、その変更は現在のセッションでのみ有効になります。

型

ブーリアン

アクセス

R / W

display

6.0			
-----	--	--	--

Field オブジェクトの `display` プロパティと同じです。

このプロパティは PDF ファイルに保存されません。変更した場合、その変更は現在のセッションでのみ有効になります。

型

整数

アクセス

R / W

例

注釈を非表示にします。

```
var annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
annot.display = display.hidden;
```

doc

6.0			
-----	--	--	--

画面注釈が含まれている文書への参照。

型

Doc

アクセス

R

events

6.0			
-----	--	--	--

画面注釈に登録されている `EventListener` を含む `Events` オブジェクト。

このプロパティは PDF ファイルに保存されません。変更した場合、その変更は現在のセッションでのみ有効になります。

型

`Events` オブジェクト

アクセス

R / W

例

簡単なフォーカスの EventListener を作成します。

```
var annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
var myFocusEvent = {
    onFocus: function () {
        console.println("Focusing...");
    }
};
annot.events.add( myFocusEvent );
```

この EventListener は、次のコードを実行すれば、後で削除できます。

```
annot.events.remove( myFocusEvent );
```

extFocusRect

6.0			
-----	--	--	--

画面注釈でフォーカスを示す矩形が表示されるとき、通常、その矩形は画面注釈のみを囲みます。

`extFocusRect` を指定すると、画面注釈は通常の矩形と `extFocusRect` を合わせた矩形を使用して、フォーカスを示します。

このプロパティは PDF ファイルに保存されません。変更した場合、その変更は現在のセッションでのみ有効になります。

型

4 つの数値の配列

アクセス

R / W

innerDeviceRect

6.0			
-----	--	--	--

このプロパティと `outerDeviceRect` は、現在のページビューに表示されている画面注釈の、内側の矩形と外側の矩形を定義します。

型

4 つの数値の配列

アクセス

R

例

innerDeviceRect を取得します。

```
annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
console.println("annot.innerDeviceRect = "
+ annot.innerDeviceRect.toSource() );
```

noTrigger

6.0			
-----	--	--	--

true の場合は、画面注釈のアクションを Acrobat のユーザインターフェイスから起動できません。通常は、画面注釈をクリックするとメディアプレーヤーの再生が始まりますが、このプロパティはそれを抑制します。

このプロパティは PDF ファイルに保存されません。変更した場合、その変更は現在のセッションでのみ有効になります。

型

ブーリアン

アクセス

R / W

例

フォームボタンでメディアクリップを制御するために、注釈の操作をオフにします。

```
annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
annot.noTrigger = true;
```

outerDeviceRect

6.0			
-----	--	--	--

このプロパティと innerDeviceRect は、現在のページビューに表示されている画面注釈の、外側の矩形と内側の矩形を定義します。

型

4 つの数値の配列

アクセス

R

page

6.0			
-----	--	--	--

画面注釈がある PDF ファイルのページ番号。

型

数値

アクセス

R

player

6.0			
-----	--	--	--

画面注釈に関連付けられている MediaPlayer への参照。このプロパティは、MediaPlayer が関連付けられている ScreenAnnot オブジェクトにのみ存在します。このプロパティは、MediaPlayer.open によって設定されるか、open を間接的に呼び出す app.media.openPlayer などのメソッドによって設定されます。

型

ScreenAnnot

アクセス

R / W

rect

6.0	④		
-----	---	--	--

デフォルトユーザ座標における、画面注釈の矩形。このプロパティを変更すると、PDF ファイルにその情報が追加されます。PDF ファイルを保存すると、新しい設定も保存されます。innerDeviceRect と outerDeviceRect も更新され、新しい矩形が反映されます。

型

4 つの数値の配列

アクセス

R / W

例

注釈の場所を微調整します。

```
var annot = this.media.getAnnot({ nPage:0, cAnnotTitle: "myScreen" });
var aRect = annot.rect;
aRect[0] += 10;
aRect[2] += 10;
annot.rect = aRect;
```

ScreenAnnot のメソッド

hasFocus

6.0			
-----	--	--	--

画面注釈に現在キーボードフォーカスがあるかどうかを示します。

戻り値

ブーリアン

setFocus

6.0			
-----	--	--	--

画面注釈にキーボードフォーカスを設定します。安全な場合は、(setFocus から制御が戻る前に) 同期的にフォーカスが設定されます。同期的にフォーカスを設定するのが安全でない場合（例えば、on イベントメソッドでプロパティが変更される場合など）は、bAllowAsync の設定に従って次のように処理されます。

- true の場合は、アイドル時にフォーカスが非同期に設定されます。
- false であるか省略されている場合、フォーカスは変更されません。

戻り値は、同期的に処理された場合は true になり、遅延して非同期に処理された場合は false になります。

パラメータ

bAllowAsync (オプション) ブーリアン値。同期的にフォーカスを設定するのが安全でない場合の setFocus の動作を決定します。true の場合は、アイドル時にフォーカスが非同期に設定されます。false であるか省略されている場合、フォーカスは変更されません。デフォルトは false です。

戻り値

ブーリアン

search

search オブジェクトは、Acrobat Search プラグインの機能にアクセスするための静的なオブジェクトです。search オブジェクトにアクセスするには、このプラグインがインストールされている必要があります ([available を参照](#))。

Index オブジェクトも参照してください。これは、search オブジェクトの一部のメソッドで返されます。

query の結果は、Acrobat の検索パネルに表示されます。

注意：Acrobat 7.0 用のインデックスは、Acrobat 5.0 以前の検索エンジンとは互換性がありません。

Macintosh 版の Acrobat 7.0 では、バージョン 5.0 以前の Acrobat で作成されたインデックスは検索できません。

search のプロパティ

attachments

7.0			
-----	--	--	--

文書本体に加えて、PDF の添付ファイルも検索するかどうかを示します。デフォルトは `false` です。

Macintosh プラットフォームで Safari Web ブラウザから文書を検索する場合は、このプロパティは無視されます。そのため、Safari 内部では、添付ファイルは検索されません。

型

ブーリアン

アクセス

R / W

available

5.0			
-----	--	--	--

Search プラグインがロード済みでクエリ機能が実行可能である場合は、`true` を返します。search オブジェクトでクエリなどの操作を行う前に、このプロパティの値を確認してください。

型

ブーリアン

アクセス

R

例

```
search オブジェクトが存在し、使用可能であることを確認します。  
if (typeof search != "undefined" && search.available) {  
    search.query("Cucumber");  
}
```

bookmarks

6.0			
-----	--	--	--

クエリでしおりを検索するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

docInfo

6.0			
-----	--	--	--

クエリで文書情報を検索するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

docText

6.0			
-----	--	--	--

クエリで文書テキストを検索するかどうかを指定します。デフォルトは `true` です。

型

ブーリアン

アクセス

R / W

docXMP

6.0			
-----	--	--	--

クエリで文書レベル XMP メタデータを検索するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

ignoreAccents

70			
----	--	--	--

用語検索の際に、アクセントや発音区別符号を無視するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

ignoreAsianCharacterWidth

6.0			
-----	--	--	--

検索クエリで文書内の仮名文字を正確にマッチさせるかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

indexes

5.0			
-----	--	--	--

検索エンジンで現在アクセス可能なすべての `Index` オブジェクトの配列。

注意：(Acrobat 7.0) このプロパティにアクセスできるのは、バッチイベントまたはコンソールイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

型

配列

アクセス

R

例

すべてのインデックスを列挙し、インデックス名を表示します。

```
for (var i = 0; i < search.indexes.length; i++) {  
    console.println("Index[" + i + "]=", search.indexes[i].name);  
}
```

jpegExif

6.0			
-----	--	--	--

PDF 文書内の JPEG 画像に関連付けられた EXIF データを検索するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

legacySearch

6.0			
-----	--	--	--

`Search5.api` プラグインがロードされている場合は `true` を返します。このプラグインを使用すれば、Acrobat 5.0 以前のバージョンの Acrobat Catalog で生成されたインデックスを検索できます。Acrobat ヘルプのインデックス検索に関する節を参照してください。

型

ブーリアン

アクセス

R

markup

6.0			
-----	--	--	--

クエリでマークアップ（注釈）を検索するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

matchCase

5.0			
-----	--	--	--

検索クエリで大文字と小文字を区別するかどうかを指定します。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

matchWholeWord

6.0			
-----	--	--	--

クエリで指定した単語に完全に一致するもののみを検索するかどうかを指定します。例えば、このオプションを `true` にして「stick」という単語を検索すると、「tick」や「sticky」という単語には一致しなくなります。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

maxDocs

5.0			
-----	--	--	--

検索クエリの結果として返す文書の最大数。デフォルトは 100 文書です。

型

整数

アクセス

R / W

objectMetadata

7.0			
-----	--	--	--

オブジェクトレベルのメタデータを検索するかどうかを指定します。この設定は、Acrobat 7.0 でツール／オブジェクトデータ／オブジェクトデータツールを選択すると表示されるデータと同じです。

デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

proximity

5.0			
-----	--	--	--

AND ブーリアン句を含む検索を実行する場合、単語の近似性を検索クエリの結果ランキングに反映するかどうかを指定します。デフォルトは `false` です。近似性について詳しくは、Acrobat ヘルプの検索機能に関する節を参照してください。

型

ブーリアン

アクセス

R / W

proximityRange

7.0			
-----	--	--	--

近似検索をする場合の、指定した単語間のワード数。このプロパティが使用されるのは、`proximity` プロパティが `true` に設定されている場合のみです。近似性について詳しくは、Acrobat ヘルプの検索機能に関する節を参照してください。

デフォルトは 900 語です。このパラメータには、ゼロ以外の任意の正の整数を指定できます。

型

整数

アクセス

R / W

refine

5.0			
-----	--	--	--

前回のクエリ結果に対して新しいクエリを適用し、検索結果を絞り込むかどうかを指定します。デフォルトは `false` です。絞り込み検索について詳しくは、Acrobat ヘルプの検索機能に関する節を参照してください。

型

ブーリアン

アクセス

R / W

soundex

X			
---	--	--	--

注意： Acrobat 6.0 以降では、このプロパティの使用は推奨されません。このプロパティは値が `false` であり、アクセスは読み取り専用に制限されています。

単語の発音（MacMillan、McMillan、McMilon など）を考慮して検索クエリを実行するかどうかを指定します。デフォルトは `false` です。soundex について詳しくは、Acrobat ヘルプの検索機能に関する節を参照してください。

型

ブーリアン

アクセス

R

stem

5.0			
-----	--	--	--

単語の語幹（run、runs、runningなど）を考慮して検索クエリを実行するかどうかを指定します。デフォルトは `false` です。語幹について詳しくは、Acrobat ヘルプの検索機能に関する節を参照してください。

型

ブーリアン

アクセス

R / W

thesaurus

X			
---	--	--	--

注意： Acrobat 6.0 以降では、このプロパティの使用は推奨されません。このプロパティは値が `false` であり、アクセスは読み取り専用に制限されています。

検索クエリで類似した単語を検索するかどうかを指定します。例えば、このプロパティを `true` にして「embellish」を検索すると、「enhanced」、「gracefully」、「beautiful」などの単語も一致します。デフォルトは `false` です。

型

ブーリアン

アクセス

R

wordMatching

6.0			
-----	--	--	--

クエリ内の各単語と文書内の単語との一致方法。有効な値は、次のとおりです。

MatchPhrase
MatchAllWords
MatchAnyWord
BooleanQuery (デフォルト)

このプロパティは、クエリに複数の単語が設定されている場合にのみ関係します。アクティブな文書を検索する場合、BooleanQuery オプションは無視されます。

型

文字列

アクセス

R / W

search のメソッド

addIndex

5.0	P		
-----	---	--	--

指定のインデックスを検索可能インデックスのリストに追加します。

パラメータ

cDIPath	デバイスに依存しない、ユーザのハードドライブにあるインデックスファイルのパス。
bSelect	(オプション) インデックスを検索用に選択するかどうか。

戻り値

Index オブジェクト。

例

Acrobat の標準ヘルプインデックスをインデックスリストに追加します。

```
search.addIndex("/c/program files/adobe/Acrobat 5.0/help/exchhelp.pdx",  
true);
```

getIndexForPath

5.0			
-----	--	--	--

インデックスリストを検索し、指定のパスに対応する Index オブジェクトを返します。

パラメータ

cDIPath	デバイスに依存しない、ユーザのハードドライブにあるインデックスファイルのパス。
---------	---

戻り値

指定のパスに対応する Index オブジェクト。

query

5.0			
-----	--	--	--

指定の文書またはインデックスで指定のテキストを検索します。search オブジェクトの関連プロパティ (matchCase、matchWholeWord、stem など) によって、結果が影響を受ける可能性があります。

パラメータ

cQuery	検索するテキスト。
cWhere	(オプション) テキストを検索する場所を指定します。有効な値は、次のとおりです。 ActiveDoc Folder Index ActiveIndexes (デフォルト)
cDIPath	(オプション) デバイスに依存しない、ユーザのコンピュータにあるフォルダまたは Catalog インデックスのパス。 cWhere が Folder または Index の場合は、このパラメータを指定する必要があります。

例

「Acrobat」という単語を検索します。

cWhere	クエリ
ActiveIndexes	search.query("Acrobat"); // 「ActiveIndexes」がデフォルト。 search.query("Acrobat", "ActiveIndexes");
ActiveDoc	search.query("Acrobat", "ActiveDoc");

cWhere	クエリ
Folder	<pre>search.query("Acrobat", "Folder", "/c/myDocuments"); search.query("Acrobat", "Folder", app.getPath("user", "documents")); search.query("Acrobat", "Folder", "//myserver/myDocuments");</pre>
Index	<pre>search.query("Acrobat", "Index", "/c/Myfiles/public/index.pdx");</pre>

removeIndex

5.0	P		
-----	----------	--	--

インデックスリストから指定の Index オブジェクトを削除します。

パラメータ

index	インデックスリストから削除する Index オブジェクト。
-------	-------------------------------

security

security オブジェクトは、暗号化や電子署名などの、セキュリティ関連の PDF 機能を提供する静的な JavaScript オブジェクトです。セキュリティ機能を実行するには、security オブジェクトの `getHandler` メソッドで取得できる `SecurityHandler` オブジェクトを使用します。

注意： security オブジェクトは、Adobe Reader でも制限なく使用できます。security オブジェクトのメソッドやプロパティを実行できるのは、特に明記されていない限り、Adobe Acrobat と Adobe Reader のバッチイベント、コンソールイベント、アプリケーション初期化イベントのみです。[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) も参照してください。JavaScript イベントについて詳しくは、[event](#) オブジェクトを参照してください。

security の定数

Acrobat 7.0 以降では、いくつかの便利な文字列が security オブジェクトに定義されています。これらの定数は、次に示すラッパーオブジェクトのプロパティとして定義されています。

HandlerName オブジェクト

これらは、使用するハンドラを指定する際に使用する定数です。

プロパティ	型	アクセス	説明
StandardHandler	文字列	R	この値は、標準セキュリティハンドラ（パスワードベース）を表します。SecurityPolicy オブジェクトの <code>handler</code> プロパティで指定できます。
PPKLiteHandler	文字列	R	この値は、PPKLite セキュリティハンドラ（証明書ベース）を表します。SecurityPolicy オブジェクトの <code>handler</code> プロパティで指定できます。この値を <code>security.getHandler</code> に渡して、新しいセキュリティコンテキストを作成することもできます。
APSHandler	文字列	R	この値は、Adobe Policy Server セキュリティハンドラを表します。SecurityPolicy オブジェクトの <code>handler</code> プロパティで指定できます。この値を <code>security.getHandler</code> に渡して、新しいセキュリティコンテキストを作成することもできます。

例

次の例では、`SecurityPolicy` オブジェクトの `handler` プロパティに、`security.StandardHandler` 定数（文字列）を指定します。

```
security.getHandler(security.PPKLiteHandler, true);
```

EncryptTarget オブジェクト

これらは、暗号化するデータを指定する際に使用する定数です。securityPolicy オブジェクトの target プロパティで使用できます。

プロパティ	型	アクセス	説明
EncryptTargetDocument	文字列	R	セキュリティポリシーで文書全体を暗号化します。
EncryptTargetAttachments	文字列	R	セキュリティポリシーで文書に埋め込まれた添付ファイルのみを暗号化します。つまり、適切なセキュリティ認証が行われない場合、文書を開くことはできますが添付ファイルを開くことはできません。これは、eEnvelope (電子封筒) のワークフローで使用されます。

例

```
var filterOptions = { target: security.EncryptTargetAttachments };
security.chooseSecurityPolicy( { oOptions: filterOptions } );
```

security のプロパティ

handlers

5.0			
-----	--	--	--

言語に依存しない、暗号化または署名に使用可能なセキュリティハンドラの名前の配列。

[getSecurityPolicies](#) も参照してください。

Acrobat のバージョンに関連する情報を次に示します。

- Acrobat 6.0 以降、このプロパティのアクセス制限は解除されました。これによって、使用可能なハンドラを取得できます。
- Acrobat 6.0 で呼び出すと、Adobe.PPKLite、Adobe.PPKMS、Adobe.AAB の 3 つのハンドラが返されます。Acrobat 7.0 以降では、Adobe.PPKMS の機能はすべて Adobe.PPKLite に移行され、Adobe.PPKMS を独立したハンドラとして使用することはできなくなりました。
- Acrobat 7.0 以降では、Adobe.APS という新しいハンドラが使用可能になりました。このハンドラは、encryptUsingPolicy、getSecurityPolicies、chooseSecurityPolicy の各メソッドを呼び出す前の認証に使用します。現在のところ、他に有効な使用方法はありません。
- Acrobat 7.0 以降では、各ハンドラを表す security の定数 ([620 ページ](#)を参照) が定義されています (Adobe.AAB は例外です。このハンドラは近い将来非推奨になる可能性が高いので、この定数は追加されていません)。getHandler を使用してハンドラの新しいインスタンスを作成するときや、ハンドラリストと比較するときは、これらの定数を使用してください。

型

配列

アクセス

R

例

このシステムで使用可能なセキュリティハンドラを取得します。

```
for ( var i=0; i < security.handlers.length; i++ )
    console.println(security.handlers[i])
```

コンソールの出力は、次のようにになります。

```
Adobe.APS
Adobe.PPKLite
Adobe.PPKMS
Adobe.AAB
```

validateSignaturesOnOpen

5.0			
-----	--	--	--

文書を開いたときに署名を自動的に検証するかどうかを示す、ユーザレベルの環境設定を取得または設定します。

注意：このプロパティはどのイベントでも取得できますが、新しい値を設定できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

型

ブーリアン

アクセス

R / W

security のメソッド

chooseRecipientsDialog

6.0			
-----	--	--	--

受信者のリストを選択するためのダイアログボックスを開き、汎用 Group オブジェクトの配列を返します。この配列を Doc の `encryptForRecipients` メソッドや `addRecipientListCryptFilter` メソッドで使用して、文書やデータを暗号化できます。

注意：これを実行できるのは、コンソールイベント、メニューイベント、アプリケーション初期化イベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

パラメータ

`oOptions` 表示オプションのパラメータを含む `DisplayOptions` オブジェクト。

戻り値

Group オブジェクトの配列。

Group オブジェクトについては、Doc の [encryptForRecipients](#) メソッドを参照してください。

DisplayOptions オブジェクト

`DisplayOptions` オブジェクトのプロパティは、次のとおりです。

プロパティ	説明
<code>bAllowPermGroups</code>	受信者リストのエントリに権限を設定できるかどうかを制御します。デフォルト値は <code>true</code> です。
<code>bPlaintextMetadata</code>	このプロパティを指定すると、メタデータをプレーンテキストにするか暗号化するかを選択するためのチェックボックスが表示されます。チェックボックスのデフォルト値は、このプロパティの値になります (<code>true</code> または <code>false</code>)。このプロパティを指定しなかった場合、チェックボックスは表示されません。
<code>cTitle</code>	ダイアログボックスに表示するタイトル。デフォルトは「受信者を選択してください」です。
<code>cNote</code>	ダイアログボックスに表示する注意。デフォルトでは、注意を表示しません。
<code>bAllowImportFromFile</code>	ファイルから受信者を取り込むためのオプションを表示するかどうかを指定します。デフォルト値は <code>true</code> です。
<code>bRequireEncryptionCert</code>	<code>true</code> の場合、受信者に暗号化証明書が含まれている必要があります。デフォルト値は <code>true</code> です。

プロパティ	説明
bRequireEmail	true の場合、受信者に電子メールアドレスが含まれている必要があります。デフォルト値は false です。
bUserCert	true の場合は、ユーザの証明書を指定するように求めるプロンプトが表示され、受信者のリストにユーザ自身を追加できるようになります。このフラグを true に設定するとプロンプトが表示されますが、証明書を指定する必要はありません。

例 1

権限を持つグループを取得します。

```
var oOptions = {
    bAllowPermGroups: true,
    bPlaintextMetadata: false,
    cTitle: "Encrypt and Email",
    cNote: "Select recipients",
    bAllowImportFromFile: false,
    bRequireEncryptionCert: true,
    bRequireEmail: true
};
var groupArray = security.chooseRecipientsDialog( oOptions );
console.println("Full name = " + groupArray[0].userEntities[0].fullName);
```

例 2

受信者のリストを取得してデータを暗号化し、その文書を電子メールで送信します。

```
var oOptions = { bAllowPermGroups: false,
    cNote: "Select the list of recipients. "
    + "Each person must have both an email address and a certificate.",
    bRequireEmail: true,
    bUserCert: true
};
var oGroups = security.chooseRecipientsDialog( oOptions );
// 受信者のリストを警告に表示
// mailList 宛ての電子メールを作成
var numCerts = oGroups[0].userEntities.length;
var cMsg = "The document will be encrypted for the following:¥n";
var mailList = new Array;
for( var g=0; g<numCerts; ++g )
{
    var ue = oGroups[0].userEntities[g];
    var oCert = ue.defaultEncryptCert;
    if( oCert == null )
        oCert = ue.certificates[0];
    cMsg += oCert.subjectCN + ", " + ue.email + "¥n";
    var oRDN = oCert.subjectDN;
    if( ue.email )
    {
        mailList[g] = ue.email;
    }
}
```

```
    else
        if ( oRDN.e )
        {
            mailList[g] = oRDN.e;
        }
    }
var result = app.alert( cMsg );
```

例 3

グループ配列内のすべてのエントリを表示します。

```
var groups = security.chooseRecipientsDialog( oOptions );
for( g in groups ) {
    console.println( "Group No. " + g );
    // 権限
    var perms = groups[g].permissions;
    console.println( "Permissions:" );
    for(p in perms) console.println( p + " = " + eval("perms." +p));
    // ユーザエンティティ
    for( u in groups[i].userEntities ) {
        var user = groups[g].userEntities[u];
        console.println( "User No. " + u );
        for(i in user) console.println( i + " = " + eval("user." +i));
    }
}
```

chooseSecurityPolicy

7.0			
-----	--	--	--

オプションに従ってセキュリティポリシーをフィルタし、その中からユーザが選択できるようにダイアログボックスを表示します。

注意：このメソッドを実行できるのは、バッチャイ vent、コンソールイベント、アプリケーション初期化イベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。このメソッドは UI を表示します。

パラメータ

<code>oOptions</code>	(オプション) 返されるセキュリティポリシーのリストをフィルタするためのパラメータを含む <code>SecurityPolicyOptions</code> オブジェクト。指定しなかった場合は、見つかったすべてのポリシーが表示されます。
-----------------------	---

戻り値

単一の `SecurityPolicy` オブジェクト。ユーザが選択をキャンセルした場合は `null`。

例

ポリシーを選択して、その名前を表示します。

```
var options = { cHandler: security.APSHandler };
var policy = security.chooseSecurityPolicy( options );
console.println("The policy chosen was: " + policy.name);
```

security.APSHandler は、security の定数の 1 つです ([230 ページ](#)を参照)。

exportToFile

6.0			
-----	--	--	--

Certificate オブジェクトを証明書ファイルとしてローカルディスクに書き出します。

注意：書き込むデータは、有効な証明書のデータである必要があります。任意のデータ型を書き込むことはできません。このメソッドは、既存のファイルを上書きしません。

security.[importFromFile](#) も参照してください。

パラメータ

oObject	ディスクに書き出す Certificate オブジェクト。
cDIPath	デバイスに依存しない保存パス。
注意：	cDIPath パラメータは、セーフパスであり (31 ページの「セーフパス」 を参照)、拡張子が .cer であることが必要です。

戻り値

書き込んだファイルのパスを返します（成功した場合）。

例

```
var outPath = security.exportToFile(oCert, "/c/outCert.cer");
```

getHandler

5.0			
-----	--	--	--

SecurityHandler オブジェクトを取得します。getHandler を呼び出すたびに新規エンジンが作成されます。新規エンジンはいくつでも作成可能ですが、UI エンジンは 1 つだけです。

注意：このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

下位互換性：Acrobat 7.0 以降では、Adobe.PPKMS を独立したハンドラとして使用できなくなりました。cName に「Adobe.PPKMS」を指定して getHandler を呼び出すと、Adobe.PPKLite ハンドラのエンジンが返されます。

パラメータ

cName	言語に依存しない、 <code>handlers</code> プロパティによって返されるセキュリティハンドラの名前。 (Acrobat 7.0) Acrobat 7.0 以降では、各ハンドラを表す定数文字列が定義されています。 security の定数 (特に HandlerName オブジェクト) を参照してください。
bUIEngine	(オプション) <code>true</code> の場合は、Acrobat ユーザインターフェイスに関連付けられている既存のセキュリティハンドラのインスタンスが返されます (したがって、例えばユーザは、このユーザインターフェイスを介してログインなどが行えます)。 <code>false</code> (デフォルト) の場合は、新規エンジンが返されます。

戻り値

`cName` で指定した `SecurityHandler` オブジェクト。ハンドラが存在しない場合は、`null` オブジェクトを返します。

例

Adobe.PPKLite SecurityHandler を選択します。

```
// 開くときに署名を検証
security.validateSignaturesOnOpen = true;

// 使用可能なすべての署名ハンドラをリスト
var a = security.handlers;
for (var i = 0; i < a.length; i++)
    console.println("a["+i+"] = "+a[i]);

// UI 用に「Adobe.PPKLite」ハンドラのエンジンを使用
var ppklite = security.getHandler(
    security.PPKLiteHandler, true);
// ログイン
ppklite.login("dps017", "/C/profiles/DPSmith.pfx");
```

この例の続きは、[signatureSign](#) の例を参照してください。

getSecurityPolicies

7.0			
-----	--	--	--

指定のオプションに従ってフィルタした、現在使用可能なセキュリティポリシーのリストを返します。セキュリティポリシーのマスターリストは、フィルタが行われる前に更新されます。最新のポリシーの取得には、デフォルトの `SecurityHandler` オブジェクトが使用されます。使用可能なポリシーがない場合や、フィルタの条件を満たすポリシーがない場合は、`null` を返します。

注意：この関数を呼び出す前に、デフォルトの `SecurityHandler` オブジェクトで `login` を呼び出せば、さらに多くのポリシーを取得できる可能性があります。

これを実行できるのは、コンソールイベントまたはアプリケーション初期化イベントのみです。
Adobe Reader では使用できません。

パラメータ

bUI	(オプション) UI を表示できるかどうかを制御するフラグ。 デフォルト値は <code>false</code> です。
oOptions	(オプション) 返されるセキュリティポリシーのリストをフィルタするためのパラメータを含む <code>SecurityPolicyOptions</code> オブジェクト。指定しなかった場合は、見つかったすべてのポリシーが返されます。

戻り値

`SecurityPolicyOptions` オブジェクトの配列または `null`。

SecurityPolicyOptions オブジェクト

`SecurityPolicyOptions` オブジェクトのプロパティは、次のとおりです。

プロパティ	説明
bFavorites	このプロパティを渡さなかった場合、ポリシーが「優先」かどうかに基づくフィルタは行われません。 <code>true</code> の場合は、「優先」のポリシーのみが返されます。 <code>false</code> の場合は、「優先」でないポリシーのみが返されます。
cHandler	このプロパティを渡さなかった場合、セキュリティフィルタに基づくポリシーのフィルタは行われません。定義した場合は、指定のセキュリティハンドラに一致するポリシーのみが返されます。有効な値は、 <code>security</code> の定数に定義されています (HandlerName オブジェクト を参照)。渡せる値は 1 つのみです。
cTarget	このプロパティを定義しなかった場合、ターゲットに基づくポリシーのフィルタは行われません。定義した場合は、指定のターゲットに一致するポリシーのみが返されます。有効な値は、 <code>security</code> の定数である <code>EncryptTarget</code> オブジェクトに定義されています (621 ページ)。渡せる値は 1 つのみです。

例 1

優先の PPKLite ポリシーのリストを取得し、その名前を表示します。この例では、`security.PPKLiteHandler` を使用しています ([security の定数](#) を参照)。

```
var options = { bFavorites:true, cHandler:security.PPKLiteHandler } ;
var policyArray = security.getSecurityPolicies( { oOptions: options } ) ;
for( var i = 0; i < policyArray.length; i++ )
    console.println( policyArray[i].name );
```

例 2

ログインを行い、APS ポリシーのリストを取得して、その名前を表示します。この例では、`security.APSHandler` を使用しています ([security の定数](#) を参照)。

```
var aps = security.getHandler( security.APSHandler, true );
aps.login();
var options = { cHandler: security.APSHandler };
var policyArray = security.getSecurityPolicies({
  bUI: true,
  oOptions: options });
for(var i = 0; i < policyArray.length; i++)
  console.println( policyArray[i].name );
```

importFromFile

6.0			
-----	--	--	--

データファイルを読み込み、`cType` で指定されているタイプのオブジェクトとして、そのデータを返します。取り込むファイルは、有効な証明書である必要があります。`security.exportToFile` も参照してください。

パラメータ

<code>cType</code>	このメソッドで返すオブジェクトのタイプ。サポートされているタイプは、「Certificate」のみです。
<code>cDIPath</code>	(オプション) <code>bUI</code> が <code>false</code> の場合、このパラメータは必須で、開くファイルをデバイスに依存しないパスで指定します。 <code>bUI</code> が <code>true</code> の場合は、ファイルを開くダイアログボックスのデフォルトパスとして使用されます。
<code>bUI</code>	(オプション) 取り込むファイルを選択するダイアログボックスを表示する場合は <code>true</code> 。デフォルトは <code>false</code> です。
<code>cMsg</code>	(オプション) <code>bUI</code> が <code>true</code> の場合は、ファイルを開くダイアログボックスでタイトルとして使用されます。 <code>cMsg</code> を指定しなかった場合は、デフォルトのタイトルが使用されます。

戻り値

`Certificate` オブジェクト。

例

```
var oMyCert = security.importFromFile("Certificate", "/c/myCert.cer");
```

SecurityHandler

SecurityHandler オブジェクトを使用すれば、署名、暗号化、ディレクトリなどのセキュリティハンドラの機能にアクセスできます。そのプロパティやメソッドは、セキュリティハンドラごとに異なります。ここでは、security オブジェクトのすべてのプロパティおよびメソッドについて説明します。これらのプロパティやメソッドが実装されているかどうかは、securityHandler オブジェクトごとに異なります。

securityHandler オブジェクトを取得するには、`security.getHandler` メソッドを使用します。

Adobe.PPKLite で署名を行うための JavaScript インターフェイスは、Acrobat 5.0 で導入されました。その他の JavaScript インターフェイスは、Acrobat 6.0 で導入されました。6.0 より前のバージョンの Acrobat では、サードパーティのセキュリティハンドラを JavaScript で使用することができませんでした。

現在でも、一部のセキュリティハンドラは JavaScript に対応していません。また、一部のセキュリティ操作に対応していない JavaScript 対応ハンドラもあります。サードパーティの公開鍵セキュリティハンドラでは、JavaScript をサポートしているものもありますが、これらはすべて Acrobat 6.0 で導入された新しい PubSec プログラミングインターフェイスを使用しています。

アドビシステムズ社が提供する JavaScript 対応ハンドラには、次のものがあります。

- Adobe.PPKLite セキュリティハンドラ（署名および暗号化をサポート。Windows オペレーティングシステム版では Microsoft Active Directory Scripting Interface (ADSI) によるディレクトリアクセスもサポート）。
- Adobe.AAB セキュリティハンドラ（ローカルアドレス帳を備え、ディレクトリ操作をサポート）。

注意：文書のパスワード暗号化に使用される標準セキュリティハンドラは、通常は JavaScript に対応していません。ただし、Acrobat 7.0 以降では、定義済みのポリシーを使用して標準セキュリティハンドラによる暗号化が行えます。詳しくは、[encryptUsingPolicy](#) を参照してください。

Acrobat 7.0 以降では、`encryptUsingPolicy` メソッドを介して `Adobe.APS` ハンドラで暗号化を行うこともできます。このハンドラではディレクトリサービスを介してディレクトリを使用可能にすることもできますが、このディレクトリからは証明書が返されないので、一般的な使用には制限があります。

注意：SecurityHandler オブジェクトは、`Security` オブジェクトの `getHandler` メソッドでのみ作成できます。このメソッドを使用できるのは、バッヂイベント、コンソールイベント、アプリケーション初期化イベントのみです（詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照）。Adobe Reader でも使用できます。

SecurityHandler のプロパティ

appearances

5.0			
-----	--	--	--

指定のセキュリティハンドラで使用可能なユーザ設定の外観の、言語に依存する名前を含む配列。外観は、署名フィールドに署名を行ったときのページ上の署名表示を作成するために使用されます。Field オブジェクトの `signatureSign` メソッドを使用して署名フィールドに署名する場合は、`signature info` オブジェクトのプロパティとして、外観の名前を指定することができます。

Acrobat には、標準的な署名の外観が定義されています。この外観は Adobe 署名プラグインで使用されており、サードパーティの署名プラグインでも使用できます。この定義には、あらかじめ外観が 1 つ設定されていますが、ユーザが外観を追加することもできます。設定済みの外観は、ユーザインターフェイスでは標準テキストという名前ですが、この名前はこのプロパティでは返されません。

選択した外観がセキュリティハンドラでサポートされていない場合、このプロパティは `null` を返します。

型

配列

アクセス

R

digitalIDs

6.0			
-----	--	--	--

セキュリティハンドラで現在選択中のデジタル ID に関する証明書。

型

オブジェクト

アクセス

R

戻り値は、次のプロパティを持つ汎用オブジェクトです。

プロパティ	型	バージョン	説明
oEndUserSignCert	Certificate オブジェクト	6.0	現在選択中のデジタル ID に関する証明書で、署名時に SecurityHandler オブジェクトで使用されます。現在何も選択されていない場合、このプロパティは未定義になります。
oEndUserCryptCert	Certificate オブジェクト	6.0	現在選択中のデジタル ID に関する証明書で、SecurityHandler オブジェクトで文書を暗号化するときに使用されます。現在何も選択されていない場合、このプロパティは未定義になります。
certs	Certificate オブジェクトの配列	6.0	この SecurityHandler オブジェクトで使用可能なすべてのデジタル ID のリストに対応する証明書の配列。
stores	文字列の配列	6.0	デジタル ID が保存されているストアを識別するための文字列の配列。Certificate オブジェクトごとに 1 つの文字列が示されます。文字列の値は、セキュリティハンドラによって異なります。Adobe.PPKLite の場合、有効な値は「PKCS12」と「MSCAPI」です。

Adobe.PPKLite セキュリティハンドラは、パスワード保護されたデジタル ID ファイル（PKCS#12 と APF の両方）および Windows の場合は Windows (MCCAPI) ストアにある、現在使用可能なデジタル ID をすべて返します。

`oEndUserSignCert` プロパティや `oEndUserCryptCert` プロパティを設定するには、ユーザインターフェイスを使用します。`oEndUserSignCert` は、`login` メソッドを使用して設定することもできます。つまり、`oEndUserCryptCert` が返されるのは、`bUIEngine` を `true` にして `getHandler` メソッドで取得した `Security Handler` オブジェクトを使用した場合のみです。

例

現在選択中のデジタル ID に関連付けられている証明書を、ファイルに書き出します。

```
var sh = security.getHandler( "Adobe.PPKMS", true );
var ids = sh.digitalIDs;
var oCert = ids.oEndUserSignCert;
security.exportToFile( oCert, "/c/MySigningCert.cer" );
```

directories

6.0			
-----	--	--	--

このセキュリティハンドラで使用可能な `Directory` オブジェクトの配列。新しい `Directory` オブジェクトを作成するには、`newDirectory` メソッドを使用します。

型

配列

アクセス

R

directoryHandlers

6.0			
-----	--	--	--

指定のセキュリティハンドラで使用可能なディレクトリハンドラの、言語に依存しない名前を含む配列。例えば、`Adobe.PPKMS` セキュリティハンドラには、Microsoft ADSI のクエリをサポートする `Adobe.PPKMS.ADSI` という名前のディレクトリハンドラがあります。`info` プロパティを使用して新しい `Directory` オブジェクトをアクティブにするには、有効なディレクトリハンドラ名を指定する必要があります。

型

配列

アクセス

R

docDecrypt

6.0			
-----	--	--	--

セキュリティハンドラで PDF ファイルを復号できる場合は、`true` を返します。

型

ブーリアン

アクセス

R

docEncrypt

6.0			
-----	--	--	--

セキュリティハンドラで PDF ファイルを暗号化できる場合は、`true` を返します。

型

ブーリアン

アクセス

R

isLoggedIn

5.0			
-----	--	--	--

この `securityHandler` オブジェクトに現在ログインしている場合は、`true` を返します。[login](#) メソッドを参照してください。

このプロパティの値は、各セキュリティハンドラ独自のルールによって決定されます。ユーザがプロファイル（証明書ファイルとも呼ばれます。PKCS#12 ファイルとして実装されています）にログインしている場合、`Adobe.PPKLite` ハンドラは `true` を返します。`Adobe.PPKMS` は、常に `true` を返します。

型

ブーリアン

アクセス

R

例

isLoggedIn を使用した一般的なスクリプト。

```
var ppklite = security.getHandler("Adobe.PPKLite", true);
console.println( "Is logged in = " + ppklite.isLoggedIn ); // False
ppklite.login( "dps017", "/C/signatures/DPSmith.pfx");
console.println( "Is logged in = " + ppklite.isLoggedIn ); // True
```

loginName

5.0			
-----	--	--	--

セキュリティハンドラで現在選択中の署名デジタル ID に関する付けられている名前。このプロパティを使用するには、login メソッドを呼び出して、署名証明書を選択する必要があります。署名証明書を選択していないか、セキュリティハンドラでこのプロパティがサポートされていない場合、戻り値は null です。

型

文字列

アクセス

R

loginPath

5.0			
-----	--	--	--

デバイスに依存しない、セキュリティハンドラへのログインに使用されたユーザのプロファイルファイルのパス。ログインしているユーザがない場合や、セキュリティハンドラでこのプロパティがサポートされていない場合、またはこのプロパティが現在ログインしているユーザに無関係な場合は、戻り値は null になります。

型

文字列

アクセス

R

name

5.0			
-----	--	--	--

言語に依存しない、セキュリティハンドラの名前。例えば、デフォルトの証明書ハンドラ、Windows 証明書ハンドラ、Entrust セキュリティハンドラの値は、Adobe.PPKLite、Adobe.PPKMS、Entrust.PPKEF です。すべてのセキュリティハンドラで、このプロパティがサポートされている必要があります。

型

文字列

アクセス

R

signAuthor

6.0			
-----	--	--	--

セキュリティハンドラで証明済み文書を作成可能かどうかを示します。証明済み文書とは、バイト範囲署名およびオブジェクト署名で署名された文書のことです。オブジェクト署名は、文書のオブジェクトツリーを走査して生成される署名で、文書の変更を検出したり防止したりするために使用されます。MDP の設定について詳しくは、[SignatureInfo](#) オブジェクトの `mdp` プロパティを参照してください。

型

ブーリアン

アクセス

R

signFDF

6.0			
-----	--	--	--

セキュリティハンドラで FDF ファイルに署名できるかどうかを示します。

型

ブーリアン

アクセス

R

signInvisible

5.0			
-----	--	--	--

セキュリティハンドラで不可視署名を作成可能かどうかを示します。

型

ブーリアン

アクセス

R

signValidate

6.0			
-----	--	--	--

セキュリティハンドラで署名を検証可能かどうかを示します。

型

ブーリアン

アクセス

R

signVisible

5.0			
-----	--	--	--

セキュリティハンドラで可視署名を作成可能かどうかを示します。

型

ブーリアン

アクセス

R

uiName

5.0			
-----	--	--	--

言語に依存する、セキュリティハンドラの文字列。この文字列は、ユーザインターフェイスでの使用に適しています。すべてのセキュリティハンドラで、このプロパティがサポートされている必要があります。

型

文字列

アクセス

R

validateFDF

6.0			
-----	--	--	--

セキュリティハンドラで FDF ファイルを介して署名を検証可能な場合は、`true` を返します。

型

ブーリアン

アクセス

R

SecurityHandler のメソッド

login

5.0			
-----	--	--	--

特定のセキュリティハンドラでデジタル ID にアクセスして選択するためのメカニズムを提供します。ユーザインターフェイスを使用し、デフォルトのデジタル ID を永続的なものとして、またはアプリケーションの実行中のみ有効なものとして、選択できます。UI を使用してそのような選択をした場合は、デジタル ID を使用する前にセキュリティハンドラにログインする必要がないこともあります。

多くの場合、パラメータは特定のハンドラに対して固有のものになります。Adobe.PPKLite ハンドラと Adobe.PPKMS ハンドラのパラメータは、次のようにになります。

`cPassword` パラメータと `cDIPPath` パラメータは、下位互換性を維持するために提供されていますが、`oParams` オブジェクトのプロパティに含めることもできます。Acrobat 6.0 以降で推奨されている呼び出し規則は、後者の方法です。

[logout](#)、[newUser](#)、[loginName](#) も参照してください。

パラメータ

<code>cPassword</code>	(オプション、Acrobat 5.0) パスワード保護されたデジタル ID にアクセスする場合に必要なパスワード。このパラメータは、デジタル ID ファイルと PKCS#11 デバイスにアクセスできるように Adobe.PPKLite でサポートされています。
<code>cDIPPath</code>	(オプション、Acrobat 5.0) デバイスに依存しない、パスワード保護されたデジタル ID ファイルまたは PKCS#11 ライブラリへのパス。このパラメータは、Adobe.PPKLite でサポートされています。

oParams	(オプション、Acrobat 6.0) 特定の SecurityHandler オブジェクトに固有のパラメータを持つ LoginParameters オブジェクト。このオブジェクトの一般的なフィールドは、次に示すとおりです。これらのフィールドには、cDIPath や cPassword が含まれており、パラメータリストを異なる形式で表すことができます。
bUI	(オプション、Acrobat 6.0) ユーザインターフェイスを使用してログインする必要がある場合は、true に設定します。false に設定すると、ユーザインターフェイスに関連付けられているデフォルトのエンジンは使用されず、セキュリティ設定ダイアログボックスにデジタル ID のログイン状態は表示されません。この属性は、このメソッドをサポートするすべてのセキュリティハンドラでサポートする必要があります。

戻り値

正常にログインすると true を返し、それ以外の場合は false を返します。

LoginParameters オブジェクト

この汎用 JavaScript オブジェクトには、login メソッドのパラメータが含まれています。このオブジェクトのプロパティは、次のとおりです。

プロパティ	型	バージョン	説明
cDIPath	文字列	5.0	デジタル ID または PKCS#11 ライブラリを含むファイルのパス。Adobe.PPKLite セキュリティハンドラでサポートされています。
cPassword	文字列	6.0	ユーザ認証に使用するパスワード。このパスワードは、パスワードで保護されたデジタル ID ファイルまたは PKCS#11 デバイスにアクセスする場合に使用します。Adobe.PPKLite セキュリティハンドラでサポートされています。Acrobat では、メモリに読み込んだパスワードの隠蔽処理などは行わないで、注意してください。
cPFX	文字列	7.0	(オプション) ログインするための 16 進エンコードされた PFX。このパラメータを指定すると、cDIPath よりも優先されます。 注意： このパラメータは、内部的な使用に限定されています。現在、16 進エンコードされた PFX ファイルを、JavaScript を使用して取得する方法はありません。
oEndUserSignCert	汎用オブジェクト	6.0	エンドユーザーの署名を実行するためのデジタル ID を選択します。このプロパティの値は Certificate オブジェクト（または Certificate オブジェクトと同じプロパティ名を持つ汎用オブジェクト）であり、選択中の証明書を示します。 このメソッドを呼び出す必要があるかどうかは、ハンドラによって異なります。例えば、Adobe.PPKLite を使用して、デジタル ID 署名が 1 つ設定された PKCS#12 ファイルにログインした場合は、署名証明書を選択する必要はありません。すべてのセキュリティハンドラには、このオブジェクトのバイナリおよび SHA1Hash プロパティを処理する機能が必要です。bUI が true の場合は、このオブジェクトは空にできます。

プロパティ	型	バージョン	説明
cMsg	文字列	6.0	bUI が true の場合にログインダイアログボックスに表示するメッセージ。
cURI	文字列	7.0	サーバへの接続に使用する URI。Adobe.APS ハンドラと Adobe.PPKLite ハンドラでサポートされています。Adobe.PPKLite の場合は、URI にローミングクレデンシャルサーバを指定します。
cUserId	文字列	7.0	サーバへの接続時に使用するユーザ名。Adobe.APS ハンドラと Adobe.PPKLite ハンドラでサポートされています。
cDomain	文字列	7.0	サーバへの接続時に使用するドメイン名。Adobe.APS ハンドラでのみサポートされています。
iSlotID	整数	7.0	ログインする PKCS#11 デバイスのスロット ID を指定します。このパラメータは、Adobe.PPKLite ハンドラでのみサポートされています。
cTokenLabel	文字列	7.0	ログインする PKCS#11 デバイスのトーカンラベルを指定します。このパラメータは、Adobe.PPKLite ハンドラでのみサポートされています。

例 1

ログインしてログアウトします。

```
// UI 用の「Adobe.PPKLite」セキュリティハンドラオブジェクトを使用
var ppklite = security.getHandler( security.PPKLiteHandler, true );
var oParams = { cPassword: "dps017", cDIPath: "/C/DPSmith.pfx" }
ppklite.login( oParams );
<..... 署名フィールドを作成して署名 .....,>
ppklite.logout();
```

ログインしたら、UI を使用して証明書を選択

```
ppklite.login(
{ oParams:
  { oEndUserSignCert: {},
    cMsg: "Select your Digital ID" },
  bUI : true
} );
```

ログインして署名証明書を選択

```
var oCert = { SHA1Hash: "00000000" };
ppklite.login(
{ oParams:
  { cDIPath: "/C/test/DPSmith.pfx",
    cPassword: "dps017",
    oEndUserSignCert: oCert,
    cMsg: "Select your Digital ID"
  },
  bUI : true
});
```

例 2

「Adobe.PPKMS」セキュリティハンドラオブジェクトを使用し、署名時に使用する証明書を選択します。

```
var ppkms = security.getHandler( "Adobe.PPKMS" );
var oCert = myCerts[0];
ppkms.login( { oParams: { oEndUserSignCert: oCert } } );
```

例 3

security.APSHandler を使用します。[security の定数](#)を参照してください。

```
var aps = security.getHandler( security.APSHandler, true );
var oParams = { cUserName: "Acrobat", cPassword: "adobedon" };
aps.login( oParams );
<..... このハンドルとポリシー ID を使用して文書を暗号化 .....>
aps.logout();
```

PDF 文書の署名について詳しくは、[signatureSign](#) を参照してください。

logout

5.0			
-----	--	--	--

SecurityHandler オブジェクトをログアウトします。このメソッドは Adobe.PPKMS ではなく、Adobe.PPKLite で使用されます。

[login](#) メソッドも参照してください。

戻り値

Acrobat 6.0 以降では、正常にログアウトすると `true` を返し、それ以外の場合は `false` を返します。以前のリリースの Acrobat では、戻り値が生成されませんでした。

newDirectory

5.0			
-----	--	--	--

新しい [Directory](#) オブジェクトを返します。このオブジェクトを永続化して、検索に使用できるようにするには、`info` プロパティを使用して、このオブジェクトをアクティブにする必要があります。既存の `directory` オブジェクトは、[directories](#) プロパティを使用して検出できます。

戻り値

新しい Directory オブジェクト

newUser

5.0			
-----	--	--	--

新しい Self-Sign 証明書を作成して、Adobe.PPKLite および Adobe.PPKMS セキュリティハンドラにそのユーザを登録します。

注意：このメソッドを使用しても、既存ファイルは上書きされません。

パラメータ

cPassword	(オプション) パスワード保護されたデジタル ID ファイルにアクセスする場合に必要なパスワード。このパラメータは、Adobe.PPKMS では無視されます。
cDIPath	(オプション) デバイスに依存しない、パスワード保護されたデジタル ID ファイルのパス。このパラメータは、Adobe.PPKMS では無視されます。 注意： Acrobat 6.0 以降では、cDIPath パラメータはセーフパスであり (31 ページの「セーフパス」 を参照)、拡張子が .pfx であることが必要です。
oRDN	(オプション) 相対識別名 (RDN)。これは、証明書の発行者名または署名者名を含む RDN オブジェクト (587 ページ) です。必須フィールドは、cn のみです。国名 c を指定する場合は、ISO 3166 規格に準拠した 2 文字 ('US' など) を使用してください。
oCPS	(オプション、Acrobat 6.0) 証明書の Certificate Policy エクステンションに埋め込まれる証明書ポリシー情報を含む汎用オブジェクト。このオブジェクトには、次のプロパティがあります。 <ul style="list-style-type: none">oid は必須のプロパティで、証明書ポリシーオブジェクト識別子を示します。url (オプション) は、証明書の発行基準となるポリシーの詳細を示す URL です。notice (オプション) は、証明書に埋め込まれた情報の要約版です。
bUI	(オプション、Acrobat 6.0) true の場合は、ユーザインターフェイスを使用して登録できます。このパラメータは、このメソッドをサポートするすべてのセキュリティハンドラでサポートされています。
cStore	(オプション、Acrobat 7.0) 生成された証明書を保存するストアを識別する文字列。Adobe.PPKLite セキュリティハンドラの場合、有効なストア識別子は「PKCS12」と「MSCAPI」です。このパラメータが省略されて cDIPath が指定されている場合、生成された証明書は、PKCS#12 ファイルに保存されます。それ以外の場合は、CAPI ストアに保存されます。

戻り値

成功した場合は true を返し、失敗した場合は例外が発生します。

例 1

新しい PPKLite Self-Sign 証明書を作成します（Acrobat 5.0 の構文）

```
var ppklite = security.getHandler(security.PPKLiteHandler);
var oORDN = { cn: "Fred NewUser", c: "US" };
var oCPS = {oid: "1.2.3.4.5",
    url: "http://www.example.com/mycps.html",
    notice: "This is a self generated certificate, hence the "
        + "recipient must verify it's authenticity through an out "
        + "of band mechanism" };
ppklite.newUser( "testtest", "/d/temp/FredNewUser.pfx", oORDN, oCPS);

// 汎用オブジェクト構文による別の書き方。追加のパラメータが使用可能
var oParams = {
    cPassword : "myPassword",
    cDIPath : "/d/temp/FredNewUser.pfx",
    oORDN,
    oCPS,
    bUI : false
};
ppklite.newUser( oParams );
```

例 2

既存の署名済みフィールドの証明書を使用して RDN を作成します

```
var f = this.getField( "mySignature" );
f.signatureValidate();
var sigInfo = f.signatureInfo();
var certs = sigInfo.certificates;
var oSubjectDN = certs[0].subjectDN;

ppklite.newUser({
    cPassword: "dps017",
    cDIPath: "/c/temp/DPSmith.pfx",
    oORDN: oSubjectDN
});
```

setPasswordTimeout

5.0			
-----	--	--	--

署名後パスワードが期限切れになるまでの秒数を設定します。このメソッドは、Adobe.PPKLite セキュリティハンドラでのみサポートされています。このハンドラの場合、新規ユーザのデフォルトタイムアウト値は 0（パスワードが常に必要）になります。

パラメータ

cPassword	タイムアウト値を設定するために必要なパスワード。
iTimeout	タイムアウト値（秒）。常に期限切れ（パスワードが常に必要）にするには、0 に設定します。期限切れが発生しないようにするには、0xFFFFFFFF に設定します。

戻り値

ユーザが Adobe.PPKLite セキュリティハンドラにログインしていなかったり、その他の理由によって失敗した場合は、例外が発生します。

例

この例では、PPKLite セキュリティハンドラにログインし、パスワードのタイムアウトを 30 秒に設定します。パスワードのタイムアウト時間（この場合は 30 秒）が経過すると、署名者は次に署名をする際、パスワードを入力する必要があります。パスワードが切れていなければ、パスワードを入力する必要はありません。

```
var ppklite= security.getHandler( "Adobe.PPKLite" );
ppklite.login( "dps017", "/d/profiles/DPSmith.pfx" );
ppklite.setPasswordTimeout( "dps017", 30 );
```

SecurityPolicy

7.0			
-----	--	--	--

Security Policy オブジェクトは、文書に暗号化を適用するために使用するセキュリティ設定のグループを表しています。このオブジェクトは、`getSecurityPolicies` と `chooseSecurityPolicy` の両方の戻り値として取得します。

SecurityPolicy のプロパティ

プロパティ	型	アクセス	説明
policyID	文字列	R	セキュリティポリシーを一意に識別するために生成される文字列。人間が読むことを意図して生成されたものではありません。 新しく作成された SecurityPolicy オブジェクトでこのプロパティが既知のポリシー ID に設定されると、このポリシーを使用するすべてのメソッドは、ポリシーの適用前に正しいセキュリティ設定を必ず取得することになります。
name	文字列	R	UI で使用されるポリシー名。ローカライズが可能です。
description	文字列	R	UI で使用されるポリシーの説明。ローカライズが可能です。
handler	文字列	R	このセキュリティポリシーを実装するセキュリティハンドラを表す列挙値。有効な値は、 620 ページの「security の定数」 の HandlerName オブジェクトに定義されています。
target	文字列	R	暗号化するターゲットデータを表す列挙値。有効な値は、 620 ページの「security の定数」 の EncryptTarget オブジェクトに定義されています。

SignatureInfo

電子署名のプロパティを含む汎用 JavaScript オブジェクト。すべてのハンドラでサポートされているプロパティと、ハンドラ固有のプロパティで構成されます。

SignatureInfo オブジェクトは、Field オブジェクトの `signatureValidate` メソッドや `signatureInfo` メソッドによって返され、FDF オブジェクトの `signatureSign` メソッドや `signatureValidate` メソッドに渡します。

書き込み可能なプロパティは、オブジェクトの署名時に指定することができます。

SignatureInfo のプロパティ

次のプロパティは、すべてのハンドラで定義されています。

SignatureInfo オブジェクトのプロパティ

プロパティ	型	アクセス	バージョン	説明
<code>buildInfo</code>	オブジェクト	R	6.0	署名のソフトウェアビルド情報とバージョン情報を含むオブジェクト。このオブジェクトの形式については、Adobe Solutions Network の Web サイトにある『PDF Signature Build Dictionary Specification』というテクニカルノートを参照してください。
<code>date</code>	日付オブジェクト	R	5.0	署名が作成された日時が、JavaScript の Date オブジェクトとして返されます。
<code>dateTrusted</code>	ブーリアン	R	7.0	<code>date</code> のソースが信頼できるかどうかを示すブーリアン値。この値が存在しない場合、 <code>date</code> のソースは信頼できないと考えられます（署名者のコンピュータのシステム時間など）。
<code>digestMethod</code>	文字列	R / W	8.0	署名のダイジェスト方式。Acrobat 8 の場合、有効な値は SHA1、SHA256、SHA384、SHA512、RIPEMD160 です。このプロパティを呼び出す場合は、どの署名ハンドラでどのダイジェスト方式がサポートされているかを理解しておく必要があります。
<code>handlerName</code>	文字列	R	5.0	署名ディクショナリの Filter 属性で指定されたセキュリティハンドラの、言語に依存しない名前。この名前は、通常、署名を作成したセキュリティハンドラの名前になりますが、署名を検証するときに使用するセキュリティハンドラの名前になることもあります。

SignatureInfo オブジェクトのプロパティ

プロパティ	型	アクセス	バージョン	説明
handlerUserName	文字列	R	5.0	handlerName で示されたセキュリティハンドラに対応する、言語に依存する名前。このプロパティは、そのセキュリティハンドラが使用可能な場合にのみ使用できます。
handlerUIName	文字列	R	5.0	handlerName で示されたセキュリティハンドラに対応する、言語に依存する名前。このプロパティは、そのセキュリティハンドラが使用可能な場合にのみ使用できます。
location	文字列	R / W	5.0	署名時にユーザが入力した場所（オプション）。物理的な場所（都市など）やホスト名を設定できます。
mdp	文字列	R / W	6.0	フィールドの署名や FDF オブジェクトの署名に使用された MDP 設定か、または署名時に使用する MDP 設定。有効な値は、次のとおりです。 allowNone allowAll default defaultAndComments 詳しくは、 653 ページの「MDP 値」 を参照してください。デフォルトの allowAll の場合は、署名に MDP が使用されないので、証明用の署名にはなりません。
name	文字列	R	5.0	署名を作成したユーザの名前。
numFieldsAltered	数値	R	5.0 のみ	非推奨になりました。前回の署名と今回の署名の間に変更されたフィールド数。署名フィールドでのみ使用されます。 Acrobat 7.0 以降では、Field オブジェクトの signatureGetModifications メソッドの機能を使用してください。

SignatureInfo オブジェクトのプロパティ

プロパティ	型	アクセス	バージョン	説明
numFieldsFilledIn	数値	R	5.0 のみ	非推奨になりました。前回の署名と今回の署名の間に記入されたフィールド数。署名フィールドでのみ使用されます。 Acrobat 7.0 以降では、Field オブジェクトの signatureGetModifications メソッドの機能を使用してください。
numPagesAltered	数値	R	5.0 のみ	非推奨になりました。前回の署名と今回の署名の間に変更されたページ数。署名フィールドでのみ使用されます。 Acrobat 7.0 以降では、Field オブジェクトの signatureGetModifications メソッドの機能を使用してください。
numRevisions	数値	R	5.0	文書のリビジョン数。署名フィールドでのみ使用されます。
reason	文字列	R / W	5.0	ユーザ指定の署名理由。
revision	数値	R	5.0	署名フィールドが対応している署名のリビジョン。署名フィールドでのみ使用されます。
sigValue	文字列	R	7.0	署名のバイナリデータ。16進エンコードされた文字列で示されます。
status	数値	R	5.0	signatureValidate の前回の呼び出しで計算された、署名の完全性のステータス。 status プロパティの戻り値については、 651 ページの「status プロパティと idValidity プロパティ」 の表を参照してください。
statusText	文字列	R	5.0	signatureValidate の前回の呼び出しで計算された署名の完全性のステータスを表す、言語に依存するテキスト文字列。ユーザへの表示に適しています。

SignatureInfo オブジェクトのプロパティ

プロパティ	型	アクセス	バージョン	説明
subFilter	文字列	R / W	6.0	署名時に使用する形式。サポートされている値の完全なリストについては、『PDF Reference』バージョン 1.7 を参照してください。公開鍵署名に使用される既知の値は、adbe.pkcs7.sha1、adbe.pkcs7.detached、adbe.x509.rsa_sha1 などです。このプロパティを呼び出す場合は、どの電子署名ハンドラでどの形式がサポートされているかを理解しておく必要があります。
timeStamp	文字列	W	7.0	署名にタイムスタンプを付けるためのサーバの URL タイムスタンプの取得でサポートされるスキームおよびトランスポーテートプロトコルは、http と https のみです。 このプロパティは書き込み専用です。署名にタイムスタンプが設定されると、確認時に dateTrusted プロパティが true に設定されます（タイムスタンプ署名が信頼できる場合）。また、verifyDate の値と署名の日付が同じになります。
verifyDate	日付オブジェクト	R	7.0	署名が確認された日時が、JavaScript の Date オブジェクトとして返されます（署名が確認された場合）。
verifyHandlerName	文字列	R	6.0	署名の検証に使用されたセキュリティハンドラの、言語に依存しない名前。署名が検証されなかった場合（status プロパティの値が 1 の場合）は、null になります。
verifyHandlerUIName	文字列	R	6.0	verifyHandlerName で示されたセキュリティハンドラに対応する、言語に依存する名前。署名が検証されなかった場合（status プロパティの値が 1 の場合）は、null になります。

SignatureInfo オブジェクトの公開鍵セキュリティハンドラのプロパティ

公開鍵セキュリティハンドラでは、次のプロパティを定義することができます。

プロパティ	型	アクセス	バージョン	説明
appearance	文字列	W	5.0	フィールドに署名する場合に使用する、ユーザ設定の外観名。PPKLite と PPKMS では、標準外観ハンドラが使用されます。この場合、外観名は「セキュリティ」ユーザ環境設定の、署名の表示方法の設定ダイアログボックスに表示されます。指定しない場合のデフォルトでは、標準テキスト外観が使用されます。可視の署名フィールドでのみ使用されます。
certificates	配列	R	5.0	署名者を識別する証明書の階層を含む配列。配列の最初の要素は、署名者の証明書です。以降の要素は、署名者の証明書を発行した認証機関までの証明書チェーンです。Self-Sign 証明書の場合、この配列にはエントリが 1 つのみ含まれます。
contactInfo	文字列	R / W	5.0	信頼性の確認に使用できるユーザ指定の連絡先情報。例えば、電話番号などを入力しておけば、文書の受信者は作成者と連絡を取ることができます。ただし、信頼確立のために拡張性のあるソリューションが必要な場合は、この方法はお勧めしません。
byteRange	配列	R	6.0	この署名によってカバーされるバイトを示す、数字の配列。
docValidity	数値	R	6.0	signatureValidate の前回の呼び出しで計算された、その署名がカバーするバイト範囲のダイジェスト値の完全性のステータス。すべての署名フィールドの署名には、バイト範囲ダイジェストが含まれています。 戻りコードについて詳しくは、 651 ページの「検証値」 を参照してください。
idPrivValidity	数値	R	6.0	署名者の ID の完全性。この値は、ハンドラ固有です。Adobe.PPKLite および Adobe.PPKMS ハンドラでサポートされる値については、 652 ページの「プライベート検証値」 を参照してください。 署名が検証されなかった場合 (status プロパティの値が 1 の場合) は、この値は 0 です。

プロパティ	型	アクセス	バージョン	説明
idValidity	数値	R	6.0	<p>署名者の ID の完全性を表す数値。</p> <p><code>idValidity</code> プロパティの戻り値については、651 ページの「status プロパティと idValidity プロパティ」 の表を参照してください。</p>
objValidity	数値	R	6.0	<p><code>signatureValidate</code> の前回の呼び出しで計算された、署名のオブジェクトダイジェスト部分の完全性のステータス。PDF 文書の場合、署名フィールドへの証明用の署名と文書レベルのアプリケーション権限の署名には、オブジェクトダイジェストが含まれています。すべての FDF ファイルは、オブジェクトダイジェストを使用して署名されます。</p> <p>戻りコードについて詳しくは、651 ページの「検証値」 を参照してください。</p>
revInfo	オブジェクト	R	7.0	<p>CRL と OCSP の 2 つのプロパティを含む汎用オブジェクト。これらのプロパティは、16 進エンコードされた文字列の配列です。各文字列には、証明書の失効確認を行うために使用された失効情報がバイナリの状態で含まれています。</p> <p>CRL プロパティの文字列は CRL を表します。 OCSP プロパティの文字列は OCSP レスポンスを表します。このデータは非常に大きくなる可能性があるので、設定を行うようにアプリケーションの環境設定で指定されている場合を除き、これらのプロパティの設定は行われません。</p>
trustFlags	数値	R	6.0	<p>この数値ビットは、署名者が信頼される対象を示します。この値は、<code>status</code> プロパティ値が 4 の場合にのみ有効です。これらの信頼設定は、Acrobat Address Book (Adobe.AAB) など、受信者の信頼データベースの信頼設定から取得されます。ビット割り当ては、次のとおりです。</p> <ul style="list-style-type: none"> 1 — 署名に対して信頼されます 2 — 文書の証明に対して信頼されます 3 — マルチメディアなどのダイナミックコンテンツに対して信頼されます 4 — Adobe 内部で使用されます 5 — 通常の PDF 制限に従わずに操作する場合に、PDF ファイル内の JavaScript を信頼します
password	文字列	W	5.0	署名に使用する秘密鍵にアクセスする際の認証に必要なパスワード。このプロパティが必要であるかどうかは、セキュリティハンドラのポリシーによって異なります。

status プロパティと idValidity プロパティ

SignatureInfo オブジェクトの `status` プロパティと `idValidity` プロパティが返すコードのリストを次の表に示します。

ステータス コード	説明
-1	署名フィールドではありません。
0	署名が空白であるか、または署名されていません。
1	不明なステータスです。これは、署名が検証されていない場合に発生します（例えば、ネットワーク接続で文書の一部のみがダウンロードされている場合など）。
2	署名が無効です。
3	署名は有効ですが、署名者の ID が検証できません。
4	署名が有効で、署名者の ID も有効です。

検証値

`docValidity` プロパティと `objValidity` プロパティの戻りコードは、次のとおりです（[649 ページの「SignatureInfo オブジェクトの公開鍵セキュリティハンドラのプロパティ」](#)を参照）。この戻りコードでは、署名の完全性について、`status` プロパティよりもさらに細かい情報が提供されます。

ステータスコード	説明
<code>kDSSigValUnknown</code>	完全性は決定されていません。
<code>kDSSigValUnknownTrouble</code>	検証プロセス中にエラーが発生したので、完全性を決定できませんでした。
<code>kDSSigValUnknownBytesNotReady</code>	ファイルを Web ブラウザで表示中などの理由により使用できないバイトがあるため、完全性を決定できませんでした。すぐに使用できないバイトがある場合でも、基本実装がビジー状態になるなどして処理が間に合わないときには、この値が返されないことがあります。Adobe では完全性チェックのビジー状態に関する話題は言及しません。ただし、Acrobat 6.0 の実装では、 <code>docValidity</code> の検証中はビジー状態になり、 <code>objValidity</code> の検証中はビジー状態になりません。
<code>kDSSigValInvalidTrouble</code>	署名の形式または情報にエラーが存在したので、ダイジェストの完全性が計算されませんでした。署名が無効であることを示す十分な理由が見つかりました。
<code>kDSSigValInvalid</code>	このダイジェストの完全性が使用されません（バイト範囲がなく、文書が検証されないなど）。
<code>kDSSigValJustSigned</code>	署名されたばかりであり、暗黙的に有効です。
<code>kDSSigValFalse</code>	ダイジェストまたは完全性が無効です。
<code>kDSSigValTrue</code>	ダイジェストまたは完全性が有効です。

プライベート検証値

署名者 ID の完全性の検証は、ID の検証に使用しているハンドラに固有のものです。この値には、ID に関する有益な情報が含まれていることがあります。ID は、`idPrivValidity` プロパティに格納されて返されます。Adobe.PPKMS および Adobe.PPKLite セキュリティハンドラの値は、次のとおりです。この値は、すべてのハンドラに共通の `idValidity` 値にもマッピングされます。

ステータスコード	<code>idValidity</code> マッピング	セキュリティ ハンドラ	説明
<code>kIdUnknown</code>	1 (未知)	PPKMS、 PPKLite	完全性は決定されていません。
<code>kIdTrouble</code>	1 (未知)	PPKMS、 PPKLite	内部エラーなどのエラーにより完全性を決定できないか、チェーンを構築できないか、または基本ポリシーをチェックできませんでした。
<code>kIdInvalid</code>	2 (無効)	PPKMS、 PPKLite	証明書が無効です。時間がネストされていないか、署名が無効であるか、制約が無効／サポートされていないか、エクステンションが無効であるか、またはチェーンが循環しています。
<code>kIdNotTimeValid</code>	2 (無効)	PPKMS、 PPKLite	証明書が有効期限の範囲外です（早すぎるか、または遅すぎます）。
<code>kIdRevoked</code>	2 (無効)	PPKMS	証明書が失効しています。
<code>kIdUntrustedRoot</code>	1 (未知)	PPKMS、 PPKLite	証明書に信頼されていないルート証明書が含まれています。
<code>kIdBrokenChain</code>	2 (無効)	PPKMS、 PPKLite	Self-Sign ルート証明書までの証明書チェーンを構築できませんでした。
<code>kIdPathLenConstraint</code>	2 (無効)	PPKLite	証明書チェーンが指定された長さの制限を超えていました。この制限は、チェーン内のいずれかの証明書の基本制約エクステンションで指定されています。
<code>kIdCriticalExtension</code>	1 (未知)	PPKMS	チェーン内の証明書のいずれかに、認識されない重要なエクステンションが含まれています。
<code>kIdJustSigned</code>	4 (有効)	PPKMS、 PPKLite	ユーザのみによって署名されています (<code>kIdIsSelf</code> と同様)。
<code>kIdAssumedValid</code>	3 (未知の ID)	PPKMS	証明書は信頼済みルートに対して有効です。ただし、失効確認は行われず、要求もありませんでした。
<code>kIdIsSelf</code>	4 (有効)	PPKMS、 PPKLite	証明書はチェックを行っているユーザのものです（それ以上のチェックは行われませんでした）。

ステータスコード	idValidity マッピング	セキュリティ ハンドラ	説明
kIdValid	4 (有効)	PPKMS、 PPKLite	証明書は信頼済みルートに対して有効です (Windows または Acrobat アドレス帳内)。
kIdRevocationUnknown	?	PPKMS、 PPKLite	証明書は信頼済みルートに対して有効です。ただし、ユーザから要求された失効確認は行われませんでした。

MDP 値

MDP は、署名の有効性を維持しながら、文書に対して行うことができる変更を制御します。変更内容は、署名フィールドのバイト範囲の外部に記録されます。ここに、追加保存されていく変更を含めたり、文書を開いてから署名を検証するまでの間にメモリ内で発生した変更を記録したりできます。MDP 設定は、文書内の最初の署名にのみ適用できます。MDP を使用すると、証明用の署名が作成されます。MDP 値は、次の 4 つの値のいずれかになります。

allowAll — 文書に変更を加えても、署名は無効になりません。この場合、MDP は署名で使用されません。これは、Acrobat 4.0 ~ 5.1 の動作です。

allowNone — 文書を変更すると、署名が無効になります。これにより、作成者の署名もロックされます。

default — 文書にフォームフィールドが存在する場合は、フォームフィールドに入力できます。それ以外の変更を行うと、署名が無効になります。

defaultAndComments — フォームフィールドへの入力（文書にフォームフィールドが存在する場合）と、注釈（コメント）の追加、削除、変更が許可されます。それ以外の変更を行うと、署名が無効になります。注釈によって、文書の外観に影響を与えるような変更（例えば、文書の一部を隠すなど）が行われることもあるので、注意してください。

SOAP

SOAP オブジェクトを使用すると、JavaScript からリモートサーバに、リモートプロシージャコールや XML メッセージを送信することができます。

SOAP 1.1 プロトコル (<http://www.w3.org/TR/SOAP/> を参照) では、JavaScript パラメータを集めてリモートプロシージャコールに（同期または非同期に）渡し、結果を再び集めて、JavaScript オブジェクトに戻すことができます。SOAP オブジェクトには、Web Services Description Language (WSDL — <http://www.w3.org/TR/wsdl> を参照) に記載されている Web サービスと通信する機能もあります。

注意：SOAP の connect、request、response の各メソッドが使用できるのは、Acrobat Professional および Acrobat Standard で開かれた文書と、Adobe Reader 6.0 以降で開かれたフォームの書き出し権限が付与されている文書のみです。

SOAP のプロパティ

wireDump

6.0			
-----	--	--	--

true の場合は、同期 SOAP リクエストを行うと、XML のリクエストやレスポンスが JavaScript コンソールに表示されます。これは、SOAP の問題をデバッグする場合に役立ちます。

注意：Acrobat 8.0 以降では、このプロパティは非推奨になりました。新しいサービスでは Net.SOAP.wireDump ([542 ページ](#)を参照) を使用してください。

型

ブーリアン

アクセス

R / W

SOAP のメソッド

connect

6.0			
-----	--	--	---

WSDL 文書の URL (cURL) を、Web サービスから呼び出し可能なメソッドを持つ JavaScript オブジェクトに変換します。

このメソッド呼び出しのパラメータや戻り値は、SOAP.request メソッドで指定されているルールに従います。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは Net.SOAP.connect ([542 ページ](#)を参照) を使用してください。

パラメータ

cURL	WSDL 文書の URL。HTTP か HTTPS の URL である必要があります。
------	---

戻り値

JavaScript メソッドを含む WSDL サービスプロキシオブジェクト。含まれるメソッドは、URL で指定された WSDL 文書内の各操作に相当します。

メソッドに必要なパラメータは、呼び出す WSDL 操作や、その操作がパラメータをエンコードする方法によって異なります。

- （SOAP 1.1 仕様の第 7 節で説明されているように）WSDL 操作が SOAP RPC エンコードを使用する場合、そのサービスメソッドの引数は WSDL 文書内のパラメータ順序と同じです。
- WSDL サービスが SOAP document/literal エンコードを使用する場合、その関数はリクエストメッセージを示す 1 つの引数を取ります。引数は、メッセージを表す JavaScript オブジェクトリテラルの場合もあれば、メッセージを表す XML フラグメントを持つ文字列、または ReadStream オブジェクトの場合もあります。サービスメソッドの戻り値は、WSDL 操作の戻り値に相当します。

各 Web サービスマソッドに対応する JavaScript 関数オブジェクトでは、次のプロパティが（設定されていれば）使用されます。デフォルトでは、いずれのプロパティも設定されていません。

プロパティ	説明
asyncHandler	Web サービスマソッドを非同期で実行することを示しています。このプロパティは、SOAP.request の oAsync パラメータに相当します。
requestHeader	Web サービスマソッドのリクエストに SOAP ヘッダを含めることを示しています。このプロパティは、SOAP.request の oReqHeader パラメータに相当します。
responseHeader	Web サービスマソッドのレスポンスで SOAP ヘッダを返すことを示しています。このプロパティは、SOAP.request の oRespHeader パラメータに相当します。
authenticator	Web サービスマソッドの認証の処理方法を示しています。このプロパティは、SOAP.request の oAuthenticate パラメータに相当します。

例外

SOAP Fault によって SOAPError が発生します。エンドポイントが使用できないなど、ネットワークレベルで問題がある場合は、NetworkError が発生します。詳しくは、[request](#) メソッドを参照してください。

例

エコーサービスの WSDL 文書を使用して、文字列と整数をエコーします。

サービス WSDL 文書の URL が必要です。これらの URL は、次の URL の「Round 2 Interop Services - using SOAP 1.2」セクションから取得できます。<http://www.whitemesa.com/interop.htm>

```
var cURL = <get a URL for this service from  
          http://www.whitemesa.com/interop.htm>;  
  
// テストサービスに接続  
var service = SOAP.connect(cURL);  
  
// このサービスがサポートするメソッドをコンソールに表示  
for(var i in service) console.println(i);  
  
var cTestString = "This is my test string";  
  
// echoString サービスを呼び出す。これは RPC エンコードのメソッド  
var result = service.echoString(cTestString);  
  
// この値は cTestString と等しくなる  
console.println(result + " == " + cTestString);  
  
// echoInteger サービスを呼び出す。JavaScript では整数がサポート  
// されていないので、独自の整数オブジェクトを作成  
var oTestInt =  
{  
    soapType: "xsd:int",  
    soapValue: "10"  
};  
var result = service.echoInteger(oTestInt);  
  
// この値は oTestInt.soapValue と等しくなる  
console.println(result + " == " + oTestInt.soapValue);
```

出力は次のようにになります。

```
echoBase64  
echoBoolean  
echoDate  
echoDecimal  
echoFloat  
echoFloatArray  
echoHexBinary  
echoInteger  
echoIntegerArray  
echoPolyMorph  
echoPolyMorphArray  
echoPolyMorphStruct  
echoString  
echoStringArray  
echoStruct  
echoStructArray  
echoVoid  
This is my test string == This is my test string  
10 == 10
```

queryServices

7.0			
-----	--	--	--

自分自身をパブリッシュしたネットワークサービスを、DNS Service Discovery (DNS-SD) を使用して見つけます。このメソッドを使用すれば、ローカルネットワークリンク内ではマルチキャスト DNS (mDNS) を使用して、企業内ではユニキャスト DNS を使用して、登録済みのサービスの場所を検索できます。サービスの場所の検索結果は常に非同期に返されます。クエリは停止されるまで継続します。これらのサービスが使用可能になったり使用不能になったりしたときには通知されます。

サービスに対するクエリの結果は一連のサービス名です。必要に応じて `resolveService` を呼び出せば、これらのサービス名をバインドできます。

サービスの検索は、ローカルネットワークリンク内でサードパーティの mDNS レスポンダを使用して行うことも、企業ネットワーク環境内でサービス自身を DNS サーバに (静的または動的に) 登録して行うこともできます。

注意 : Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは `Net.Discovery.queryServices` ([543 ページ](#)を参照) を使用してください。

パラメータ

cType	検索する DNS SRV サービス名。有効な値には、次のようなものがあります。 「 <code>http</code> 」 — Web サーバを検索します 「 <code>ftp</code> 」 — FTP サーバを検索します その他の例は、DNS SRV サービス名のレジストリを参照してください。
oAsync	ネットワークでサービスが見つかったとき、または以前に報告されたサービスが削除されたときに通知される通知オブジェクト。通知メソッドは、 <code>queryServices</code> メソッドから制御が戻るまでは呼び出されず、処理がアイドル状態になったときに呼び出されます。 oAsync パラメータには、次のメソッドが実装されています。 <code>addServices</code> — このメソッドは、クエリに一致する使用可能なサービスが見つかったときに呼び出されます。パラメータは、追加されたサービスの <code>Service Description</code> オブジェクトの配列です。 <code>removeServices</code> — このメソッドは、 <code>addServices</code> 通知メソッドによって以前に追加したサービスが使用できなくなったときに呼び出されます。パラメータは、削除されたサービスの <code>Service Description</code> オブジェクトの配列です。
	注意 : Acrobat 7.0 では、mDNS を使用して見つけたサービス (つまり「local.」ドメイン内) のみが動的に更新されます。
aDomains	(オプション) クエリを行う対象となるドメインの配列。次のドメインのみが有効です。 <code>ServiceDiscovery.local</code> — マルチキャスト DNS (mDNS) を使用してローカルネットワークリンク内でサービスを検索します。このドメインは、非定型のネットワーク環境でネットワークサービスを検索するのに便利ですが、ネットワークサービスを検索できる範囲が、現在のネットワークルータ内のみに限定されます。 <code>ServiceDiscovery.DNS</code> — ユニキャスト DNS を使用して、デフォルト DNS ドメイン内でサービスを検索します。このドメインは、DNS サーバのコンテキストでネットワークサービスを検索するには便利ですが、サービスの登録には、通常、IS 部門の支援が必要なので、あまり動的な使用はできません。

戻り値

クエリの期間を管理するサービスクエリオブジェクト。次のいずれかの条件を満たすまで、クエリは継続します。

- `queryServices` から返されるサービスクエリオブジェクトがガーベッジコレクトされる。
 - `queryServices` から返されるサービスクエリオブジェクトの `stop` メソッドが呼び出される。

メソッド	説明
stop	クエリを中断します。このメソッドは、通知コールバックから呼び出せますが、処理がアイドル状態になるまで処理は中断されません。

例外

Acrobat の標準の例外

Service Description オブジェクト

`addServices` や `removeServices` に渡す `Service Description` オブジェクトには、次のプロパティがあります。

プロパティ	説明
name	サービスの Unicode 表示名。
domain	サービスが見つかった DNS ドメイン。サービスがローカルネットワークで見つかった場合は、「local」になります。
type	見つかったサービスの DNS SRV サービス名。これは、queryServices に渡された cType と同じです。このプロパティは、複数のクエリに同じ通知コールバックを使用するときに便利です。

例

自分自身をパブリッシュしたネットワークサービスを、DNS Service Discovery を使用して見つけます。

このコード例は、実行する場所によって出力される結果が異なります。

```
};

SOAP.queryServices({
    cType: "http",
    oAsync:oNotifications,
    aDomains:[ServiceDiscovery.local, ServiceDiscovery.DNS]
});
```

現在のネットワーク環境によって出力が異なります。DNS Service Discovery でアドバタイズされているサービスがなければ、何も出力されません。次に代表的な出力例を示します。

```
ADD: My Web Server in domain local.
ADD: Joe's Web Server in domain local.
ADD: Example.org Web Server in domain example.org.
```

resolveService

7.0			
-----	--	--	--

接続を確立するために、ネットワークアドレスおよびポートにサービス名をバインドできます。接続情報は非同期に返されますが、時間が経てばネットワーク内のサービスの場所が変更される場合があるので、一時的な情報として扱う必要があります（例えば、DHCP のリースの有効期限が切れたり、サービスが新しいサーバへ移動されたりする場合があります）。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは `Net.Discovery.resolveService` ([543 ページを参照](#)) を使用してください。

パラメータ

cType	解決する DNS SRV サービス名。
cDomain	サービスが見つかったドメイン。
cService	解決するサービス名。
oResult	サービスが解決されたときに呼び出すオブジェクト。
660 ページの「oResult パラメータに関する追加の注意事項」 を参照してください。	

戻り値

解決の期間を管理するサービスクエリオブジェクト。次のいずれかの条件を満たすまで、解決処理は継続されます。

- `resolveService` から返されるサービスクエリオブジェクトがガーベッジコレクトされる。
- `oResult` オブジェクトの `resolve` メソッドが呼び出される。これは、（サービスの解決、エラー、タイムアウトのいずれかで）処理が完了したことを示しています。
- `resolveService` から返されるサービスクエリオブジェクトの `stop` メソッドが呼び出される。

メソッド	説明
stop	解決処理を中断します。このメソッドは、通知コールバックから呼び出せますが、アイドル状態になるまで処理は中断されません。

例外

Acrobat の標準の例外

oResult パラメータに関する追加の注意事項

`oResult` オブジェクトは、サービスが解決されると呼び出される通知オブジェクトです。通知メソッドは、`resolveService` メソッドから制御が戻るまでは呼び出されず、処理がアイドル状態になったときに呼び出されます。`oResult` パラメータには、次のメソッドを実装します。

メソッド	説明
<code>resolve</code>	このメソッドは、サービスが解決されたときや解決できないときに、2つのパラメータ (<code>nstatus</code> と <code>oInfo</code>) とともに呼び出されます。 <code>nStatus</code> パラメータは、サービスが解決できたかどうかを示すステータスです（次を参照）。サービスが正しく解決できた場合は、 <code>ServiceInfo</code> オブジェクトのインスタンスである <code>oInfo</code> オブジェクトに接続情報が指定されます（次を参照）。

`resolve` メソッドに渡される `nstatus` パラメータの値は、次のいずれかです。

値	説明
0	サービスが正しく解決されました。
1	サービスが解決される前にタイムアウトしました。Acrobat 7 以降のデフォルトタイムアウトは 60 秒です。
-1	サービスの解決処理中にネットワークエラーが発生しました。

`resolve` メソッドに渡される `ServiceInfo` オブジェクトには、次のプロパティがあります。

プロパティ	説明
<code>target</code>	サービスを提供するマシンの IP アドレスまたは DNS 名。
<code>port</code>	サービスを提供するマシンのポート。
<code>info</code>	サービスから提供された、名前と値のペアを持つオブジェクト。例えば、HTTP サービスの場合は、 <code>path</code> プロパティに Web サービスのパスが格納されているので、サービス URL は <code>http://<target>:<port>/<info["path"]></code> と表現できます。

例

このコード例は、実行する場所によって出力される結果が異なります。DNS Service Discovery によってアドバタイズされているサービスがなければ、何も出力されません。

```
var oNotifications =
{
    resolve: function(status, info)
    {
        if(status == 0)
            console.println("RESOLVE: http://"
                + info.target + ":" + info.port + "/"
                + info.info.path);
        else console.println("ERROR: " + status);
    }
};
SOAP.resolveService({
    cType: "http",
    cDomain: "local.",
    cService: "Joe's Web Server",
    oResult: oNotifications
});
```

現在のネットワーク環境によって出力が異なりますが、次に代表的な出力例を示します。

```
RESOLVE: http://172.16.0.0:80/index.html
```

request

6.0			F
-----	--	--	----------

SOAP HTTP エンドポイントに対してリモートプロシージャ呼び出し (RPC) を開始するか、XML メッセージを送信します。このメソッドは、エンドポイントの応答を待つ（同期処理）か、通知オブジェクトのメソッドを呼び出します（非同期処理）。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは `Net.SOAP.request` ([542 ページを参照](#)) を使用してください。

パラメータ

`cURL` SOAP HTTP エンドポイントの URL。URL メソッドは、次のいずれかである必要があります。

`http` — ポートの URI でサーバに接続します。例えば、`http://serverName:portNumber/URI` のようになります。

`https` — ポートの URI で、セキュア (SSL) サーバに接続します。例えば、`https://serverName:portNumber/URI` のようになります。

`oRequest` リモートプロシージャ名とパラメータ、または送信する XML メッセージを指定するオブジェクト。

[665 ページの「oRequest パラメータに関する追加の注意事項」](#) を参照してください。

oAsync	(オプション) メソッドの非同期呼び出しを指定するオブジェクト。デフォルトでは、リクエストは同期処理されます。このオブジェクトは、Acrobat 7.0 で変更されています。
	(Acrobat 6.0) oAsync オブジェクトリテラルには、 <code>response</code> という名前の関数が必要です。この関数の呼び出しには、2 つのパラメータ (<code>oResult</code> および <code>cURI</code>) が必要です。 <code>oResult</code> は、リクエストの同期呼び出しを行った場合に返される結果のオブジェクトと同じものです。 <code>cURI</code> は要求先のエンドポイントの URI です。
	(Acrobat 7.0) oAsync オブジェクトのレスポンスコールバックで、次のパラメータが使用できます。
	<code>response</code> — SOAP リクエストからのレスポンスオブジェクト。
	<code>uri</code> — SOAP リクエストの送信先 URI。
	<code>exception</code> — エラーがあった場合は例外オブジェクト (次の例外を参照)、それ以外は <code>null</code> 。
	<code>header</code> — レスポンス SOAP ヘッダ (<code>oRespHeader</code> パラメータの説明を参照)。レスポンスヘッダがない場合は、 <code>null</code> 。
cAction	(オプション) SOAP 1.1 では、このパラメータが <code>SOAPAction</code> ヘッダとして渡されます。SOAP 1.2 では、このパラメータが <code>Content-Type</code> ヘッダのアクションパラメータとして渡されます。
	デフォルトでは、このアクションは空の文字列になります。
	SOAPAction は、HTTP ヘッダに書き込まれる URN です。これを使用してファイアウォールおよびサーバは SOAP リクエストをフィルタします。必要な <code>SOAPAction</code> ヘッダがある場合は、通常、SOAP サービスの WSDL ファイルまたは SOAP サービスの説明に記述されています。
bEncoded	(オプション) SOAP 仕様に説明されている SOAP エンコードを使用してリクエストがエンコードされます。指定しなければ、リテラルエンコードが使用されます。デフォルトは <code>true</code> です。
cNamespace	(オプション) リクエストで SOAP エンコードを使用しない場合のメッセージスキーマの名前空間。
	デフォルトでは、スキーマ宣言が省略されます。
oReqHeader	(オプション、Acrobat 7.0) リクエストに含める SOAP ヘッダを指定するオブジェクト。デフォルトでは、SOAP 本文のみが設定されてリクエストが送信されます。
	このオブジェクトの指定方法は、 <code>oRequest</code> オブジェクトの場合と同じですが、リクエストの説明に指定できる次の 2 つのプロパティが追加されている点が異なります。
	<code>soapActor</code> — SOAP ヘッダを処理する受信者 (または URI で指定されたアクタ)。デフォルトは、リクエストを処理する最初の受信者です。
	<code>soapMustUnderstand</code> — 受信者がこのヘッダタイプを理解しなければリクエスト本文を解釈できないことを示すブーリアン値。デフォルトでは、ヘッダを理解しなくてもよい設定になっています。

<code>oRespHeader</code>	(オプション、Acrobat 7.0) 関数の同期呼び出しを行った場合、メソッドの処理が完了したときに返される SOAP ヘッダが設定されるオブジェクト（それ以外の場合、ヘッダは <code>oAsync</code> コールバックメソッドに渡されます）。 デフォルトでは、ヘッダは返されません。 オブジェクトの形式については、 <code>cResponseStyle</code> パラメータの説明を参照してください。
<code>cVersion</code>	(オプション、Acrobat 7.0) XML メッセージを生成する際に使用する SOAP プロトコルのバージョン。1.1 か 1.2 のいずれかです。 デフォルトでは、「SOAPVersion.version_1_1」が使用されます。
<code>oAuthenticate</code>	(オプション、Acrobat 7.0) HTTP 認証の処理方法や Web Service Security に使用する証明書を指定するオブジェクト。デフォルトでは、BASIC 認証モードと DIGEST 認証モードの HTTP 認証を処理するために、ユーザインターフェイスが表示されます。 <code>oAuthenticate</code> オブジェクトに設定できるプロパティは、次のとおりです。 <code>Username</code> — 認証に使用するユーザ名を含む文字列。 <code>Password</code> — 認証に使用する証明書を含む文字列。 <code>UsePlatformAuth</code> — プラットフォーム認証を使用することを示すブーリアン値。 <code>true</code> の場合は、 <code>Username</code> と <code>Password</code> が無視されて、基盤となるプラットフォームのネットワークコードが使用されます。この場合、認証 UI がユーザに表示されるか、または現在ログインしているユーザの資格証明が使用されます（両方行われることもあります）。デフォルトは <code>false</code> です。サポートされているのは Windows プラットフォームのみです。
<code>cResponseStyle</code>	(オプション、Acrobat 7.0) 戻り値 (SOAP 本文の場合) と <code>oRespHeader</code> オブジェクト (SOAP ヘッダの場合) の構成を示す列挙型。 <code>SOAPMessageStyle.js</code> — (デフォルト) レスポンスは、返されたメッセージ (Acrobat 6.0 が生成したもの) の SOAP 本文 (または SOAP ヘッダ) を表すオブジェクトになります。リクエストに SOAP エンコードを使用するときにはこのパラメータを使用することをお勧めしますが、リテラルエンコードを使用するときにはお勧めしません。その場合は、XML または Message スタイルを使用することをお勧めします。 <code>SOAPMessageStyle.XML</code> — レスポンスは、SOAP 本文 (または SOAP ヘッダ) が XMLfragment として格納されたストリームオブジェクトになります。このレスポンスに添付ファイルが関連付けられている場合は、 <code>Stream</code> オブジェクトに <code>oAttachments</code> オブジェクトプロパティが設定されます。オブジェクトキーは、添付ファイルの一意の名前です。値は、添付ファイルの内容を含む <code>Stream</code> オブジェクトです。 <code>SOAPMessageStyle.Message</code> — レスポンスは、XML メッセージに対応する SOAP 本文 (または SOAP ヘッダ) を表すオブジェクトになります。これは、JavaScript のレスポンススタイルと次の点が異なります。 <ul style="list-style-type: none">• XML 要素は、1 つのオブジェクトとしてではなく、オブジェクトの配列として返されます。これによって、要素の順序が維持され、同じ名前の要素が複数使用できます。• XML 属性は、<code>soapAttributes</code> プロパティを使用して維持されます。• 名前空間が処理されて、<code>soapName</code> プロパティと <code>soapQName</code> プロパティに返されます。• 要素の内容は、<code>soapValue</code> プロパティに設定されます。

cRequestStyle	(オプション、Acrobat 7.0.5) <code>oRequest</code> の解釈方法を変更できます。有効な値は、次のとおりです。 <code>SOAPRequestStyle.SOAP</code> — (デフォルト) SOAP メッセージングモデルを使用してリクエストを行います。 <code>SOAPRequestStyle.RawPost</code> — <code>oRequest</code> パラメータを HTTP Post のリクエスト本文として使用します。 <code>oRequest</code> は、 <code>ReadStream</code> オブジェクトである必要があります。このメソッドを文書のコンテキストで呼び出す場合は、文書がブラウザで開かれている必要があります。また、文書の送信元の URL (スキーム、サーバ、ポート) が、 <code>cURL</code> パラメータと一致している必要があります。リクエストのレスポンスが含まれた <code>ReadStream</code> オブジェクトが返されます。
cContentType	(オプション、Acrobat 7.0.5) HTTP Content-Type ヘッダを指定できます。デフォルトでは、SOAP メッセージングの HTTP Content-Type が使用されます。

戻り値

同期呼び出しを行った場合 (`oAsync` パラメータがない場合) はレスポンスオブジェクト。非同期呼び出しを行った場合は何も返しません。このオブジェクトについては、前述の `cResponseStyle` の説明を参照してください。

返される SOAP 型と JavaScript 型との対応関係は、次のとおりです。

SOAP 型	JavaScript 型
<code>xsd:string</code>	文字列
<code>xsd:integer</code>	数値
<code>xsd:float</code>	数値
<code>xsd:dateTime</code>	日付
<code>xsd:boolean</code>	ブーリアン
<code>xsd:hexBinary</code>	<code>ReadStream</code> オブジェクト
<code>xsd:base64Binary</code>	<code>ReadStream</code> オブジェクト
<code>SOAP-ENC:base64</code>	<code>ReadStream</code> オブジェクト
<code>SOAP-ENC:Array</code>	配列
型情報なし	文字列

例外

SOAP エンドポイントから `SOAPFault` が返されると `SOAPError` が発生します。`SOAPError` 例外オブジェクトのプロパティは、次のとおりです。

プロパティ	説明
<code>faultCode</code>	このエラーの SOAP エラーコードを示す文字列。
<code>faultActor</code>	このエラーを発生させた SOAP アクタを示す文字列。
<code>faultDetail</code>	エラーの詳細を示す文字列。

`NetworkError` は、基盤となる HTTP トランスポート層またはネットワーク接続で障害が発生した場合に発生します。`NetworkError` 例外オブジェクトの `statusCode` プロパティに、HTTP ステータスコードまたは -1 (ネットワーク接続が確立できなかった場合) が設定されます。

Acrobat の標準の例外が発生する場合もあります。

注意：メソッドの非同期呼び出しを行った場合は、例外オブジェクトが `response` コールバックメソッドに渡される場合もあります。

oRequest パラメータに関する追加の注意事項

`oRequest` パラメータは、呼び出すリモートプロシージャ名とパラメータを指定するオブジェクトリテラルです。このオブジェクトリテラルでは、リモートプロシージャの完全修飾されたメソッド名をキーとして使用します。名前空間とメソッド名はコロンで区切る必要があります。

例えば、メソッドの名前空間が `http://mydomain/methods` で、メソッド名が `echoString` の場合、完全修飾名は `http://mydomain/methods:echoString` になります。このキーの値はオブジェクトリテラルです。このオブジェクトリテラルの各キーはメソッドのパラメータで、各値は対応するパラメータの値です。例えば、次のようにになります。

```
oRequest: {  
    "http://soapinterop.org/:echoString":{inputString: "Echo!"}  
}
```

パラメータをリモートプロシージャに渡すと、次の表のように、JavaScript 型が SOAP 型に自動的にバインドされます。

JavaScript 型	SOAP 型
文字列	xsd:string
数値	xsd:float
日付	xsd:dateTime
ブーリアン	xsd:boolean
ReadStream オブジェクト	SOAP-ENC:base64
配列	SOAP-ENC:Array
その他	型情報なし

注意：`xsd` 名前空間は、XML スキーマデータ型名前空間 (<http://www.w3.org/2001/XMLSchema>) を参照します。`SOAP-ENC` 名前空間は、SOAP エンコード名前空間 (<http://schemas.xmlsoap.org/soap/encoding/>) を参照します。

`oRequest` オブジェクトがサポートするプロパティは、次のとおりです。

プロパティ	説明
<code>soapType</code>	<p>生成する SOAP メッセージの値に使用する SOAP 型。前述の自動的にバインドされるデータ型以外のデータ型が必要な場合に使用します。この型は、<名前空間><型>のように名前空間で修飾する必要があります。例えば、次のようになります。</p> <pre>http://mydomain/types:myType</pre> <p>ただし、<code>xsd</code> (XML スキーマデータ型)、<code>xsi</code> (XML スキーマインスタンス)、<code>SOAP-ENC</code> (SOAP エンコード) 名前空間は、SOAP メッセージで暗黙的に定義されているので、<code>soapType</code> で使用することができます。例えば、XML スキーマデータ型の整数型を表す場合は、<code>xsd:int</code> となります。</p>
<code>soapValue</code>	<p>(Acrobat 6.0) SOAP メッセージの生成時に使用される値。文字列または <code>ReadStream</code> オブジェクトです。<code>soapValue</code> はエスケープされずに渡されます (つまり、エスケープされた XML エンティティにはなりません)。例えば、XML メッセージで「<」は「&lt;」に変換されません。したがって、<code>soapValue</code> に未加工の XML フラグメントを指定すると、それがそのまま XML メッセージに渡されます。</p> <p>(Acrobat 7.0) <code>soapValue</code> に、ノードの配列も使用できるようになりました。これはリクエストメッセージ中のノードに対する、順序付けられた一連の子要素になります。</p>
<code>soapName</code>	<p>SOAP メッセージの生成時に、オブジェクトリテラルのキー名の代わりに使用する要素名。</p> <p>例えば、JavaScript では整数はサポートされませんが、SOAP メソッドの整数パラメータを次のようにして作成できます。</p> <pre>var oIntParameter = { soapType: "xsd:int", soapValue: "1" };</pre> <p>これにより、<code>SOAP.request</code> メソッドの <code>oRequest</code> パラメータでは次のように指定できます。</p> <pre>oRequest: { "http://soapinterop.org/:echoInteger": { inputInteger: oIntParameter } }</pre> <p>この方法については、667 ページの「例 1」 を参照してください。</p>
<code>soapAttributes</code>	<p>(Acrobat 7.0) リクエストノードに対応する要素を作成するときに使用する XML 属性を含んだオブジェクト。オブジェクトキーに属性名を指定し、値に属性値を指定します。</p>
<code>soapQName</code>	<p>(Acrobat 7.0) リクエストノードの名前空間修飾名 (QName) を指定するオブジェクト。例えば、<code><ns:local xmlns:ns="urn:example.org"></code> という要素では、要素名はローカル名 (local) と名前空間 (urn:example.org) で構成される QName になります。</p> <p>このオブジェクトには、次の 2 つのプロパティがあります。</p> <ul style="list-style-type: none"> <code>localName</code> — QName のローカル名を示す文字列。 <code>nameSpace</code> — QName の名前空間を示す文字列。

プロパティ	説明
soapAttachment	(Acrobat 7.0) ブーリアン値。SwA 仕様に従って、ノードの soapValue コンテンツを添付ファイルとしてエンコードする必要があるかどうかを示します。対応する soapAttachment プロパティが true の場合、soapValue はストリームであることが必要です。そうでない場合は例外が発生します。
soapParamOrder	(Acrobat 7.0) RPC パラメータをサーバに送信する順序を示す配列。この配列は、パラメータ名を表す文字列のセットです。この値が適用されるのは、bEncoding が true の場合のみです。

例 1

request を使用して、エコーサービスのリモートプロシージャを呼び出します。

サービス WSDL 文書の URL が必要です。この URL は、次の URL の「Round 2 Interop Services - using SOAP 1.2」セクションから取得できます。<http://www.whitemesa.com/interop.htm>

```
var cURL = <get a URL for this service from  
          http://www.whitemesa.com/interop.htm>;  
  
var cTestString = "This is my test string";  
  
// echoString という SOAP メソッドを呼び出す。これは RPC エンコードのメソッド  
var response = SOAP.request(  
{  
    cURL: cURL,  
    oRequest: {  
        "http://soapinterop.org/:echoString": {  
            inputString: cTestString  
        }  
    },  
    cAction: "http://soapinterop.org/"  
});  
  
var result =  
response["http://soapinterop.org/:echoStringResponse"]["return"];  
  
// この値は cTestString と等しくなる  
console.println(result + " == " + cTestString);  
  
// echoInteger SOAP メソッドを呼び出す。JavaScript では整数がサポート  
// されていないので、独自の整数オブジェクトを作成  
var oTestInt =  
{  
    soapType: "xsd:int",  
    soapValue: "10"  
};  
  
var response = SOAP.request(  
{  
    cURL: cURL,  
    oRequest: {  
        "http://soapinterop.org/:echoInteger": {  
            soapValue: "10"  
        }  
    },  
    cAction: "http://soapinterop.org/:echoInteger"  
});  
  
var result =  
response["http://soapinterop.org/:echoIntegerResponse"]["return"];  
  
// この値は oTestInt と等しくなる  
console.println(result + " == " + oTestInt);
```

```
        inputInteger: oTestInt
    }
},
cAction: "http://soapinterop.org/"
});

var result =
response["http://soapinterop.org/:echoIntegerResponse"]["return"];

// この値は oTestInt.soapValue と等しくなる
console.println(result + " == " + oTestInt.soapValue);
```

出力は次のようにになります。

```
This is my test string == This is my test string
10 == 10
```

例 2

SOAP ヘッダを設定して、それを取得します。

```
var cURL = <URL of a Service>;
var NS = "http://www.example.com/soap/:";
var oHeader = {};
oHeader[NS + "testSession"] =
{
    soapType: "xsd:string",
    soapValue: "Header Test String"
};
var oResultHeader = {};
var oRequest = {};
oRequest[NS + "echoHeader"] = {};
var response = SOAP.request(
{
    cURL: cURL,
    oRequest: oRequest,
    cAction: "http://soapinterop.org/",
    oReqHeader: oHeader,
    oRespHeader: oResultHeader
});
```

例 3

HTTP 認証を使用したエコーサービスのリクエスト

```
var oAuthenticator =
{
    Username: "myUserName",
    Password: "myPassword"
};
var response = SOAP.request(
{
    cURL: cURL,
    oRequest: {
        "http://soapinterop.org/:echoString": {

```

```
        inputString: cTestString
    },
},
cAction: "http://soapinterop.org/",
oAuthenticate: oAuthenticator
});
```

response

6.0			F
-----	--	--	----------

応答を待たずに、SOAP HTTP エンドポイントに対してリモートプロシージャ呼び出し（RPC）を開始するか、XML メッセージを送信します。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは `Net.SOAP.response` ([542 ページ](#)を参照) を使用してください。

パラメータ

cURL	SOAP HTTP エンドポイントの URL。URL メソッドは、次のいずれかである必要があります。 <code>http</code> — ポートの URI でサーバに接続します。例えば、 <code>http://serverName:portNumber/URI</code> のようになります。 <code>https</code> — ポートの URI で、セキュア（SSL）サーバに接続します。 例えば、 <code>https://serverName:portNumber/URI</code> のようになります。 <code>SOAP.request</code> の <code>cURL</code> パラメータを参照してください。
oRequest	リモートプロシージャ名とパラメータ、または送信する XML メッセージを指定するオブジェクト。 <code>SOAP.request</code> の <code>oRequest</code> パラメータを参照してください。
cAction	(オプション) SOAP 仕様に説明されている、このリクエストに対する <code>SOAPAction</code> ヘッダ。 デフォルトでは、 <code>SOAPAction</code> は空になります。 <code>SOAP.request</code> の <code>cAction</code> パラメータを参照してください。
bEncoded	(オプション) ブーリアン値。リクエストのエンコードに、SOAP 仕様に説明されている SOAP エンコードが使用されたかどうかを示します。デフォルトは <code>true</code> です。
cNamespace	(オプション) SOAP エンコードを使用しない場合 (<code>bEncoded</code> フラグが <code>false</code> の場合) のメッセージスキーマの名前空間。 デフォルトでは、名前空間はありません。
oReqHeader	(オプション、Acrobat 7.0) リクエストに含める SOAP ヘッダを指定するオブジェクト。 デフォルトでは、SOAP 本文のみが設定されてリクエストが送信されます。 <code>SOAP.request</code> の <code>oReqHeader</code> パラメータを参照してください。

cVersion	(オプション、Acrobat 7.0) 使用する SOAP プロトコルのバージョン。 デフォルトでは、「SOAPVersion.version_1_1」が使用されます。 SOAP. request の cVersion パラメータを参照してください。
oAuthenticate	(オプション、Acrobat 7.0) 証明書の発行に使用する認証方式のタイプを指定するオブジェクト。 デフォルトでは、HTTP 認証が必要になると対話処理を行うための UI が表示されます。 SOAP. request の oAuthenticate パラメータを参照してください。
cRequestStyle	(オプション、Acrobat 7.0.5) SOAP. request の cRequestStyle と同じですが、サーバからレスポンスが返されない点が異なります。
cContentType	(オプション、Acrobat 7.0.5) SOAP. request の cContentType と同じですが、サーバからレスポンスが返されない点が異なります。

戻り値

ブーリアン

例外

エンドポイントが使用できないなど、ネットワークレベルで問題がある場合は、NetworkError が発生します。

例

[667 ページの「例 1」](#) を参照してください。

streamDecode

6.0			
-----	--	--	--

指定のエンコードタイプ (cEncoder) で oStream オブジェクトをデコードします。このメソッドは、デコードされた ReadStream オブジェクトを返します。このメソッドは、SOAP メソッドで Base64 または 16 進でエンコードされたデータが返された場合などに使用します。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは Net.streamDecode ([544 ページ](#)を参照) を使用してください。

パラメータ

oStream	指定のエンコードタイプでデコードするストリームオブジェクト。
cEncoder	有効な文字列は、「hex」(16 進のエンコード) および「base64」(Base64 エンコード) です。

戻り値

ReadStream オブジェクト

streamDigest

7.0			
-----	--	--	--

指定のエンコードタイプ (cEncoder) で、ostream オブジェクトのダイジェストを生成します。このメソッドは、ostream から計算されたダイジェストを含む ReadStream オブジェクトを返します。このメソッドは、元のデータストリームの完全性を検証するためにダイジェストを計算したり、Web サービスの認証方式の一部としてダイジェストを計算する場合などに使用します。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは Net.streamDigest ([544 ページ](#)を参照) を使用してください。

パラメータ

oStream	ダイジェストを計算するストリームオブジェクト。計算には、指定のメッセージダイジェストアルゴリズムが使用されます。
cEncoder	使用するダイジェストアルゴリズム。cEncoder パラメータには、次のいずれかの値を設定する必要があります。 StreamDigest.MD5 — MD5 ダイジェストアルゴリズム (RFC 1321 を参照) を使用してコンテンツのダイジェストを生成します。 StreamDigest.SHA1 — SHA-1 ダイジェストアルゴリズム (RFC 3174 を参照) を使用してコンテンツのダイジェストを生成します。

戻り値

ストリームのバイナリダイジェストを含む ReadStream オブジェクト。文字列として使用するには、SOAP.streamEncode を使用して、テキスト形式 (Base64 や 16 進など) に結果を変換する必要があります。

例

文字列のダイジェストを生成します。最初に文字列をストリームに変換し、続いて streamDigest を呼び出します。

```
var srcStm = SOAP.streamFromString("This is a string I want to digest");
var digest = SOAP.streamDigest(srcStm, StreamDigest.SHA1);
```

streamEncode

6.0			
-----	--	--	--

この関数は、ストリームオブジェクトをエンコードします。SOAP メソッドに Base64 または 16 進でエンコードしたデータを渡す必要がある場合などに、このメソッドを使用します。

注意：Acrobat 8.0 以降では、このメソッドは非推奨になりました。新しいサービスでは Net.streamEncode ([544 ページ](#)を参照) を使用してください。

パラメータ

<code>oStream</code>	指定のエンコードタイプでエンコードするストリームオブジェクト。
<code>cEncoder</code>	エンコードタイプを指定する文字列。有効な値は、「hex」（16進のエンコード）と「base64」（Base64 エンコード）です。

戻り値

適切にエンコードされた `ReadStream` オブジェクト。

streamFromString

6.0			
-----	--	--	--

この関数は、文字列を `ReadStream` オブジェクトに変換します。SOAP メソッドにデータを渡す場合などに使用します。

パラメータ

<code>cString</code>	変換する文字列。
----------------------	----------

戻り値

`ReadStream` オブジェクト

stringFromStream

6.0			
-----	--	--	--

この関数は、`ReadStream` オブジェクトを文字列に変換します。SOAP メソッドのレスポンスで返されたストリームオブジェクトの内容を調べる場合などに使用します。

パラメータ

<code>oStream</code>	変換する <code>ReadStream</code> オブジェクト。
----------------------	--------------------------------------

戻り値

文字列

Sound

5.0			
-----	--	--	--

このオブジェクトは、文書に保持されているサウンドを表します。すべての Sound オブジェクトの配列は、Doc の [sounds](#) プロパティから取得できます。Doc の [getSound](#)、[importSound](#)、[deleteSound](#) の各メソッドも参照してください。

Sound のプロパティ

name

Sound オブジェクトに関連付けられている名前。

型

文字列

アクセス

R

例

現在の文書に埋め込まれているすべてのサウンドの名前をコンソールに出力します。

```
console.println("Dumping all sound objects in this document.");
var s = this.sounds;
for (var i = 0; i < this.sounds.length; i++)
    console.println("Sound[" + i + "]=" + s[i].name);
```

Sound のメソッド

pause

現在再生中のサウンドを一時停止します。サウンドが既に一時停止されている場合は、サウンドが再開されます。

play

サウンドを非同期で再生します。

stop

現在再生中のサウンドを停止します。

Span

6.0			
-----	--	--	--

リッチテキストフォームフィールドや注釈に含まれている特定のテキスト範囲とそれに関連付けられているプロパティを表す汎用オブジェクト。リッチテキスト値は、テキストと書式を表す `span` オブジェクトの配列で構成されています。

注意：`span` オブジェクトは、フィールドや注釈のリッチテキスト値のコピーです。リッチテキスト値を変更またはリセットしてフィールドを更新するには、`Field` オブジェクトの `richValue` プロパティ（または `Annotation` オブジェクトの `richContents` プロパティ）や、`event` オブジェクトの `richValue`、`richChange`、`richChangeEx` の各プロパティを使用します。

Span のプロパティ

alignment

テキストの水平方向の整列。1 行のテキストの整列は、行の最初の範囲（Span）によって決定されます。`alignment` の値は、次のとおりです。

`left`
`center`
`right`

デフォルト値は `left` です。

型

文字列

アクセス

R / W

例

`alignment` の使用法については、[superscript](#) の例を参照してください。

fontFamily

テキストの描画に使用されるフォントファミリー。これはファミリーネームの配列で、順に検索が行われます。配列の最初のエントリは、使用するフォントのフォント名です。2 番目のエントリは、最初のフォントと正確に一致するものがない場合に使用するオプションの汎用ファミリーネームです。汎用ファミリーネームには、次のものがあります。

`symbol`、`serif`、`sans-serif`、`cursive`、`monospace`、`fantasy`

デフォルトの汎用ファミリーネームは `sans-serif` です。

型

配列

アクセス

R / W

例

リッチテキストフィールドの defaultStyle フォントファミリーを設定します。

```
f = this.getField("Text1");
style = f.defaultStyle;

// ユーザのシステムに Courier Std がない場合は monospace フォントを使用
style.fontFamily = ["Courier Std", "monospace"];
f.defaultStyle = style;
```

fontStretch

テキストの描画に使用するフォントファミリーから、標準、狭いまたは広い文字幅を指定します。
fontStretch の値は、次のとおりです。

ultra-condensed、extra-condensed、condensed、semi-condensed、normal、
semi-expanded、expanded、extra-expanded、ultra-expanded

デフォルト値は normal です。

型

文字列

アクセス

R / W

fontStyle

テキストを斜体フォントで描画するように指定します。

italic
normal

デフォルトは normal です。

型

文字列

アクセス

R / W

fontWeight

テキストの描画に使用するフォントの太さ。目安としては、700よりも小さい数字が標準、700以上が太字と考ることができます。fontWeight の値は、次のとおりです。

100、200、300、400、500、600、700、800、900

デフォルト値は 400 です。

型

数値

アクセス

R / W

strikethrough

strikethrough が `true` の場合は、テキストに取り消し線が描画されます。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

subscript

テキストを下付き文字に指定します。`true` の場合、ポイントサイズが小さくベースラインが低い下付きテキストが描画されます。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

superscript

テキストを上付き文字に指定します。`true` の場合、ポイントサイズが小さくベースラインが高い上付きテキストが描画されます。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

例

様々なプロパティを使用して、リッチテキストをリッチテキストフィールドに書き込みます。詳しい説明と例については、Field オブジェクトの [richValue](#) プロパティを参照してください。

```
var f = this.getField("myRichField");

// Span オブジェクトを保持する配列を作成
var spans = new Array();

// 各 Span オブジェクトを作成
spans[0] = new Object();
spans[0].alignment = "center";
spans[0].text = "The answer is x";

spans[1] = new Object();
spans[1].text = "2/3";
spans[1].superscript = true;

spans[2] = new Object();
spans[2].superscript = false;
spans[2].text = ". ";

spans[3] = new Object();
spans[3].underline = true;
spans[3].text = "Did you get it right?";
spans[3].fontStyle = "italic";
spans[3].textColor = color.red;

// Span オブジェクトの配列を、field.richValue を使用して
// フィールドに割り当てる
f.richValue = spans;
```

text

範囲 (Span) 内のテキスト。

型

文字列

アクセス

R / W

例

`text` の使用法については、[superscript](#) の例を参照してください。

textColor

テキストの描画に使用される RGB カラーを表すカラー配列（[color](#) オブジェクトを参照）。デフォルトのカラーは黒です。

型

カラー配列

アクセス

R / W

例

`textColor` の使用法については、[superscript](#) の例を参照してください。

textSize

テキストのポイントサイズ。`textSize` の値は、0 ~ 32767 の任意の数字です。テキストサイズ 0 は、すべてのテキストデータがフィールドの矩形に収まる最大のポイントサイズを使用することを表します。

デフォルトのテキストサイズは 12.0 です。

型

数値

アクセス

R / W

例

`textSize` の使用法については、[Field](#) オブジェクトの [richValue](#) プロパティの例を参照してください。

underline

`underline` が `true` の場合、テキストに下線が付けられます。デフォルトは `false` です。

型

ブーリアン

アクセス

R / W

例

`underline` の使用法については、[superscript](#) の例を参照してください。

spell

このオブジェクトを使用すると、注釈やフォームフィールドなどでスペルチェックを行うことができます。
spell オブジェクトを使用するには、Acrobat Spelling プラグインとスペルチェック用の辞書をインストールしておく必要があります。

注意：7.0 より前のバージョンの Adobe Reader では、spell オブジェクトは使用できません。
Adobe Reader 7.0 では、customDictionaryCreate、customDictionaryDelete、
customDictionaryExport を除くすべてのプロパティとメソッドにアクセスできます。

spell のプロパティ

available

5.0			
-----	--	--	--

spell オブジェクトが使用可能な場合は true。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

型

ブーリアン

アクセス

R

例

```
console.println("Spell checking available: " + spell.available);
```

dictionaryNames

5.0			
-----	--	--	--

使用可能な辞書名の配列。この配列のサブセットを check、checkText、checkWord や、
spellDictionaryOrder に渡すと、特定の辞書を使用したり、辞書の検索順序を指定したりすることができます。

ユーザが使用可能な辞書名のリストを取得するには、コンソールで spell.dictionaryNames を実行します。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

型

配列

アクセス

R

dictionaryOrder

5.0			
-----	--	--	--

「スペルチェック」環境設定パネルで指定されている辞書の検索順序の配列。Spelling プラグインは、最初に Doc の spellDictionaryOrder 配列の単語を検索し（設定されている場合）、その後でこの辞書配列を検索します。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

型

配列

アクセス

R

domainNames

5.0			
-----	--	--	--

他のプラグインが Spelling プラグインに登録したスペルチェック範囲名の配列。この配列のサブセットを check に渡すと、スペルチェックの範囲を限定することができます。

ユーザのインストールに応じて、例えば、次の範囲名が有効となります。

Everything
Form Field
All Form Fields
Comment
All Comments

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

型

配列

アクセス

R

languages

6.0			
-----	--	--	--

このプロパティは、使用可能な ISO 639-2、ISO 3166-1 の言語／国コードの配列を返します。この配列のサブセットを、check、checkText、checkWord や、customDictionaryCreate のメソッド、Doc の spellLanguageOrder プロパティに渡すと、特定の言語を使用したり、言語の検索順序を指定したりすることができます。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

ユーザのインストールに応じて、例えば、次の一覧の言語／国コードが有効となります。

コード	説明
ca_ES	カタロニア語
cs_CZ	チェコ語
da_DK	デンマーク語
nl_NL	オランダ語
en_CA	英語：カナダ
en_GB	英語：英
en_US	英語：米
fi_FI	フィンランド語
fr_CA	フランス語：カナダ
fr_FR	フランス語
de_DE	ドイツ語
de_CH	ドイツ語：スイス
el_GR	ギリシャ語
hu_HU	ハンガリー語
it_IT	イタリア語
nb_NO	ノルウェー語：ボクモール
nn_NO	ノルウェー語：ニーノシク
pl_PL	ポーランド語
pt_BR	ポルトガル語：ブラジル
pt_PT	ポルトガル語
ru_RU	ロシア語

コード	説明
es_ES	スペイン語
sv_SE	スウェーデン語
tr_TR	トルコ語

注意：Acrobat 7.0 と Acrobat 6.0 では、返されるエントリが異なります。Acrobat 6.0 の ISO コードは、JavaScript から入力されると内部的に新しい ISO コードにマッピングされます。したがって、Acrobat 6.0 向けの JavaScript コードはそのまま使用できます。コードは出力時には変換されません。

型

配列

アクセス

R

例

使用可能なすべての言語コードをリストします。

```
console.println( spell.languages.toSource() );
```

languageOrder

6.0			
-----	--	--	--

辞書の検索順序を表す ISO 639-2、ISO 3166 言語コードの配列。これは、ユーザが「スペルチェック」環境設定パネルで指定した順序です。Spelling プラグインは、最初に Doc の spellLanguageOrder 配列の単語を検索し（設定されている場合）、その後でこの言語配列を検索します。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

型

配列

アクセス

R

例

辞書検索順のリストを取得します。

```
console.println( spell.languageOrder.toSource() );
```

spell のメソッド

addDictionary



注意：Acrobat 6.0 以降、このメソッドはサポートされなくなり、常に `false` が返されるようになりました。代わりに、[customDictionaryOpen](#) メソッドを使用してください。

使用可能な辞書のリストに辞書を追加します。

辞書は、実際には、`DDDxxxxx.hyp`、`DDDxxxxx.lex`、`DDDxxxxx.clx`、`DDDxxxxx.env` の 4 ファイルで構成されています。`cFile` パラメータには、`.hyp` ファイルのパスをデバイスに依存しない形式で指定します。例えば「`/c/temp/testdict/TST.hyp`」というファイルを指定した場合、Spelling プラグインは `TST.hyp` ファイルが置かれているフォルダ (`testdict`) から、他の 3 ファイルを探します。ファイル名の先頭 3 文字は 4 つのファイルで共通にし、かつ他の辞書とは違うものにする必要があります。これによって 4 つのファイルが関連付けられます。Macintosh の場合であっても、ファイル名の最後には、ドットの後に前述の拡張子を付ける必要があります。

パラメータ

<code>cFile</code>	デバイスに依存しない、辞書ファイルのパス。
<code>cName</code>	スペルチェックダイアログボックスで使用される辞書名。 <code>check</code> 、 <code>checkText</code> 、 <code>checkWord</code> メソッドの入力パラメータとして使用できます。
<code>bShow</code>	(オプション) <code>true</code> (デフォルト) の場合、 <code>cName</code> 値が「ユーザ :」の名前として、すべてのリストやメニューにこの名前が表示されます。例えば、 <code>cName</code> が「Test」の場合、「ユーザ : Test」がリストやメニューに追加されます。 <code>false</code> の場合、このカスタム辞書の名前はどのリストやメニューにも表示されません。

戻り値

`false`

addWord



辞書に新しい単語を追加します。[removeWord](#) も参照してください。

注意：Acrobat 7.0 以降では、このメソッドは、コンソールイベントまたはバッティイベントでのみ使用可能です。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

内部的には、`spell` オブジェクトによってユーザ辞書「Not-A-Word」がスキャンされ、指定した単語がそこに含まれている場合はその単語が削除されます。単語がこの辞書に含まれていない場合は、追加されます。実際の辞書は変更されません。

Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cWord	追加する新しい単語。
cName	(オプション) 辞書名または言語コード。現在インストールされている辞書名の配列は、 <code>dictionaryNames</code> または <code>languages</code> で取得できます。

戻り値

成功した場合は `true`、それ以外の場合は `false`。

check

5.0			
-----	--	--	--

フォームフィールドや注釈などのオブジェクトでスペルミスのある単語を訂正するための、スペルチェックダイアログボックスを表示します。

注意 : Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

aDomain	(オプション) フォームフィールドや注釈など、 <code>Spelling</code> プラグインでチェックする範囲名の配列。範囲名の配列を指定しなかった場合は、「EveryThing」の範囲でチェックされます。登録されているチェック範囲名の配列は、 <code>domainNames</code> プロパティで取得できます。
aDictionary	(オプション) スペルチェックで使用する辞書名または言語コードの配列。配列内の辞書の順序で、スペルミスのある単語がチェックされます。現在インストールされている辞書名の配列は、 <code>spell.dictionaryNames</code> または <code>spell.languages</code> で取得できます。このパラメータを省略した場合は、 <code>spellDictionaryOrder</code> リスト、 <code>dictionaryOrder</code> リストの順に検索されます。

戻り値

フラグの付いたすべての単語をユーザが変更するか無視した場合、`true` を返します。すべての単語のチェックが終了する前にユーザがダイアログボックスを終了すると、`false` を返します。

例

辞書を設定し、現在の文書の注釈やフォームフィールドをスペルチェックして、コンソールに結果を表示します。

```
var dictionaries = ["de", "French", "en-GB"];
var domains = ["All Form Fields", "All Annotations"];
if (spell.check(domains, dictionaries) )
    console.println("You get an A for spelling.");
else
    console.println("Please spell check this form before you submit.");
```

checkText

5.0			
-----	--	--	--

指定した文字列中のスペルミスのある単語を訂正するためのスペルチェックダイアログボックスを表示します。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cText	チェックする文字列。
aDictionary	(オプション) スペルチェックで使用する辞書名または言語コードの配列。配列内の辞書の順序で、スペルミスのある単語がチェックされます。インストールされている辞書名の配列は、spell.dictionaryNames または spell.languages で取得できます。このパラメータを省略した場合は、spellDictionaryOrder リスト、dictionaryOrder リストの順に検索されます。

戻り値

スペルチェックダイアログボックスの新しい結果を表す文字列。

例

特定のフィールドをスペルチェックして、フィールドのスペルを更新します。

```
var f = this.getField("Text Box") // フォームテキストボックス
f.value = spell.checkText(f.value); // ユーザに辞書を選択させる
```

checkWord

5.0			
-----	--	--	--

指定された単語のスペルをチェックします。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cWord	チェックする単語。
aDictionary	(オプション) スペルチェックでスペルミスのある単語のチェックに使用する、辞書名または言語コードの配列。スペルチェックは、配列に現れる順序で辞書を使用します。インストールされている辞書名の配列は、spell.dictionaryNames または spell.languages で取得できます。このパラメータを省略した場合は、spellDictionaryOrder リスト、dictionaryOrder リストの順に検索されます。

戻り値

スペルが正しい場合は null オブジェクト。それ以外の場合は、修正候補の配列。

例 1

修正候補の配列をリストボックスに代入します。

```
var word = "subpinna"; /* 「subpoena」のスペルミス */
var dictionaries = ["English"];
var f = this.getField("Alternatives") // 修正候補を表示するためのリストボックス
f.clearItems();
f.setItems(spell.checkWord(word, dictionaries));
```

例 2

次のスクリプトは、文書内のスペルミスのある単語に波形の注釈マークを付け、注釈の内容に修正候補を設定します。このスクリプトは、コンソール、文書内のマウスボタンを放すアクション、メニュー、バッチシーケンスから実行できます。

```
var ckWord, numWords;
for (var i = 0; i < this.numPages; i++)
{
    numWords = this.getPageNumWords(i);
    for (var j = 0; j < numWords; j++)
    {
        ckWord = spell.checkWord(this.getPageNthWord(i, j))
        if ( ckWord != null )
        {
            this.addAnnot({
                page: i,
                type: "Squiggly",
                quads: this.getPageNthWordQuads(i, j),
                author: "A. C. Acrobat",
                contents: ckWord.toString()
            });
        }
    }
}
```

customDictionaryClose

6.0			
-----	--	--	--

customDictionaryOpen または customDictionaryCreate で開かれたカスタム辞書を閉じます。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cName	この辞書が開かれたとき、または作成されたときに使用された辞書名。
-------	----------------------------------

戻り値

成功した場合は `true`。失敗した場合は `false`。

customDictionaryCreate

6.0			
-----	--	--	--

このメソッドは、新しいカスタム辞書ファイルを作成し、使用可能な辞書のリストに追加します。

注意：このメソッドを使用できるのは、コンソールイベントまたはバッチャイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#)を参照してください。

パラメータ

<code>cName</code>	スペルチェックダイアログボックスで使用される辞書名。 <code>check</code> 、 <code>checkText</code> 、 <code>checkWord</code> メソッドの入力パラメータとして使用できます。
<code>cLanguage</code>	(オプション) このパラメータは、辞書を言語に関連付けるために使用します。使用可能な言語のリストは、 <code>spell.languages</code> プロパティから取得できます。
<code>bShow</code>	(オプション) <code>true</code> (デフォルト) の場合、 <code>cName</code> パラメータが「ユーザ:」の名前として、すべてのリストやメニューにこの名前が表示されます。例えば、 <code>cName</code> が「Test」の場合、「ユーザ: Test」がリストやメニューに追加されます。 <code>bShow</code> が <code>false</code> の場合、このカスタム辞書の名前はどのリストやメニューにも表示されません。

戻り値

成功した場合は `true`。失敗した場合は `false`。辞書が保存されているディレクトリへの読み取りおよび書き込み権限がない場合、このメソッドは失敗します。

例

この文書『JavaScript for Acrobat API Reference』を Acrobat で開いて、次のスクリプトをコンソールで実行します。このスクリプトは、各しおり内の最初の単語を抽出します。その単語が既に辞書内にあれば、何も行いません。不明な単語は、JavaScript のオブジェクト名、プロパティ名、メソッド名と見なして、「JavaScript」という新たに作成した辞書に追加します。

```
spell.customDictionaryCreate("JavaScript", "en", true);
function GetJSTerms(bm, nLevel)
{
    var newWord = bm.name.match(re);
    var ckWord = spell.checkWord( newWord[0] );
    if ( ckWord != null )
    {
        var cWord = spell.addWord( newWord[0], "JavaScript");
        if ( cWord ) console.println( newWord[0] );
    }
    if (bm.children != null)
        for (var i = 0; i < bm.children.length; i++)
            GetJSTerms(bm.children[i], nLevel + 1);
}
```

```
console.println("Adding New words to the JavaScript"
+ "dictionary:");
var re = /^\w+/";
GetJSTerms(this.bookmarkRoot, 0);
```

customDictionaryDelete

6.0			
-----	--	--	--

このメソッドは、`customDictionaryOpen` または `customDictionaryCreate` によって開かれたカスタム辞書ファイルを閉じて削除します。

注意：このメソッドを使用できるのは、コンソールイベントまたはバッヂイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

パラメータ

cName	削除する辞書の名前。これは、この辞書が開かれたとき、または作成されたときに使用された辞書名です。
-------	--

戻り値

成功した場合は `true`。失敗した場合は `false`。ファイルシステムへの適切なアクセス権がない場合、このメソッドは失敗します。

例

カスタム辞書を削除します。

```
spell.customDictionaryDelete("JavaScript");
```

customDictionaryExport

6.0			
-----	--	--	--

`spell` の `customDictionaryOpen` メソッドや `customDictionaryCreate` メソッドによって開かれたカスタム辞書を、新しいファイルに書き出します

書き出すディレクトリを指定するように求めるプロンプトが表示されます。カスタム辞書は、`customDictionaryCreate` で指定した辞書名と言語を使用して、`.clm` ファイルとして保存されます。例えば、作成時に指定した辞書名が「JavaScript」、言語が「en」の場合、書き出されるファイル名は `JavaScript-eng.clm` になります。

書き出されたカスタム辞書は、次に説明する `customDictionaryOpen` 呼び出しで使用することができます。

注意：このメソッドを使用できるのは、コンソールイベントまたはバッヂイベントのみです。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

パラメータ

cName	この辞書が開かれたとき、または作成されたときに使用された辞書名。
-------	----------------------------------

戻り値

成功した場合は `true`。失敗した場合は `false`。ファイルシステムへの適切なアクセス権がない場合、このメソッドは失敗します。

例

カスタム辞書を書き出します。書き出した辞書は、他のユーザに送信することができます ([customDictionaryCreate](#) の例を参照)。

```
spell.customDictionaryExport("JavaScript");
```

customDictionaryOpen

6.0			
-----	--	--	--

書き出されたカスタム辞書を、使用可能な辞書のリストに追加します。[customDictionaryExport](#) を参照してください。

注意：カスタム辞書ファイルを作成するには、`customDictionaryCreate` メソッドや `customDictionaryExport` メソッドを使用します。

Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cDIPath	デバイスに依存しない、カスタム辞書ファイルのパス。
cName	スペルチェックダイアログボックスで使用される辞書名。 <code>check</code> 、 <code>checkText</code> 、 <code>checkWord</code> メソッドの入力パラメータとして使用できます。
bShow	(オプション) <code>true</code> (デフォルト) の場合、 <code>cName</code> パラメータが「ユーザ:」の名前として、すべてのリストやメニューにこの名前が表示されます。例えば、 <code>cName</code> が「Test」の場合、「ユーザ: Test」がリストやメニューに追加されます。 <code>bShow</code> が <code>false</code> の場合、このカスタム辞書の名前はどのリストやメニューにも表示されません。

戻り値

成功した場合は `true`。失敗した場合は `false`。ファイルの読み取り権限がない場合、このメソッドは失敗します。

例

この例は、[customDictionaryCreate](#) の例や [customDictionaryExport](#) の例の続きです。書き出されたカスタム辞書を、使用可能な辞書のリストに追加します。

書き出されたカスタム辞書は、読み取り／書き込み権限のある任意のフォルダに配置してください。配置場所の選択肢としては、ユーザの `dictionaries` フォルダがあります。このフォルダの場所は、`app.getPath` メソッドで取得できます。

```
app.getPath("user", "dictionaries");
```

書き出された辞書を配置したら、フォルダレベルの JavaScript を追加して、この辞書を自動的にリストに追加することができます。ユーザの JavaScript フォルダのパスは、次のメソッドを実行することで取得できます。

```
app.getPath("user", "javascript");
```

このフォルダ内に .js ファイルを作成し、次の行を追加します。

```
var myDictionaries = app.getPath("user", "dictionaries");
spell.customDictionaryOpen( myDictionaries, "JavaScripts", true);
```

Acrobat を起動し直すと、「JavaScript」辞書が開かれ、使用可能になります。

ignoreAll

6.0			
-----	--	--	--

現在の文書のスペルチェック除外単語リストに単語を追加したり、このリストから単語を削除したりします。

注意：Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cWord	除外リストに追加または削除する単語。
bIgnore	(オプション) <code>true</code> (デフォルト) の場合、文書の除外単語リストに単語が追加されます。 <code>false</code> の場合は、除外リストから単語が削除されます。

戻り値

成功した場合は `true`。開かれている文書がない場合は例外が発生します。

例

```
var bIgnored = spell.ignoreAll("foo");
if (bIgnored) console.println("¥"foo¥" will be ignored);
```

removeDictionary

X	P		X
---	---	--	---

注意：Acrobat 6.0 以降、このメソッドはサポートされなくなりました。このメソッドの戻り値は、常に `false` です。[customDictionaryClose](#) メソッドを使用してください。

`addDictionary` によって追加されたユーザ辞書を削除します。

パラメータ

cName	削除する辞書の名前。 <code>addDictionary</code> で使用された名前と同じである必要があります。
-------	--

戻り値

false

removeWord

5.0	P		
-----	---	--	--

辞書から単語を削除します。customDictionaryCreate または customDictionaryExport によって作成されたユーザ辞書から単語を削除することはできません。

[addWord](#) も参照してください。

注意： 内部的には、spell オブジェクトによってユーザ辞書がスキャンされ、指定した単語がそこに含まれている場合はその単語が削除されます。単語がこの辞書に含まれていない場合は、ユーザ辞書「Not-A-Word」に追加されます。実際の辞書は変更されません。

Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cWord	削除する単語。
cName	(オプション) 辞書名または言語コード。インストールされている辞書名の配列は、dictionaryNames または languages で取得できます。

戻り値

成功した場合は true、それ以外の場合は false。

userWords

5.0			
-----	--	--	--

ユーザが辞書に追加した単語の配列、または削除した単語の配列を取得します。[addWord](#) および [checkWord](#) も参照してください。

注意： Adobe Reader の場合、このプロパティは、バージョン 7.0 以降でのみ使用可能です。

パラメータ

cName	(オプション) 辞書名または言語コード。インストールされている辞書名の配列は、dictionaryNames または languages で取得できます。cName を指定しなかった場合は、デフォルトの辞書が使用されます。デフォルトの辞書は、「スペルチェック」環境設定ダイアログボックスで指定されている最初の辞書です。
bAdded	(オプション) true の場合はユーザが追加した単語の配列を返し、false の場合はユーザが削除した単語の配列を返します。デフォルトは true です。

戻り値

ユーザが追加または削除した単語の配列。

例

「JavaScript」辞書に追加された単語をリストします（[customDictionaryCreate](#) の例を参照）。

```
var aUserWords = spell.userWords({cName: "JavaScript"});  
aUserWords.toSource();
```

Statement

5.0			X
-----	--	--	---

このオブジェクトは、SQL で更新やクエリを実行して、その結果を取得するときに使用します。Statement オブジェクトを作成するには、`connection.newStatement` を使用します。

次の事項も参照してください。

- [Connection](#) オブジェクト
- [ADBC](#) オブジェクト
- [Column](#) オブジェクト、[ColumnInfo](#) オブジェクト、[Row](#) オブジェクト、[TableInfo](#) オブジェクト

Statement のプロパティ

columnCount

クエリで返される行に含まれている列数。更新操作の場合は未定義になります。

型

数値

アクセス

R

rowCount

更新によって影響を受ける行数。クエリで返される行数ではありません。クエリのコンテキストでは値が未定義になります。

型

数値

アクセス

R

Statement のメソッド

execute

Statement オブジェクトのコンテキストで SQL 文を実行します。失敗すると例外が発生します。

注意： execute が失敗せず、すべてのデータが返されなかつたとしても、それは文が正しいという保証にはなりません。

パラメータ

cSQL	実行する SQL 文。
------	-------------

例

execute メソッドの使用テクニックをいくつか示します。

ClientData データベースのすべてのフィールドを選択します。

```
statement.execute("Select * from ClientData");
```

データベーステーブルや列の名前にスペースが含まれている場合は、エスケープした引用符でこれらの名前を囲む必要があります。例えば、次のようになります。

```
var execStr1 = "Select firstname, lastname, ssn from \"Employee Info\"";  
var execStr2 = "Select \"First Name\" from \"Client Data\"";  
statement.execute(execStr1);  
statement.execute(execStr2);
```

SQL 文字列全体を一重引用符で囲めば、より簡潔に記述できます。これにより、テーブル名や列名を二重引用符で囲むことができます。

```
var execStr3 = 'Select "First Name", "Second Name" from "Client Data" ';  
statement.execute(execStr3);
```

詳しい例については、[getRow](#) および [nextRow](#) を参照してください。

getColumn

指定した列のデータを表す Column オブジェクトを取得します。

注意： これらのメソッドの 1 つで列を取得した後、再度同じ列を取得しようとすると、失敗する場合があります。

パラメータ

nColumn	データを取得する列。列番号または（列の名前を表す）文字列を指定できます（ColumnInfo オブジェクトを参照）。
---------	--

nDesiredType	（オプション）列のデータを表すのに最適な ADBC の JavaScript 型。
--------------	---

戻り値

指定された列のデータを表す `Column` オブジェクト。失敗した場合は `null`。

getColumnArray

結果セットに含まれる各列を表す `Column` オブジェクトの配列を取得します。列のデータを表すのに最適な ADBC の JavaScript 型は、推測によって決定されます。

注意：これらのメソッドの 1 つで列を取得した後、再度同じ列を取得しようとすると、失敗する場合があります。

戻り値

`Column` オブジェクトの配列。失敗した場合は `null`、または長さ 0 の配列。

getRow

現在の行を表す `Row` オブジェクトを取得します。このオブジェクトには、各列の情報が含まれています。`getColumnArray` と同様、列のデータ型は推測によって決定されます。

`getRow` を呼び出す前に、`nextRow` を呼び出す必要があります。`nextRow` を呼び出さずに `getRow` を 2 回続けて呼び出すと、2 回目の `getRow` の呼び出しでは `null` が返されます。

戻り値

`Row` オブジェクト。

例 1

すべての `Row` オブジェクトには、データ行の各列のプロパティが含まれています。次に例を示します。

```
var execStr = "SELECT firstname, lastname, ssn FROM ¥"Employee Info¥"";  
statement.execute(execStr);  
statement.nextRow();  
row = statement.getRow();  
console.println("The first name of the first person retrieved is: "  
    + row.firstname.value);  
console.println("The last name of the first person retrieved is: "  
    + row.lastname.value);  
console.println("The SSN of the first person retrieved is: "  
    + row.ssn.value);
```

例 2

列名にスペースが含まれている場合、前述の構文（`row.firstname.value` など）では行プロパティにアクセスできません。代わりに、次のようにします。

```
Connect = ADBC.newConnection("Test Database");
statement = Connect.createStatement();
var execStr = 'Select "First Name", "Second Name" from "Client Data" ';
statement.execute(execStr);
statement.nextRow();

// この PDF ファイルに値を指定
this.getField("name.first").value = row["First Name"].value;
this.getField("name.last").value = row["Second Name"].value;
```

nextRow

前に実行されたクエリで生成されたデータの次の行のデータを取得します。`execute` で結果セットを得た後に最初の行を取得するには、このメソッドを呼び出す必要があります。

戻り値

なし。（次の行がないなどの理由で）失敗した場合は例外が発生します。

例

2つのボタンと文書レベルの JavaScript を作成し、データベースから取得したデータを PDF フォームに入力する簡単な例を示します。

以下で定義している `getNextRow` ボタンでは、（次の行が存在しないために）例外が発生しない限り、`nextRow` メソッドを使用してデータベースから次の行を取得します。例外が発生した場合はデータベースに再接続し、`nextRow` を使用してデータの最初の行を（再度）取得します。

```
/* ボタンスクリプト */
// getConnected ボタン
if (getConnected())
    populateForm(statement.getRow());

// getNextRow ボタン
try {
    statement.nextRow();
} catch(e) {
    getConnected();
}
var row = statement.getRow();
populateForm(row);

/* 文書レベルの JavaScript */
// getConnected() 文書レベルの JavaScript
function getConnected()
{
    try {
        ConnectADBCdemo = ADBC.newConnection("ADBCdemo");
        if (ConnectADBCdemo == null)
```

```
        throw "Could not connect";
        statement = ConnectADBCdemo.newStatement();
        if (statement == null)
            throw "Could not execute newStatement";
        if (statement.execute("Select * from ClientData"))
            throw "Could not execute the requested SQL";
        if (statement.nextRow())
            throw "Could not obtain next row";
        return true;
    } catch(e) {
        app.alert(e);
        return false;
    }
}
// populateForm()
/* データベースの行のデータを、PDF ファイル内の対応するテキストフィールドにマッピング。*/
function populateForm(row)
{
    this.getField("firstname").value = row.FirstName.value;
    this.getField("lastname").value = row.LastName.value;
    this.getField("address").value = row.Address.value;
    this.getField("city").value = row.City.value;
    this.getField("state").value = row.State.value;
    this.getField("zip").value = row.Zipcode.value;
    this.getField("telephone").value = row.Telephone.value;
    this.getField("income").value = row.Income.value;
}
```

TableInfo

この汎用 JavaScript オブジェクトには、テーブルの基本情報が含まれています。このオブジェクトは、`connection.getTableList` によって返されます。含まれているプロパティは、次のとおりです。

プロパティ	型	アクセス	説明
<code>name</code>	文字列	R	テーブルの識別名。この文字列を SQL 文で使用して、 <code>TableInfo</code> オブジェクトが関連付けられているテーブルを識別することができます。
<code>description</code>	文字列	R	テーブルに関するデータベース固有の情報を含む文字列。

Template

Template オブジェクトは、文書内の名前付きのページです。テンプレートページは表示／非表示を切り替えることができ、コピーを作成したり、他のページに適用したりすることができます。Template オブジェクトは、一般に、コンテンツを動的に作成するために使用します（例えば、ページが足りなくなったら自動的に追加される請求書など）。

Doc の [templates](#) プロパティと、[createTemplate](#)、[getTemplate](#)、[removeTemplate](#) の各メソッドも参照してください。

Template のプロパティ

hidden

5.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
-----	-------------------------------------	--------------------------	-------------------------------------

テンプレートを非表示にするかどうかを指定します。非表示にされたテンプレートは、そのコピーが作成されるか非表示が解除されるまで、ユーザに表示されることはありません。非表示のテンプレートを表示させた場合は、文書の最後に追加されます。

Adobe Reader では、このプロパティは次のように動作します。

- 5.1 より前のバージョンの Adobe Reader では、このプロパティを設定すると、例外が発生します。
- Adobe Reader 5.1 と 6.0 では、拡張フォーム機能文書権限が付与されていれば、このプロパティを設定できます。
- Adobe Reader 7.0 では、いかなる場合も、このプロパティは設定できません。

型

ブーリアン

アクセス

R / W

name

5.0	<input type="text"/>	<input type="text"/>	<input type="text"/>
-----	----------------------	----------------------	----------------------

テンプレートの作成時に指定されたテンプレート名。

型

文字列

アクセス

R

Template のメソッド

spawn

5.0	D		F
-----	---	--	---

テンプレートに基づいて、文書に新しいページを作成します。

パラメータ

nPage	(オプション) ページ番号のインデックス (0 から数えます)。bOverlay の値に応じて、このページの前またはこのページに重ねて、新しいページが作成されます。デフォルトは 0 です。
bRename	(オプション) ページ上のフォームフィールドの名前を変更するかどうかを指定します。デフォルトは true です。
bOverlay	(オプション) true (デフォルト) の場合、テンプレートは指定のページ上に重ねられます。false の場合は、指定のページの前に新しいページとして挿入されます。 ページを文書の最後に追加するには、bOverlay を false に設定し、nPage を文書のページ数に設定します。
<p>注意：証明済み文書や、「拡張フォーム機能権限」が付与されている文書では、bOverlay パラメータは無効となります。これらのタイプの文書ではテンプレートを重ねることはできません。</p>	
oXObject	(オプション、Acrobat 6.0) このパラメータの値には、spawn を前回呼び出したときの戻り値を指定します。

戻り値

6.0 より前のバージョンの Acrobat では、このメソッドは何も返しませんでした。現在、spawn は、作成されたページのページコンテンツを表すオブジェクトを返します。返されたオブジェクトは、spawn を再度呼び出すときに、オプションパラメータ oXObject の値として使用できます。

注意：同じページを繰り返し作成すると、ファイルサイズが非常に大きくなります。この問題を回避するために、spawn は作成したページのページコンテンツを表すオブジェクトを返すようになりました。spawn メソッドを再度呼び出すときに、この戻り値を oXObject パラメータの値として使用すれば、同じページを作成することができます。

例 1

すべてのテンプレートのコピーを作成し、それを 1 つずつ文書の最後に追加します。

```
var a = this.templates;
for (i = 0; i < a.length; i++)
  a[i].spawn(this.numPages, false, false);
```

例 2 (Acrobat 6.0)

`oXObject` パラメータと戻り値を使用して、同じテンプレートをもとに、ページを 31 回作成します。この手法を使用すると、ファイルサイズが大きくなりすぎることを回避できます。

```
var t = this.templates;
var T = t[0];
var XO = T.spawn(this.numPages, false, false);
for (var i=0; i<30; i++) T.spawn(this.numPages, false, false, XO);
```

Thermometer



このオブジェクトは、ステータスウィンドウとプログレスバーを組み合わせたもので、時間のかかる処理が進行中であることを示します。Thermometer オブジェクトを取得するには、`app.thermometer` を使用します。

例

Thermometer オブジェクトのすべてのプロパティとメソッドの使用例を次に示します。

```
var t = app.thermometer;           // thermometer オブジェクトを取得
t.duration = this.numPages;
t.begin();
for ( var i = 0; i < this.numPages; i++)
{
    t.value = i;
    t.text = "Processing page " + (i + 1);
    if (t.cancelled) break;          // 処理がキャンセルされた場合は停止
    ... process the page ...
}
t.end();
```

Thermometer のプロパティ

cancelled

進行中の処理をユーザがキャンセルできるかどうかを指定します。処理をキャンセルするには、Esc キーを押すか（Windows および UNIX プラットフォームの場合）、Command + ピリオドキーを押します（Macintosh プラットフォームの場合）。

型

ブーリアン

アクセス

R

duration

最大値に達した状態の thermometer 表示に対応する値を設定します。以降、thermometer は value を設定することによって目盛りが増えています。デフォルトの duration は 100 です。

型

数値

アクセス

R / W

text

thermometer に表示するテキスト文字列を設定します。

型

文字列

アクセス

R / W

value

thermometer の現在の値を設定し、表示を更新します。使用できる値の範囲は、0 (空) から duration で設定した値までです。例えば、thermometer の duration が 10 の場合、現在の値は 0 ~ 10 である必要があります。値が 0 未満の場合は、0 に設定されます。値が duration よりも大きい場合は、duration に設定されます。

型

数値

アクセス

R / W

Thermometer のメソッド

begin

thermometer を初期化し、現在の値を duration のパーセントとして表示します。

例

現在の文書の各ページにある単語をカウントします。現在までの合計を通知すると同時に、thermometer を使用して進行状況を示します。

```
var t = app.thermometer; // thermometer オブジェクトを取得
t.duration = this.numPages;
t.begin();
var cnt=0;
for ( var i = 0; i < this.numPages; i++)
{
    t.value = i;
    t.text = "Processing page " + (i + 1);
    cnt += getPageNumWords(i);
    console.println("There are " + cnt + "words in this doc.");
    if (t.cancelled) break;
}
t.end();
```

end

thermometer の現在の値を duration（最大値に達した状態）に設定して thermometer を描画し、thermometer の表示を終了します。

this

JavaScript の特殊なキーワードである `this` は、現在のオブジェクトを指します。Acrobat では、現在のオブジェクトは次のように定義されます。

- オブジェクトのメソッドでは、そのメソッドが属しているオブジェクト。
- コンストラクタ関数では、構築中のオブジェクト。
- 文書レベルのスクリプトまたはフィールドレベルのスクリプトでは、`Doc` オブジェクト。このオブジェクトを使用して、文書のプロパティや関数を設定したり取得したりすることができます。
- フォルダレベルの JavaScript ファイルで定義されている関数では、未定義。これらの関数で `Doc` を必要とする場合は、呼び出し側の関数から `Doc` を渡してください。

例えば、次の関数が Plug-in フォルダレベルで定義されているとします。

```
function PrintPageNum(doc)
{
    /* 現在のページ番号をコンソールに表示。*/
    console.println("Page = " + doc.pageNum);
}
```

次のスクリプトを実行すると、現在のページ番号がコンソールに 2 回出力された後、そのページが印刷されます。

```
/* Doc を渡す必要がある。*/
PrintPageNum(this);
/* 前の呼び出しと同じ。*/
console.println("Page = " + this.pageNum);
/* 現在のページを印刷。*/
this.print(false, this.pageNum, this.pageNum);
```

スクリプトで定義されている変数や関数は、`this` オブジェクトを親とします。例えば、次のようにになります。

```
var f = this.getField("Hello");
```

これは、次のコードと同じです。

```
this.f = this.getField("Hello");
```

ただし、この場合の変数 `f` は、スクリプトの実行後にガーベッジコレクトされる可能性がある点が異なります。

変数名と関数名の競合

`Doc` のプロパティ名やメソッド名を、変数名に使用しないでください。次のように、予約語 `var` の後にメソッド名を使用すると、例外が発生します。

```
var getField = 1; // 「TypeError: redeclaration of function getField」と表示される
```

プロパティ名を使用しても例外は発生しませんが、プロパティがオブジェクトを参照している場合は、プロパティの値を変更できないことがあります。

```
// 「title」は「1」を返すが、文書は「1」という名前になる。
var title = 1;
```

```
// プロパティは変更されず、info はオブジェクトのまま
var info = 1; // 「info」は [object Info] を返す
```

次に、変数名の競合を避ける例を示します。

```
var f = this.getField("mySignature"); // ppklite 署名ハンドラを使用  
// 「info」ではなく「Info」を使用して競合を回避  
var Info = f.signatureInfo();  
  
// いくつかの標準の signatureInfo プロパティ  
console.println("name = " + Info.name);
```

TTS

4.05			
------	--	--	--

JavaScript の TTS オブジェクトを使用すると、テキストを音声に変換することができます。TTS オブジェクトを使用するには、ユーザのコンピュータに Text-To-Speech エンジンがインストールされている必要があります。Text-To-Speech エンジンは、テキストをデジタルオーディオに変換して、テキストを読み上げます。ほとんどの場合、Text-To-Speech エンジンはアクセシビリティを提供することを目的としていますが、そのほかにも様々な応用が可能です。

現時点では、このオブジェクトは Windows 専用の機能です。これを使用するには、オペレーティングシステムに Microsoft Text-to-Speech エンジンがインストールされている必要があります。

TTS オブジェクトは JavaScript オブジェクトなので、Windows と Macintosh のどちらのプラットフォームにも存在しますが、Macintosh では使用できません。

注意：Acrobat 5.0 で、アクセシビリティに関する仕様が大幅に変更され、一般的なスクリーンリーダを直接統合する方法が採用されました。このため、バージョン 4.05 で定義されていた TTS オブジェクトのスクリーンリーダ機能のいくつかは、スクリーンリーダと重複するので、バージョン 5.0 で削除されました。ただし、マルチメディア文書で役立つ機能もあるので、TTS オブジェクトは当面残されています。

TTS のプロパティ

available

TTS オブジェクトと Text-To-Speech エンジンが使用可能である場合、true になります。

型

ブーリアン

アクセス

R

例

```
console.println("Text to speech available: " + tts.available);
```

numSpeakers

Text-To-Speech エンジンで使用可能なスピーカーの数。[speaker](#) プロパティおよび [getNthSpeakerName](#) × ソッドも参照してください。

型

整数

アクセス

R

pitch

スピーカの音声のベースラインピッチを設定します。ピッチの有効範囲は 0 ~ 10 で、デフォルトは 5 です。

型

整数

アクセス

R / W

soundCues



非推奨になりました。`false` が返されます。

型

ブーリアン

アクセス

R / W

speaker

Text-To-Speech を実行するときに、音質の異なるスピーカをユーザが指定できるようにします。
[numSpeakers](#) プロパティおよび [getNthSpeakerName](#) メソッドも参照してください。

型

文字列

アクセス

R / W

speechCues

X			
---	--	--	--

非推奨になりました。`false` が返されます。

型

ブーリアン

アクセス

R / W

speechRate

Text-To-Speech エンジンでテキストを読み上げる速度を設定します。`speechRate` の値は、1 分あたりの単語数を表します。

型

整数

アクセス

R / W

volume

音量を設定します。有効な値は、0（ミュート）～10（最大）です。

型

整数

アクセス

R / W

TTS のメソッド

getNthSpeakerName

Text-To-Speech エンジンで使用可能な `n` 番目のスピーカーの名前を取得します ([numSpeakers](#) および [speaker](#) プロパティも参照)。

パラメータ

nIndex	目的のスピーカ名のインデックス。
--------	------------------

戻り値

指定されたスピーカの名前。

例

使用可能なすべてのスピーカを列挙します。

```
for (var i = 0; i < tts.numSpeakers; i++) {  
    var cSpeaker = tts.getNthSpeakerName(i);  
    console.println("Speaker[" + i + "] = " + cSpeaker);  
    tts.speaker = cSpeaker;  
    tts.qText ("Hello");  
    tts.talk();  
}
```

pause

TTS オブジェクトの Text-To-Speech の出力を一時停止します。待ち状態になっている残りのテキストを再生するには、`resume` を呼び出します。

qSilence

一定時間の無音をテキストのキューに入れます。

パラメータ

nDuration	無音時間（ミリ秒単位）。
-----------	--------------

qSound

指定のサウンドをキューに入れて、`talk` で再生できるようにします。パラメータ `cSound` には、有効なサウンドキュー名のリストから 1 つを指定します。これらの名前は、`SoundCues` フォルダに保存されているサウンドファイル（存在する場合）に直接マッピングされます。

```
tts.qSound("DocPrint"); // DocPrint.wav を再生
```

`SoundCues` フォルダは、ビューアのプログラムレベルに存在している必要があります（例えば、`C:\Program Files\Adobe\Acrobat 5.0\SoundCues` など）。

注意：このメソッドは Windows 専用です。`qSound` に指定できるのは、22 KHz、16 ビットでエンコードされた PCM の `.wav` ファイルのみです。また、MS SAPI のキューの遅延問題を避けるために、1 秒以上のサウンドであることが必要です。1 秒未満の場合は、サウンドを編集して最後に無音を追加してください。

パラメータ

cSound	使用するサウンドキューの名前。
--------	-----------------

qText

テキストをキューに入れて、talk で再生できるようにします。

パラメータ

cText	音声に変換するテキスト。
-------	--------------

例

```
tts.qText("Hello, how are you?");
```

reset

キューに入っているテキストの再生を停止し、キューをクリアします。また、TTS オブジェクトのプロパティをすべてデフォルト値にリセットします。resume でテキストの再生を再開することはできません。

resume

一時停止した TTS オブジェクトのテキストの再生を再開します。

stop

キューに入っているテキストの再生を停止し、キューをクリアします。resume でテキストの再生を再開することはできません。

talk

キューの内容を音声にするために Text-To-Speech エンジンに送ります。テキストの音声出力が一時停止されていた場合は、talk によってキューのテキストの再生が再開されます。

例

```
tts.qText("Hello there!");  
tts.talk();
```

util

静的な JavaScript オブジェクト。文字列や日付をフォーマットしたり解析したりするための、様々なユーティリティメソッドや便利な関数が定義されています。

util のメソッド

crackURL

7.0.5			
-------	--	--	--

URL を個々の構成要素に分割します。

パラメータ

cURL URL を指定する文字列。

戻り値

次のプロパティを含むオブジェクト。

プロパティ	説明
cScheme	URL のスキーム。file、http、https のいずれかです。
cUser	(オプション) URL で指定されているユーザ名。
cPassword	(オプション) URL で指定されているパスワード。
cHost	URL のホスト名。
nPort	URL のポート番号。
cPath	(オプション) URL のパス部分。
cParameters	(オプション) URL のパラメータ文字列の部分。
cFragments	(オプション) URL のフラグメント。

パラメータが指定されていない場合や、有効な形式の URL でない場合、または URL のスキームが file、http、https のいずれでもない場合は、パラメータエラーが発生します。

例

次のコードを実行すると、

```
util.crackURL("http://example.org/myPath?name0=value0&name1=value1#frag");
```

戻り値は次のようになります。

```
{  
    cScheme: "http",  
    cHost: "example.org",  
    nPort: 80,  
    cPath: "/myPath",  
    cParameters: "?name0=value0&name1=value1",  
    cFragments: "frag"  
}
```

iconStreamFromIcon

7.0			
-----	--	--	--

XObject ベースの Icon オブジェクトを Icon Stream オブジェクトに変換します。

パラメータ

oIcon Icon Stream オブジェクトに変換する Icon オブジェクト。

戻り値

Icon Stream オブジェクト

このメソッドを使用すれば、Doc の importIcon メソッドや getIcon メソッドで取得したアイコンを、`app.addToolBar` などのメソッド（パラメータに Icon Stream を指定する必要があるメソッド）で使用できるようになります。

例

文書レベルの名前付きアイコンツリーにアイコンを取り込んで、アプリケーションにツールボタンを追加します。

```
this.importIcon("myIcon", "/C/temp/myIcon.jpg", 0);  
var oIcon = util.iconStreamFromIcon(this.getIcon("myIcon"));  
app.addButton({  
    cName: "myButton",  
    oIcon: oIcon,  
    cExec: "console.println('My Button!');",  
    cTooltext: "My button!",  
    nPos: 0  
});
```

printd

3.01			
------	--	--	--

指定のフォーマットで日付を返します。

パラメータ

cFormat	日時のフォーマット。次のいずれかのタイプを使用できます。
	<ul style="list-style-type: none">日時データのプレースホルダとしてサポートされているサブ文字列のパターンを指定した文字列。認識可能な日時文字列については、後述の表を参照してください。Acrobat 5.0 以降の場合、フォーマットを指定する数値。サポートされている値（とフォーマットの例）は、次のとおりです。<ul style="list-style-type: none">0 — PDF 日付フォーマット。例：D:20000801145605+07'00'1 — ユニバーサル。例：D:20000801145605+07'00'2 — ローカライズされた文字列。例：2000/08/01 14:56:05Acrobat 7.0 以降では、<code>bXFAPicture</code> を <code>true</code> にすると、XFA Picture 節のフォーマットを使用してこのパラメータが解釈されます。
oDate	フォーマットする日付オブジェクト。日付オブジェクトは、コア JavaScript の <code>Date</code> コンストラクタや、 <code>util.scand</code> メソッドから取得できます。
bXFAPicture	(オプション、Acrobat 7.0) XFA Picture 節のフォーマットを使用して <code>cFormat</code> の値を解釈するかどうかを指定するブーリアン値。XFA Picture 節では、ローカライズされた日時が幅広くサポートされています。詳しくは、『XFA Specification, Version 2.2』の日時のフォーマットに関する節を参照してください（ 28 ページの「関連ドキュメント」 を参照）。
	デフォルトは <code>false</code> です。

戻り値

フォーマットされた日付文字列。

cFormat 文字列のパターン

文字列	説明	例
mmmm	月の名称	September
mmm	月の略称	Sep
mm	月を表す先行ゼロ付きの数値	09
m	月を表す先行ゼロのない数値	9
dddd	曜日の名称	Wednesday
ddd	曜日の略称	Wed

文字列	説明	例
dd	先行ゼロ付きの日	03
d	先行ゼロのない日	3
yyyy	4桁の年	1997
yy	2桁の年	97
HH	先行ゼロ付きの 24 時間制の時間	09
H	先行ゼロのない 24 時間制の時間	9
hh	先行ゼロ付きの 12 時間制の時間	09
h	先行ゼロのない 12 時間制の時間	9
MM	先行ゼロ付きの分	08
M	先行ゼロのない分	8
ss	先行ゼロ付きの秒	05
s	先行ゼロのない秒	5
tt	am / pm	am
t	1文字で表した am / pm	a
j	日本の元号（略称）	
注意： Acrobat 6.0 で導入されましたが、Acrobat 7.0 でこのフォーマット文字列は非推奨になりました。XFA Picture 節のフォーマットを使用してください。		
jj	日本の元号	
注意： Acrobat 6.0 で導入されましたが、Acrobat 7.0 でこのフォーマット文字列は非推奨になりました。XFA Picture 節のフォーマットを使用してください。		
¥	エスケープ文字として使用	

例 1

現在の日付を省略しない形式でフォーマットします。

```
var d = new Date();
console.println("Today is " + util.printd("mmmm dd, yyyy", d));
```

例 2 (Acrobat 5.0)

ローカルフォーマットで日付を表示します

```
console.println(util.printd(2, new Date()));
```

例 3 (Acrobat 7.0)

XFA Picture 節を使用して、現在の日付をコンソールに出力します。

```
// コンソールで実行
console.println(
    util.printd("EEE, 'the' D 'of' MMMM, YYYY", new Date(), true));
// 出力例
Tue, the 13 of July, 2004
```

Picture 節はロケールを判別します。通常、picture 節は、その環境のロケール内で処理されます。ただし、picture を特定のロケールで処理するように指定することは可能です。この機能は、その環境のロケールとは異なる特定のロケールのデータのフォーマットまたは解析を行うときに役立ちます。複合 picture 節に対するこの拡張機能の構文は次のとおりです。

```
category-name(locale-name){picture-symbols}
```

次のコードをコンソールで実行します。

```
util.printd("date(fr){DD MMMM, YYYY}", new Date(), true)
```

この日の出力は次のとおりです。

```
13 juillet, 2004
```

XFA-Picture 節では、簡体字中国語、繁体字中国語、日本語および韓国語（CCJK）の時刻と日付が幅広くサポートされています。次に示す、テキストフィールドのカスタムフォーマットスクリプトの例では、現在の日付が日本語ロケール用にフォーマットされて表示されます。

```
event.value = util.printd("date(ja){ggYY/M/D}", new Date(), true)
```

printf

3.01			
------	--	--	--

1つ以上の引数を、フォーマット文字列に従って文字列としてフォーマットします。同名の C 関数に似ています。このメソッドは、フォーマット文字列（cFormat）に従って入力引数を変換およびフォーマットし、その結果を文字列として返します。

フォーマット文字列は、2種類のオブジェクトで構成されています。

- 通常の文字。結果の文字列にコピーされます。
- 変換仕様。各仕様が printf の 2番目以降の引数に対応し、変換およびフォーマットを行います。

各変換仕様は、次のように構成されています。

```
%[,nDecSep][cFlags][nWidth][.nPrecision]cConvChar
```

次の表に、変換仕様のコンポーネントを示します。

nDecSep	カンマ (,) と、小数点／区切り文字のフォーマットを指定する次のいずれかの数値。 0 — 区切り文字にカンマ、小数点にピリオドを使用します。 1 — 区切り文字はなく、小数点にピリオドを使用します。 2 — 区切り文字にピリオド、小数点にカンマを使用します。 3 — 区切り文字はなく、小数点にカンマを使用します。
cFlags	数値変換にのみ有効で、フォーマットを定義するいくつかの文字で構成されています (順序は任意)。 + — 数値を常に符号付きでフォーマットします。 space — 最初の文字が符号でない場合は、前に 1 つのスペースを置きます。 0 — フィールドの空白の桁をゼロで埋めます。 # — 代替出力形式を指定します。f (浮動小数点数) の場合、常に出力に小数点を付けます。
nWidth	フィールドの最小幅を指定する数値。符号と小数点を含め、最低でもここで指定した文字数にフォーマットします。必要に応じてそれ以上の文字数になることもあります。変換後の引数の文字数がフィールド幅より少ない場合は、値の左側を埋めてフィールド幅になるようにします。埋め込み文字には通常はスペースが使用されますが、ゼロの埋め込みフラグが指定されている場合 (cFlags に 0 が含まれている場合) は 0 を使用します。
nPrecision	ピリオド (.) と、小数点以下の桁数を指定する数値 (浮動小数点数に変換する場合)。
cConvChar	引数の解釈方法を指定します。 a — 整数 (必要に応じて切り捨てられます) f — 浮動小数点数 s — 文字列 x — 符号なしの 16 進表記の整数 (必要に応じて切り捨てられます)

パラメータ

cFormat	使用するフォーマット文字列。
arguments	最初のパラメータ (フォーマット文字列) で指定されている % タグの部分に代入するデータを表す、オプションの引数。オプションの引数の数は、% タグの数と同じである必要があります。

注意：util.printf 関数では、プロパティで引数を指定するオブジェクトリテラルは使用できません。引数は、通常のカンマで区切ったリスト形式で入力します。

戻り値

指定どおりにフォーマットされた結果文字列。

例

無理数を渡し、util.printf の各フォーマットを使用して表示します。

```
var n = Math.PI * 100;  
console.clear();  
console.show();  
console.println(util.printf("Decimal format: %d", n));  
console.println(util.printf("Hex format: %x", n));  
console.println(util.printf("Float format: %.2f", n));  
console.println(util.printf("String format: %s", n));
```

このスクリプトの出力は、次のようにになります。

```
Decimal format: 314  
Hex format: 13A  
Float format: 314.16  
String format: 314.159265358979
```

printx

3.01			
------	--	--	--

フォーマット文字列 `cFormat` に従って、ソース文字列 `cSource` をフォーマットします。`cFormat` のフォーマット文字列には、どのような文字列でも指定できます。次に示す特殊なマスク文字が使用できます。

値	説明
?	次の文字をコピーします。
X	次の英数字をコピーし、他の文字はスキップします。
A	次の英字をコピーし、他の文字はスキップします。
9	次の数値をコピーし、他の文字はスキップします。
*	これ以降の残りのソース文字列をコピーします。
¥	エスケープ文字。
>	次の指示があるまで、大文字に変換します。
<	次の指示があるまで、小文字に変換します。
=	次の指示があるまで、大文字／小文字を保持します（デフォルト）。

パラメータ

`cFormat` 使用するフォーマット文字列。

`cSource` 使用するソース文字列。

戻り値

フォーマットされた文字列。

例

文字列を米国の電話番号としてフォーマットします。

```
var v = "aaa14159697489zzz";
v = util.printx("9 (999) 999-9999", v);
console.println(v);
```

scand

4.0			
-----	--	--	--

フォーマット文字列のルールに基づいて、日付を JavaScript の日付オブジェクトに変換します。この関数は、日付コンストラクタを直接使用する場合に比べて、柔軟性に富んでいます。

注意： scand に 2 衔の年を与えた場合、あいまいさの解決には Date Horizon という手法が使用されます。50 未満の場合は 21 世紀（2000 を加える）であると見なされ、50 以上の場合は 20 世紀（1900 を加える）と見なされます。

指定の日付 cDate には、cFormat と同じフォーマットを使用する必要があります。

パラメータ

cFormat	日付のフォーマットに使用するルール。cFormat には、printd の場合と同じ構文が使用されます。
cDate	変換する日付。

戻り値

変換された Date オブジェクトまたは null（変換に失敗した場合）。

例 1

```
/* 現在の日付を文字列にする。*/
var cDate = util.printd("mm/dd/yyyy", new Date());
console.println("Today's date: " + cDate);
/* 解析して日付に戻す。*/
var d = util.scand("mm/dd/yyyy", cDate);
/* 逆順で出力。*/
console.println("Yet again: " + util.printd("yyyy mmm dd", d));
```

例 2

テキストフィールドの値を取得し、正しいフォーマットの日付であるかどうか確認し、警告ボックスまたはコンソールウィンドウに結果を表示します。

このメソッドは、変換に失敗した場合には `null` を返します。これは、不正なデータが入力された場合に発生する可能性があります。したがって、戻り値が `null` であるかどうかテストしています。

```
var d= util.scand("mm/dd/yyyy", this.getField("myDate").value);
if ( d== null )
    app.alert("Please enter a valid date of the form" +
              " $"mm/dd/yyyy$".")
else {
    console.println("You entered the date: "
                  + util.printd("mmmm dd, yyyy",d));
}
```

spansToXML

6.0			
-----	--	--	--

Span オブジェクトの配列を、XML (XFA) 文字列 (『PDF Reference』バージョン 1.7 を参照) に変換します。

パラメータ

Span オブジェクトの配列

XML 文字列に変換する Span オブジェクトの配列。

戻り値

文字列

例

リッチテキストフィールドの値を取得し、すべてのテキストを青色に変更し、XML 文字列に変換してからコンソールに出力します。

```
var f = getField("Text1");
var spans = f.richValue;
for(var index = 0; index < spans.length; index++)
    spans[index].textColor = color.blue;
console.println(util.spansToXML(spans));
```

streamFromString

7.0			
-----	--	--	--

文字列を `ReadStream` オブジェクトに変換します。

パラメータ

cString	ReadStream オブジェクトに変換する文字列。
cCharSet	(オプション) cString で指定した文字列のエンコーディング。utf-8、utf-16、Shift-JIS、BigFive、GBK、UHC のいずれかを指定できます。デフォルトは utf-8 です。

戻り値

ReadStream オブジェクト

例

この文書のテキストフィールドに指定されているテキストを取得して、添付文書にそのテキストを追加します。

```
var v = this.getField("myTextField").value;
var oFile = this.getDataObjectContents("MyNotes.txt");
var cFile = util.stringFromStream(oFile, "utf-8");
cFile += "¥r¥n" + cFile;
oFile = util.streamFromString(cFile, "utf-8");
this.setDataObjectContents("MyNotes.txt", oFile);
```

この例では、Doc の `getDataObjectContents` および `setDataObjectContents` メソッドと、`util.stringFromStream` を使用しています。

stringFromStream

7.0			
-----	--	--	--

ReadStream オブジェクトを文字列に変換します。

パラメータ

oStream	文字列に変換する ReadStream オブジェクト。
cCharSet	(オプション) oStream で指定した文字列のエンコーディング。utf-8、utf-16、Shift-JIS、BigFive、GBK、UHC のいずれかを指定できます。デフォルトは utf-8 です。

戻り値

文字列

例

この文書には、テキストファイルが埋め込まれていると仮定します。この例では、添付ファイルの内容を読み込んで、それを複数行のテキストフィールドに表示します。

```
var oFile = this.getDataObjectContents("MyNotes.txt");
var cFile = util.stringFromStream(oFile, "utf-8");
this.getField("myTextField").value = cFile;
```

この例では、`getDataObjectContents` を使用して、添付文書のファイルストリームを取得しています。

xmlToSpans

6.0			
-----	--	--	--

XML (XFA) 文字列 (『PDF Reference』バージョン 1.7 を参照) を、Span オブジェクトの配列に変換します。Span オブジェクトは、フィールドや注釈で richValue または richContents を指定するのに適しています。

パラメータ

文字列 Span オブジェクトの配列に変換する XML (XFA) 文字列。

戻り値

Span オブジェクトの配列

例

「Text1」からリッチテキストを取得し、XML に変換してから再び Span オブジェクトの配列に変換して、テキストフィールドに戻します。

```
var f = getField("Text1");
var spans = f.richValue;
var str = util.spansToXML(spans);
var spans = util.xmlToSpans(str);
f.richValue = spans;
```

XFA

6.0.2			
-------	--	--	--

XFA オブジェクトを使用すると、XFA の `appModel` コンテナにアクセスできます。詳しくは、『Developing Acrobat Applications Using JavaScript』を参照してください（[28 ページの「関連ドキュメント」](#)を参照）。他の XFA マニュアルは、Acrobat Developer Center の XML のセクションから入手できます。

XFA オブジェクトは、`XMLData.parse` メソッドや `XMLData.applyXPath` メソッドによって返されます。

例

次のコードでは、PDF 文書が Acrobat 形式であるか、LiveCycle Designer で作成された XML 形式であるかを調べます。このスクリプトは、コンソールまたはバッチシーケンスから実行できます。バッチシーケンスで使用すれば、選択されているすべての文書を XML 形式（ダイナミックまたはスタティック）と Acrobat 形式に分類できます。

```
// Acrobat 8.0 を使用していることが前提。
console.println("Document name: " + this.documentFileName);
// Acrobat フォームでは xfa オブジェクトが定義されていない。
if ( typeof xfa == "object" ) {
    if ( this.dynamicXFAForm )
        console.println(" This is a dynamic XML form.");
    else
        console.println(" This is a static XML form.");
    if ( this.XFAForeground )
        console.println(" This document has imported artwork");
}
else console.println(" This is an Acrobat form.");
```

XMLData

XMLData は、静的なオブジェクトです。これを使用すれば、XML 文書ツリーを表す JavaScript オブジェクトを作成できます。また、XFA データ DOM を使用して任意の XML 文書を操作することもできます（XFA には、データ DOM のほかにも DOM がありますが、XMLData オブジェクトではデータ DOM のみが使用されます）。

Doc の `dynamicXFAForm` プロパティが `true` の PDF 文書では、XMLData オブジェクトが使用されている場合がありますが、そのオブジェクトでフォームフィールドを操作することは（2つのデータ DOM はまったく別のものなので）できません。

XMLData のメソッド

applyXPath

7.0			
-----	--	--	--

XPath 式を使用して、XML 文書の操作やクエリを行えます。XPath 言語については、W3C の文書『XML Path Language (XPath)』を参照してください。この文書は、<http://www.w3.org/TR/xpath> から入手できます。

XPath 式は、既知の 4 つの型であるブーリアン、数値、文字列、ノードセットのいずれかとして評価されます。JavaScript では、これらはそれぞれ、ブーリアン型、数値型、文字列型、オブジェクト型で結果が返されます。

オブジェクトが返される場合は、単一のノードで始まるツリーまたはノードリストで始まるツリー（ツリーリスト）を表す XFA オブジェクト型のオブジェクトが返されます。このオブジェクトの型は、`XMLData.parse` で返される型と同じです。

注意：XFA には、XPath に似た SOM 式というクエリメカニズムがあります。XML コミュニティでは XPath が広く使用されているので、`applyXPath` メソッドを使用すると、必要に応じて XPath 式を使用できます。

パラメータ

<code>oXml</code>	XML 文書ツリーを表す XFAObject オブジェクト。
注意：	このパラメータの値には XML ツリーのルートを指定します。nodeList XFA オブジェクトを指定した場合は、例外が発生します。
<code>cXPath</code>	文書ツリーに対して実行する XPATH クエリが指定された文字列パラメータ。

戻り値

ブーリアン、数値、文字列または XFAObject。

例

この例では、`XMLData.applyXPath` の戻り値のそれぞれの型（ブーリアン、数値、文字列、`XFAObject`）を示しています。XML データ文字列（この例では「Robat」家の家系図）から情報を抽出します。

```
var cXMLDoc = "<family name = 'Robat'>¥
  <grandad id = 'm1' name = 'A.C.' gender='M'>¥
    <child> m2 </child>¥
    <personal>¥
      <income>100000</income>¥
    </personal>¥
  </grandad>¥
  <dad id = 'm2' name = 'Bob' gender='M'>¥
    <parent> m1 </parent>¥
    <spouse> m3 </spouse>¥
    <child> m4 </child>¥
    <child> m5 </child>¥
    <child> m6 </child>¥
    <personal>¥
      <income>75000</income>¥
    </personal>¥
  </dad>¥
  <mom id = 'm3' name = 'Mary' gender='F'>¥
    <spouse> m2 </spouse>¥
    <personal>¥
      <income>25000</income>¥
    </personal>¥
  </mom>¥
  <daughter id = 'm4' name = 'Sue' gender='F'>¥
    <parent> m2 </parent>¥
    <personal>¥
      <income>40000</income>¥
    </personal>¥
  </daughter>¥
  <son id = 'm5' name = 'Jim' gender='M'>¥
    <parent> m2 </parent>¥
    <personal>¥
      <income>35000</income>¥
    </personal>¥
  </son>¥
  <daughter id = 'm6' name = 'Megan' gender='F'>¥
    <parent> m2 </parent>¥
    <personal>¥
      <income>30000</income>¥
    </personal>¥
  </daughter>¥
</family>";
var myXML= XMLData.parse( cXMLDoc, false);
```

次の行は、XFAObject を返します。

mom のデータを取得します。

```
var a = XMLData.applyXPath(myXML, "//family/mom")
a.saveXML('pretty');
<?xml version="1.0" encoding="UTF-8"?>
<mom id="m3" name="Mary" gender="F">
  <spouse> m2 </spouse>
  <personal>
    <income>25000</income>
  </personal>
</mom>
// income 要素の値を取得
a.personal.income.value = "20000"; // income を変更
```

dad の name (属性) を取得

```
var b = XMLData.applyXPath(myXML, "//family/dad/@name");
b.saveXML('pretty');
<?xml version="1.0" encoding="UTF-8"?>
name="Bob"
// 変数に割り当てる
var dadsName = b.value; // dadsName = "Bob"
```

dad ノードのすべての属性を取得します。

```
var b = XMLData.applyXPath( myXML, "//family/dad/attribute::*" );
for(var i=0; i < b.length; i++)
  console.println(b.item(i).saveXML('pretty'))
```

このループを実行すると、コンソールには次のように出力されます。

```
<?xml version="1.0" encoding="UTF-8"?>
id="m2"
<?xml version="1.0" encoding="UTF-8"?>
name="Bob"
<?xml version="1.0" encoding="UTF-8"?>
gender="M"
```

ここから次のように特定の情報を抽出します。

```
console.println("For attribute 2, we have " + b.item(2).name + " = '" +
  + b.item(2).value + "'.");
```

その結果、次のように出力されます。

```
For attribute 2, we have gender = 'M'.
```

dad の 2 番目の child を取得します。

```
var c = XMLData.applyXPath(myXML, "//family/dad/child[position()=2]");
c.saveXML('pretty')
<?xml version="1.0" encoding="UTF-8"?>
<child> m5 </child>
```

これが、dad の 2 番目の child の id です。次の例では、この child の家族データを取得します。

次の行は、文字列を返します。

```
// XPath メソッドを使用して dadsName の値を計算。  
var dadsName = XMLData.applyXPath(myXML, "string(//family/dad/@name)")  
// この 1 行で、dadsName に「Bob」という値が割り当てられる。
```

dad の 2 番目の child の家族情報を取得します。次の行では、c = "m5" になります。この applyXPath 呼び出しの戻り値は、文字列です。normalize-space 関数は、引数を文字列に変換して、前後の空白を削除します。

```
var c = XMLData.applyXPath(myXML,  
    "normalize-space(//family/dad/child[2])");  
var d = "//*[@id = '$' + c + '$']"; // これは、d= "//*[@id = 'm5']" と同じ  
XMLData.applyXPath(myXML, d).saveXML('pretty'); // 取得した値を示す  
<son id="m5" name="Jim" gender="M">  
  <parent> m2 </parent>  
  <personal>  
    <income>35000</income>  
  </personal>  
</son>
```

家族のルートの 6 番目の子ノードの情報を表示します。name および concat という XPath 関数を使用します。

```
var e = XMLData.applyXPath(myXML,  
    "concat(name(//family/child::*[position()=6]), '=',  
    //family/child::*[position()=6]/@name)");  
console.println(e); // 「daughter=Megan」と出力される
```

「Robat」家の全員の名前を取得します。

```
e = XMLData.applyXPath(myXML, "//family/child::*");  
for ( var i = 1; i <= e.length; i++ ) {  
  var str = "string(//family/child::*["+i+"]/@name)";  
  console.println(XMLData.applyXPath(myXML, str));  
}
```

次のように出力されます。

```
A.C.  
Bob  
Mary  
Sue  
Jim  
Megan
```

次のコードでは、ブーリアン値が返されます。

```
var f = XMLData.applyXPath( myXML, "//family/dad/@id = 'm2'" );  
if ( f == true ) console.println("dad's id is 'm2'");  
else console.println("dad's id is not 'm2'" );
```

次のコードでは、数値が返されます。

```
// dad の income を取得  
g = XMLData.applyXPath( myXML, "number(//family/dad/personal/income)" );  
// dad の給与を 2 倍にする。数値型への暗黙の変換が行われる  
console.println("Dad's double salary is " +  
    XMLData.applyXPath( myXML, "//family/dad/personal/income * 2" ) );
```

「Robat」家の総所得を計算します。

```
console.println("Total income of A.C. Robat's family is "
+ XMLData.applyXPath( myXML, "sum(/income)" ) + ".");
```

この行を実行すると、コンソールには次のように出力されます。

```
Total income of A.C. Robat's family is 305000.
```

個々の income をリストします。

```
var g = XMLData.applyXPath( myXML, "//income")
for ( var i = 0; i < g.length; i++) console.println(g.item(i).value);
```

parse

7.0			
-----	--	--	--

XML 文書ツリーを表すオブジェクトを作成します。パラメータは、XFA データ DOM の `loadXML` メソッドのパラメータと同じです。

このメソッドは、単一のノードで始まるツリーまたはノードリストで始まるツリー（ツリーリスト）を表す XFA オブジェクト型のオブジェクトを返します。

パラメータ

param1 XML 文書を表す文字列。

param2 (オプション) XML 文書のルートノードを無視するかどうかを指定するブーリアン値。デフォルト値は `true` です。`false` の場合、ルートノードは無視されません。

戻り値

XFAObject

例 1

`XMLData.applyXPath` メソッドの例で使用した XML 文書について考えます。

```
var x = XMLData.parse( cXMLDoc, false );
var y = x.family.name; // XFAObject
console.println(y.value); // コンソールに「Robat」と出力される
```

dad の情報を取得します。

```
y = x.family.dad.id; // XFAObject
console.println(y.value); // コンソールに「m2」と出力される

y = x.family.dad.name.value; // y = "Bob"
x.family.dad.name.value = "Robert"; // 名前を「Robert」に変更
y = x.family.dad.name.value; // y = "Robert"

y = x.family.dad.personal.income.value; // y = "75000"
x.family.dad.personal.income.value = "80000"; // dad を昇給させる
```

例 2

簡単な XML 文書を作成して操作します。

```
x = XMLData.parse("<a> <c>A.</c><d>C.</d> </a>", false);  
x.saveXML("pretty");
```

この行の出力は、次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>  
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">  
  <a>  
    <c>A.</c>  
    <d>C.</d>  
  </a>  
</xfa:data>
```

ここで、簡単な文書をもう 1 つ作成します。

```
y = XMLData.parse("<b>Robat</b>", false);  
y.saveXML("pretty");
```

この行の出力は、次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>  
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">  
  <b>Robat</b>  
</xfa:data>
```

x に y を追加します。

```
x.nodes.append(y.clone(true).nodes.item(0));  
x.saveXML("pretty");
```

実行結果は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>  
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">  
  <a>  
    <c>A.</c>  
    <d>C.</d>  
  </a>  
  <b>Robat</b>  
</xfa:data>
```

ここで、次のコードを実行します。

```
x.nodes.insert(y.clone(true).nodes.item(0), x.nodes.item(0));  
x.saveXML("pretty")
```

実行結果は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>  
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">  
  <b>Robat</b>  
  <a>  
    <c>A.</c>  
    <d>C.</d>  
  </a>  
  <b>Robat</b>  
</xfa:data>
```

ここで、この 2 つのノードを削除します。

```
x.nodes.remove( x.nodes.namedItem("b") );
x.nodes.remove( x.nodes.namedItem("b") );
```

この時点で、元の XML 文書に戻ります。

```
<?xml version="1.0" encoding="UTF-8"?>
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
  <a>
    <c>A.</c>
    <d>C.</d>
  </a>
</xfa:data>
```

次の行を実行します。

```
x.a.nodes.insert( y.clone(true).nodes.item(0), x.a.nodes.item(0));
x.saveXML("pretty");
```

出力は次のようにになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
  <a>
    <b>Robat</b>
    <c>A.</c>
    <d>C.</d>
  </a>
</xfa:data>
```

ここで、挿入したノードを削除します。

```
x.a.nodes.remove( x.a.nodes.namedItem("b"));
```

y (実際には y の複製) を、要素 a の最初の子と 2 番目の子の間に挿入します。

```
x.a.nodes.insert( y.clone(true).nodes.item(0), x.a.nodes.item(1));
```

次のように出力されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
  <a>
    <c>A.</c>
    <b>Robat</b>
    <d>C.</d>
  </a>
</xfa:data>
```

挿入したノードを削除します。

```
x.a.nodes.remove( x.a.nodes.namedItem("b"));
```

最後に a に y を追加します。

```
x.a.nodes.append( y.clone(true).nodes.item(0));
```

次のように出力されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xfa:data xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
  <a>
    <c>A.</c>
    <d>C.</d>
    <b>Robat</b>
  </a>
</xfa:data>
```

ここでは、Acrobat 8.0 およびそれ以前のバージョンで導入された新しい機能および変更について説明します。

Acrobat 8.0 での変更

JavaScript インタプリタで、E4X (ECMA-357) がサポートされるようになりました。仕様については、『ECMAScript for XML (E4X) Specification』(<http://www.ecma-international.org/publications/standards/Ecma-357.htm>) を参照してください。E4X の使用例については、[245 ページの metadata](#) の例を参照してください。

環境設定ダイアログボックス (Ctrl+K キー) の「JavaScript」分類に、「グローバルオブジェクトセキュリティポリシーを有効にする」というセキュリティ関連の新しいチェックボックスが追加されました。

- チェックを付けると（デフォルトでチェックが付いています）、グローバル変数を設定するたびに設定元が記憶され、その設定元のみがその変数にアクセスできるようになります。例えば、ファイルで変数を設定した場合は、そのファイル（変数を設定したときと同じパスにあるファイル）のみがその変数にアクセスできるようになります。また、特定の URL にある文書で変数を設定した場合は、その Web ホストのみがその変数にアクセスできるようになります。

これらの制限には、重要な例外があります。セキュリティによる制限がないコンテキスト（コンソール、バッチシーケンス、フォルダレベルの JavaScript）では、グローバル変数の定義やアクセスを行うことができます。セキュリティによる制限がないコンテキストについて詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

- チェックを解除すると、設定元でない文書もその変数にアクセスできるようになります（これは、8.0 より前のバージョンの Acrobat と同じ動作です）。

詳しい説明と例については、[global](#) オブジェクトの節 ([479 ページ](#)) を参照してください。

`app.execMenuItem` で実行できるメニュー項目が、セーフメニュー項目のリストに登録されている項目に制限されるようになりました。[120 ページ](#) のセキュリティに関する注意事項を参照してください。

Acrobat 8.0 では、次のプロパティとメソッドが導入されました。

<u>Certificate</u> <u>オブジェクト</u>	プロパティ： <code>privateKeyValidityEnd</code> <code>privateKeyValidityStart</code> <code>validityEnd</code> (バージョン 7.0.5 で導入されましたが、未記載でした) <code>validityStart</code> (バージョン 7.0.5 で導入されましたが、未記載でした)
<u>CertificateSpecifier</u> <u>オブジェクト</u>	プロパティ： <code>subjectDN</code> <code>keyUsage</code> <code>urlType</code> url プロパティの説明に加筆。 <code>flags</code> プロパティに、4 つの新しいビットフラグを追加 (<code>subjectDN</code> 、 <code>issuerDN</code> 、 <code>keyUsage</code> 、 <code>url</code>)。

colorConvertAction	オブジェクト プロパティ： action alias colorantName convertIntent convertProfile embed isProcessColor matchAttributesAll matchAttributesAny matchIntent matchSpaceTypeAll matchSpaceTypeAny preserveBlack useBlackPointCompensation
Doc	オブジェクト プロパティ： xfa (バージョン 6.0.2 で導入されましたが、未記載でした) XFAForeground メソッド： colorConvertPage embedOutputIntent exportAsFDFStr exportAsXFDFStr getColorConvertAction
Net	オブジェクト プロパティ： SOAP (Net.SOAP のプロパティやメソッドは、SOAP.wireDump、SOAP.connect、SOAP.request、SOAP.response のエイリアスです) Discovery (Net.Discovery のメソッドは、SOAP.queryServices と SOAP.resolveService のエイリアスです) HTTP メソッド： 544 ページ を参照 steamDecode streamDigest streamEncode
Net.HTTP	オブジェクト メソッド： request
PrintParams	オブジェクト プロパティ： booklet constants.bookletBindings constants.bookletDuplexMode constants.handling.booklet (詳しくは、 pageHandling を参照)

<u>RDN</u> オブジェクト	プロパティ：RDN オブジェクトに次の 22 のプロパティを追加。 <code>businessCategory</code> 、 <code>countryOfCitizenship</code> 、 <code>countryOfResidence</code> 、 <code>dateOfBirth</code> 、 <code>dc</code> 、 <code>dnQualifier</code> 、 <code>gender</code> 、 <code>generationQualifier</code> 、 <code>givenName</code> 、 <code>initials</code> 、 <code>l</code> 、 <code>name</code> 、 <code>nameAtBirth</code> 、 <code>placeOfBirth</code> 、 <code>postalAddress</code> 、 <code>postalCode</code> 、 <code>pseudonym</code> 、 <code>serialNumber</code> 、 <code>sn</code> 、 <code>st</code> 、 <code>street</code> 、 <code>title</code>
<u>SecurityHandler</u> オブジェクト	プロパティ：次の 3 つのプロパティは未掲載でした。 <u>docDecrypt</u> <u>docEncrypt</u> <u>validatePDF</u>
<u>SeedValue</u> オブジェクト	3 つの新しいフラグを追加 (<code>legalAttestations</code> 、 <code>shouldAddRevInfo</code> 、 <code>digestMethod</code>)。 <code>reasons</code> プロパティの動作を変更。 <code>digestMethod</code> プロパティを追加。 <code>version</code> プロパティの説明の表現を変更。 <code>shouldAddRevInfo</code> プロパティを追加。
<u>SignatureInfo</u> オブジェクト	プロパティ： <code>digestMethod</code>
Acrobat 8.0 では、次のプロパティとメソッドが変更されました。	
<u>app</u> オブジェクト メソッド：	<u>openDoc</u> (cDest パラメータを追加)
<u>Doc</u> オブジェクト メソッド：	<u>addAnnot</u> (author プロパティに新しいデフォルト値を追加) <u>addRecipientListCryptFilter</u> (Acrobat 7.0 の場合の注意事項を追加) <u>getLegalWarnings</u> (文書にその情報が追加されなくなった。戻り値を変更) <u>importTextData</u> (コンソールイベントまたはバッチイベントでのみ実行可能)
FDF オブジェクト メソッド：	<u>addEmbeddedFile</u> (バージョン 8 の場合の nSaveOption パラメータの説明を更新)
<u>Field</u> オブジェクト メソッド：	<u>buttonImportIcon</u> (コンソールイベントまたはバッチイベントでのみ実行可能)
<u>FullScreen</u> オブジェクト	プロパティ： <u>escapeExits</u> (false の設定は、コンソールイベントとバッチイベントでのみ可能)
<u>LoginParameters</u> オブジェクト	cURI および cUserId プロパティの表現を変更
<u>submitForm</u> メソッド	cPassword パラメータは使用できなくなりました。Acrobat 8.0 以降では、パスワードで暗号化した FDF ファイルは作成できません。このパラメータを使用した場合、パスワードで暗号化した FDF をフォームから送信しようとすると、ESErrorInvalidArgs という例外が発生し、フォームの送信は行われません。

Acrobat 7.0.5 での変更

クイックバーの 5 列目と 6 列目が削除されました。

Acrobat 7.0.5 では、次のプロパティとメソッドが導入されました。

<u>Collab</u> オブジェクト	メソッド： documentToStream
<u>Data</u> オブジェクト	プロパティ： description
<u>Doc</u> オブジェクト	プロパティ： hostContainer isModal viewState メソッド： addRequirement removeRequirement
<u>Embedded PDF</u> オブジェクト	プロパティ： messageHandler メソッド： postMessage
<u>HostContainer</u> オブジェクト	プロパティ： messageHandler メソッド： postMessage
<u>util</u> オブジェクト	メソッド： crackURL
Acrobat 7.0.5 では、次のプロパティとメソッドが変更されました。	
<u>SOAP</u> オブジェクト	メソッド： request (cRequestStyle および cContent パラメータを追加) response (cRequestStyle および cContent パラメータを追加)

Acrobat 7.0 での変更

バージョン 6.0.2 では別のマニュアルになっていた『Acrobat Multimedia JavaScript Reference』が『Acrobat JavaScript Scripting Reference』に含まれるようになり、『JavaScript for Acrobat API Reference』という名前に変更されました。マルチメディア JavaScript のオブジェクト、プロパティ、メソッドのリストは、[751 ページの「Acrobat 6.0.2 での新しい機能」](#) を参照してください。

メニューイベントでの JavaScript の実行にセキュリティによる制限が加えられるようになりました。セキュリティによる制限があるコンテキストで、その制限を解除してコードを実行できるようになりました。詳しくは、[32 ページの「セキュリティによる制限がないコンテキストとセキュリティによる制限があるコンテキスト」](#) を参照してください。

7.0 より前のバージョンの Acrobat では、アプリケーションの JavaScript フォルダの中に、AForm.js、ADBC.js、Annots.js、AnWizard.js、media.js、SOAP.js という JavaScript ファイルがありました。Acrobat 7.0 以降では、これらのファイルは Acrobat Professional、Acrobat Standard、Adobe Reader とともに提供されなくなりました。その代わりに、パフォーマンスを向上させるために、プリコンパイルしたバイトコードの形式で提供されています。アプリケーションフォルダの debugger.js ファイルは、バイトコードでは提供されていません。

ユーザの JavaScript フォルダにあるファイルは、プリコンパイルされたバイトコードファイルになることはありません。

ユーザが作成した .js ファイルは、glob.js とともに、ユーザの JavaScript フォルダに置くことをお勧めします。メニュー項目をセットアップする JavaScript コード ([addMenuItem](#)) は、ユーザの JavaScript フォルダにある config.js に記述してください。このフォルダの場所をプログラムで見つけるには、コンソールから app.getPath("user", "javascript") を実行してください。

Adobe Reader でコンソールウィンドウを表示できるようになりました。編集／環境設定／JavaScript で、「エラーとメッセージをコンソールに表示」を選択します。エラーや例外が発生した場合に加えて、console.show() を使用してプログラムでもコンソールを開けます。詳しくは、[console](#) オブジェクトを参照してください。

JavaScript デバッグウィンドウのデバッグ機能は、Windows 版と Macintosh 版の Adobe Reader で使用できます。Adobe Reader でデバッグを行うには、debugger.js という JavaScript ファイルをインストールして、Windows のレジストリを適切に編集する必要があります。技術的な内容について詳しくは、『Developing Acrobat Applications Using JavaScript』を参照してください。

Acrobat 7.0 での新しい機能

Acrobat 7.0 では、次のプロパティとメソッドが導入されました。

[Alerter](#) オブジェクト メソッド：
[dispatch](#)

[Annotation](#) オブジェクト プロパティ：
[callout](#)
[caretSymbol^a](#)
[creationDate^a](#)
[dash^b](#)
[delay^b](#)
[doCaption](#)
[intent](#)
[leaderExtend](#)
[leaderLength](#)
[lineEnding](#)
[opacity^b](#)
[refType](#)
[richDefaults^a](#)
[seqNum^b](#)
[state^a](#)
[stateModel^a](#)
[style](#)
[subject^a](#)

[Annot3D](#) オブジェクト プロパティ：
[activated](#)
[context3D](#)
[innerRect](#)
[name](#)
[page](#)
[rect](#)

[app](#) オブジェクト プロパティ：
[constants](#)
メソッド：
[beginPriv](#)
[browseForDoc](#)
[endPriv](#)
[execDialog](#)
[launchURL](#)
[trustedFunction](#)
[trustPropagatorFunction](#)

Dialog オブジェクト	メソッド： enable end load store
Doc オブジェクト	プロパティ： docID ^a dynamicXFAForm external hidden mouseX mouseY noautocomplete nocache requiresFullSave
	メソッド： addWatermarkFromFile addWatermarkFromText embedDocAsDataObject encryptUsingPolicy getAnnot3D getAnnots3D getDataObjectContents getOCGOrder openDataObject removeScript setDataObjectContents setOCGOrder
Doc.media オブジェクト	メソッド： getOpenPlayers
Field オブジェクト	メソッド： signatureGetModifications
OCG オブジェクト	プロパティ： constants initState locked メソッド： getIntent setIntent
PlayerInfo オブジェクト	メソッド： honors

PrintParams オブジェクト	プロパティ： nUpAutoRotate nUpNumPagesH nUpNumPagesV nUpPageBorder nUpPageOrder
search オブジェクト	プロパティ： attachments ignoreAccents objectMetadata proximityRange
security オブジェクト	security の定数 メソッド： chooseSecurityPolicy getSecurityPolicies
SecurityPolicy オブジェクト	SecurityPolicy のプロパティ
SOAP オブジェクト	メソッド： queryServices resolveService streamDigest
util オブジェクト	メソッド： iconStreamFromIcon streamFromString stringFromStream
XMLData オブジェクト	メソッド： applyXPath parse
a. バージョン 6.0 から存在していましたが、バージョン 7.0 で掲載されました。	
b. バージョン 5.0 から存在していましたが、バージョン 7.0 で掲載されました。	

Acrobat 7.0 での変更

変更または拡張されたオブジェクト、メソッド、プロパティ

次のプロパティとメソッドが変更または拡張されました。

app オブジェクト	メソッド： addToolBar execMenuItem getPath mailGetAddrs openDoc
----------------------------	---

console オブジェクト	Adobe Reader でコンソールウィンドウを表示できるようになりました。
Doc オブジェクト	メソッド： createTemplate mailDoc print saveAs submitForm
Field オブジェクト	メソッド： signatureSetSeedValue
Index オブジェクト	メソッド： build
OCG オブジェクト	プロパティ： name
PrintParams オブジェクト	プロパティ： pageHandling
search オブジェクト	プロパティ： indexes
security オブジェクト	プロパティ： handlers メソッド： getHandler
SecurityHandler オブジェクト	プロパティ： digitalIDs メソッド： login newUser
SOAP オブジェクト	メソッド： connect request
spell オブジェクト	7.0 より前のバージョンの Adobe Reader では、 spell オブジェクトは使用できません。 メソッド： addWord
util オブジェクト	メソッド： printd

Acrobat 6.0 での変更

このバージョンの Acrobat には、セーフパスという概念が導入されました。詳しくは、[31 ページの「セーフパス」](#)を参照してください。

Acrobat 6.0 での新しい機能

Acrobat 6 では、次のプロパティとメソッドが導入されました。

ADBC オブジェクト	SQL 型
AlternatePresentation オブジェクト	プロパティ： active type メソッド： start stop
Annotation オブジェクト	プロパティ： borderEffectIntensity borderEffectStyle inReplyTo richContents toggleNoView メソッド： getStateInModel transitionToState
app オブジェクト	プロパティ： fromPDFConverters printColorProfiles printerNames runtimeHighlight runtimeHighlightColor thermometer viewerType メソッド： addToolBar getPath mailGetAddrs newFDF openFDF popUpMenuEx removeToolBar

Bookmark オブジェクト	メソッド： setAction
catalog オブジェクト	プロパティ： isIdle jobs メソッド： getIndex remove
Certificate オブジェクト	プロパティ： keyUsage usage
Collab オブジェクト	メソッド： addStateModel removeStateModel
Connection オブジェクト	メソッド： close
dbg オブジェクト	プロパティ： bps メソッド： c cb q sb si sn so sv
Directory オブジェクト	プロパティ： info メソッド： connect

[DirConnection](#) オブジェクト プロパティ：

[canList](#)
[canDoCustomSearch](#)
[canDoCustomUISearch](#)
[canDoStandardSearch](#)
[groups](#)
[name](#)
[uiName](#)

メソッド：

[search](#)
[setOutputFields](#)

[Doc](#) オブジェクト

プロパティ：

[alternatePresentations](#)
[documentFileName](#)
[metadata](#)
[permStatusReady](#)

メソッド：

[addLink](#)
[addRecipientListCryptFilter](#)
[addRequirement](#)
[encryptForRecipients](#)
[exportAsText](#)
[exportXFAData](#)
[getLegalWarnings](#)
[getLinks](#)
[getOCGs](#)
[getPrintParams](#)
[importXFAData](#)
[newPage](#)
[removeLinks](#)
[setAction](#)
[setPageAction](#)
[setPageTabOrder](#)

[Error](#) オブジェクト

プロパティ：

[fileName](#)
[lineNumber](#)
[message](#)
[name](#)

メソッド：

[toString](#)

event オブジェクト	プロパティ： fieldFull richChange richChangeEx richValue
FDF オブジェクト	プロパティ： deleteOption isSigned numEmbeddedFiles メソッド： addContact addEmbeddedFile addRequest close mail save signatureClear signatureSign signatureValidate
Field オブジェクト	プロパティ： buttonFitBounds comb commitOnSelChange defaultStyle radiosInUnison richText richValue rotation メソッド： getLock setLock signatureGetSeedValue signatureSetSeedValue
Index オブジェクト	メソッド： build

Link オブジェクト	プロパティ： borderColor borderWidth highlightMode rect メソッド： setAction
OCG オブジェクト	プロパティ： name state メソッド： setAction
PrintParams オブジェクト	プロパティ： binaryOK bitmapDPI colorOverride colorProfile constants downloadFarEastFonts fileName firstPage flags fontPolicy gradientDPI interactive lastPage pageHandling pageSubset printAsImage printContent printerName psLevel rasterFlags reversePages tileLabel tileMark tileOverlap tileScale transparencyLevel usePrinterCRD useT1Conversion

Report オブジェクト	プロパティ： style
search オブジェクト	プロパティ： docInfo docText docXMP bookmarks ignoreAsianCharacterWidth jpegExif legacySearch markup matchWholeWord wordMatching
security オブジェクト	メソッド： chooseRecipientsDialog getSecurityPolicies importFromFile
SecurityHandler オブジェクト	プロパティ： digitalIDs directories directoryHandlers signAuthor signFDF メソッド： newDirectory
SOAP オブジェクト	プロパティ： wireDump メソッド： connect request response streamDecode streamEncode streamFromString stringFromStream

Span オブジェクト	プロパティ： alignment fontFamily fontStretch fontStyle fontWeight text textColor textSize strikethrough subscript superscript underline
spell オブジェクト	プロパティ： languages languageOrder メソッド： customDictionaryClose customDictionaryCreate customDictionaryExport customDictionaryOpen ignoreAll
Thermometer オブジェクト	プロパティ： cancelled duration value text メソッド： begin end
util オブジェクト	メソッド： printd spansToXML xmlToSpans

Acrobat 6.0 での変更

変更または拡張されたオブジェクト、メソッド、プロパティ

次のプロパティとメソッドが変更または拡張されました。

app オブジェクト

メソッド：

[addMenuItem](#)
[alert](#)
[listMenuItems](#)
[listToolbarButtons](#)
[response](#)

Doc オブジェクト

プロパティ：

[layout](#)
[zoomType](#)

メソッド：

[createDataObject](#)
[exportAsFDF](#)
[exportAsXFDF](#)
[exportDataObject](#)
[flattenPages](#)
[getField](#) (拡張されたメソッドを参照)
[getURL](#)
[importDataObject](#)
[importIcon](#)
[print](#)
[saveAs](#)
[spawnPageFromTemplate](#)
[submitForm](#)

event オブジェクト

プロパティ：

[changeEx](#)

Field オブジェクト

プロパティ：

[name](#)

メソッド：

[buttonImportIcon](#)
[signatureInfo](#)
[signatureSign](#)
[signatureValidate](#)

global オブジェクト

パーシスタントグローバルデータは、ブーリアン型、数値型、文字列型の変数にのみ適用できます。Acrobat 6.0 では、パーシスタントグローバル変数の最大サイズが 32 KB から 2 ~ 4 KB に縮小されました。この制限を超えたデータは、すべて切り捨てられます。

search オブジェクト	メソッド： query
SecurityHandler オブジェクト	Acrobat 5.0 では、PPKLite Signature Handler オブジェクトに次のプロパティやメソッドが導入されました。Acrobat 6.0 では、これらは SecurityHandler オブジェクトのプロパティやメソッドになりました。これらに関する説明が追加されており、新しいパラメータが追加されたものもあります。 注意： JavaScript のメソッドを使用して署名する場合、ユーザの電子署名プロファイルは、以前のバージョンの Acrobat で使用していた .apf ではなく、.pxf ファイルである必要があります。.apf プロファイルを新しい .pxf タイプに変換するには、UI (アドバンスト／デジタル ID の管理／デジタル ID ファイル／デジタル ID ファイルを選択) を使用して、.apf プロファイルを取り込みます。
	プロパティ： appearances isLoggedIn loginName loginPath name signInvisible signVisible uiName

Template オブジェクト	メソッド： spawn
---------------------------------	--------------------------------

拡張されたメソッド

Acrobat 6.0 では、Document.[getField](#) メソッドが、個々のウィジェットの Field オブジェクトを取得するように拡張されました。ウィジェットとその操作方法について詳しくは、[Field](#) オブジェクトを参照してください。

Acrobat 6.0 での非推奨

search オブジェクト	プロパティ： soundex thesaurus
spell オブジェクト	メソッド： addDictionary removeDictionary

Acrobat 6.0.2 での新しい機能

Acrobat 6.0.2 では、次のオブジェクト、プロパティ、メソッドが導入されました。

XFA オブジェクト

次の表に、Multimedia プラグインのオブジェクト、プロパティ、メソッドを示します。Acrobat 6.0.2 では、マルチメディア JavaScript について説明した『Acrobat Multimedia JavaScript Reference』というマニュアルがありました。

<u>app</u> オブジェクト	プロパティ： <u>media</u> <u>monitors</u>
<u>app.media</u> オブジェクト	プロパティ： <u>align</u> <u>canResize</u> <u>closeReason</u> <u>defaultVisible</u> <u>ifOffScreen</u> <u>layout</u> <u>monitorType</u> <u>openCode</u> <u>over</u> <u>pageEventNames</u> <u>raiseCode</u> <u>raiseSystem</u> <u>renditionType</u> <u>status</u> <u>trace</u> <u>version</u> <u>windowType</u>

[app.media](#) オブジェクト
(続き)

メソッド：

[addStockEvents](#)
[alertFileNotFound](#)
[alertSelectFailed](#)
[argsDWIM](#)
[canPlayOrAlert](#)
[computeFloatWinRect](#)
[constrainRectToScreen](#)
[createPlayer](#)
[getAltTextData](#)
[getAltTextSettings](#)
[getAnnotStockEvents](#)
[getAnnotTraceEvents](#)
[getPlayers](#)
[getPlayerStockEvents](#)
[getPlayerTraceEvents](#)
[getRenditionSettings](#)
[getURLData](#)
[getURLSettings](#)
[getWindowBorderSize](#)
[openPlayer](#)
[removeStockEvents](#)
[startPlayer](#)

[Doc](#) オブジェクト

プロパティ：

[innerAppWindowRect](#)
[innerDocWindowRect](#)
[media](#)
[outerAppWindowRect](#)
[outerDocWindowRect](#)
[pageWindowRect](#)

[Doc.media](#) オブジェクト

プロパティ：

[canPlay](#)

メソッド：

[deleteRendition](#)
[getAnnot](#)
[getAnnots](#)
[getRendition](#)
[newPlayer](#)

[event](#) オブジェクト

Multimedia で使用する新しい画面タイプ。関連付けられたイベント名とともに使用します。

[Events](#) オブジェクト メソッド：

[add](#)
[dispatch](#)
[remove](#)

[EventListener](#) オブジェクト メソッド：

[afterBlur](#)
[afterClose](#)
[afterDestroy](#)
[afterDone](#)
[afterError](#)
[afterEscape](#)
[afterEveryEvent](#)
[afterFocus](#)
[afterPause](#)
[afterPlay](#)
[afterReady](#)
[afterScript](#)
[afterSeek](#)
[afterStatus](#)
[afterStop](#)
[onBlur](#)
[onClose](#)
[onDestroy](#)
[onDone](#)
[onError](#)
[onEscape](#)
[onEveryEvent](#)
[onFocus](#)
[onGetRect](#)
[onPause](#)
[onPlay](#)
[onReady](#)
[onScript](#)
[onSeek](#)
[onStatus](#)
[onStop](#)

[Marker](#) オブジェクト プロパティ：

[frame](#)
[index](#)
[name](#)
[time](#)

Markers オブジェクト	プロパティ： player メソッド： get
MediaOffset オブジェクト	プロパティ： frame marker time
MediaPlayer オブジェクト	プロパティ： annot defaultSize doc events hasFocus id innerRect isOpen isPlaying markers outerRect page settings uiSize visible メソッド： close open pause play seek setFocus stop triggerGetRect where
MediaReject オブジェクト	プロパティ： rendition
MediaSelection オブジェクト	プロパティ： selectContext players rejects rendition

[MediaSettings](#) オブジェクト プロパティ :

[autoPlay](#)
[baseURL](#)
[bgColor](#)
[bgOpacity](#)
[endAt](#)
[floating](#)
[duration](#)
[floating](#)
[layout](#)
[monitor](#)
[monitorType](#)
[page](#)
[palindrome](#)
[players](#)
[rate](#)
[repeat](#)
[showUI](#)
[startAt](#)
[visible](#)
[volume](#)
[windowType](#)

[Monitor](#) オブジェクト プロパティ :

[colorDepth](#)
[isPrimary](#)
[rect](#)
[workRect](#)

[Monitors](#) オブジェクト メソッド :

[bestColor](#)
[bestFit](#)
[desktop](#)
[document](#)
[filter](#)
[largest](#)
[leastOverlap](#)
[mostOverlap](#)
[nonDocument](#)
[primary](#)
[secondary](#)
[select](#)
[tallest](#)
[widest](#)

[PlayerInfo](#) オブジェクト プロパティ：
[id](#)
[mimeTypes](#)
[name](#)
[version](#)
メソッド：
[canPlay](#)
[canUseData](#)

[PlayerInfoList](#) オブジェクト メソッド：
[select](#)

[Rendition](#) オブジェクト プロパティ：
[altText](#)
[doc](#)
[fileName](#)
[type](#)
[uiName](#)
メソッド：
[getPlaySettings](#)
[select](#)
[testCriteria](#)

[ScreenAnnot](#) オブジェクト プロパティ：
[altText](#)
[alwaysShowFocus](#)
[display](#)
[doc](#)
[events](#)
[extFocusRect](#)
[innerDeviceRect](#)
[noTrigger](#)
[outerDeviceRect](#)
[page](#)
[player](#)
[rect](#)
メソッド：
[hasFocus](#)
[setFocus](#)

Acrobat 5.0 での変更

Acrobat 5.0 での新しい機能

ADBC オブジェクト	メソッド： getDataSourceList newConnection
Annotation オブジェクト	プロパティ： alignment AP arrowBegin arrowEnd author contents doc fillColor hidden modDate name noView page point points popupRect print rect readOnly rotate strokeColor textFont type soundIcon width

[app](#) オブジェクト

プロパティ：
[activeDocs](#)
[fs](#)
[plugIns](#)
[viewerVariation](#)

メソッド：

[addMenuItem](#)
[addSubMenu](#)
[clearInterval](#)
[clearTimeOut](#)
[listMenuItems](#)
[listToolbarButtons](#)
[newDoc](#)
[openDoc](#)
[popUpMenu](#)
[setInterval](#)
 [setTimeOut](#)

[Bookmark](#) オブジェクト

プロパティ：
[children](#)
[color](#)
[doc](#)
[name](#)
[open](#)
[parent](#)
[style](#)

メソッド：

[createChild](#)
[execute](#)
[insertChild](#)
[remove](#)

[color](#) オブジェクト

メソッド：
[convert](#)
[equal](#)

[Connection](#) オブジェクト

メソッド：
[getColumnList](#)
[getTableList](#)
[console](#)

[Data](#) オブジェクト

プロパティ：

[creationDate](#)
[modDate](#)
[MIMETYPE](#)
[name](#)
[path](#)
[size](#)

[Doc](#) オブジェクト

プロパティ：

[bookmarkRoot](#)
[disclosed](#) (5.0.5)
[icons](#)
[info](#)
[layout](#)
[securityHandler](#)
[selectedAnnots](#)
[sounds](#)
[templates](#)
[URL](#)

メソッド：

[addAnnot](#)
[addField](#)
[addIcon](#)
[addThumbnails](#)
[addWeblinks](#)
[bringToFront](#)
[closeDoc](#)
[createDataObject](#)
[createTemplate](#)
[deletePages](#)
[deleteSound](#)
[exportAsXFDF](#)
[exportDataObject](#)
[extractPages](#)
[flattenPages](#)
[getAnnot](#)
[getAnnots](#)
[getDataObject](#)
[getIcon](#)
[getPageBox](#)
[getPageLabel](#)
[getPageNthWord](#)
[getPageNthWordQuads](#)
[getPageRotation](#)
[getPageTransition](#)

[Doc](#) オブジェクト
(続き)

[getSound](#)
[importAnXFDF](#)
[importDataObject](#)
[importIcon](#)
[importSound](#)
[importTextData](#)
[insertPages](#)
[movePage](#)
[print](#)
[removeDataObject](#)
[removeField](#)
[removeIcon](#)
[removeTemplate](#)
[removeThumbnails](#)
[removeWeblinks](#)
[replacePages](#)
[saveAs](#)
[selectPageNthWord](#)
[setPageBoxes](#)
[setPageLabels](#)
[setPageRotations](#)
[setPageTransitions](#)
[submitForm](#)
[syncAnnotScan](#)

[event](#) オブジェクト

プロパティ：
[changeEx](#)
[keyDown](#)
[targetName](#)

[Field](#) オブジェクト

プロパティ：
[buttonAlignX](#)
[buttonAlignY](#)
[buttonPosition](#)
[buttonScaleHow](#)
[buttonScaleWhen](#)
[currentValueIndices](#)
[doNotScroll](#)
[doNotSpellCheck](#)
[exportValues](#)
[fileSelect](#)
[multipleSelection](#)
[rect](#)
[strokeColor](#)
[submitName](#)
[valueAsString](#)

Field オブジェクト
(続き)

メソッド：

[browseForFileToSubmit](#)
[buttonGetCaption](#)
[buttonGetIcon](#)
[buttonSetCaption](#)
[buttonSetIcon](#)
[checkThisBox](#)
[defaultIsChecked](#)
[isBoxChecked](#)
[isDefaultChecked](#)
[setAction](#)
[signatureInfo](#)
[signatureSign](#)
[signatureValidate](#)

FullScreen オブジェクト

プロパティ：

[backgroundColor](#)
[clickAdvances](#)
[cursor](#)
[defaultTransition](#)
[escapeExits](#)
[isFullScreen](#)
[loop](#)
[timeDelay](#)
[transitions](#)
[usePageTiming](#)
[useTimer](#)

global オブジェクト

メソッド：

[subscribe](#)

identity オブジェクト

プロパティ：

[corporation](#)
[email](#)
[loginName](#)
[name](#)

Index オブジェクト

プロパティ：

[available](#)
[name](#)
[path](#)
[selected](#)

PlayerInfo オブジェクト	プロパティ： certified loaded name path version
PPKLite Signature Handler オブジェクト（現在は SecurityHandler オブジエ クトにあります）	プロパティ： appearances isLoggedIn loginName loginPath name signInvisible signVisible uiName メソッド： login logout newUser setPasswordTimeout
Report オブジェクト	プロパティ： absIndent color absIndent メソッド： breakPage divide indent outdent open mail writeText save writeText
search オブジェクト	プロパティ： available indexes markup maxDocs proximity refine soundex stem

<u>search</u> オブジェクト (続き)	メソッド： addIndex getIndexForPath query removeIndex
<u>security</u> オブジェクト	プロパティ： handlers validateSignaturesOnOpen
	メソッド： getHandler
<u>spell</u> オブジェクト	プロパティ： available dictionaryNames dictionaryOrder domainNames
	メソッド： addDictionary addWord check checkText checkWord removeDictionary removeWord userWords
<u>Statement</u> オブジェクト	プロパティ： columnCount rowCount
	メソッド： execute getColumn getColumnArray getRow nextRow
<u>Template</u> オブジェクト	プロパティ： hidden name
	メソッド： spawn

Acrobat 5.0 での変更

コンソールがエディタの役割も果たすようになりました。JavaScript コードを実行できます。

次のプロパティとメソッドが変更または拡張されました。

app オブジェクト	language execMenuItem
Doc オブジェクト	exportAsPDF print submitForm
event オブジェクト	type
Field オブジェクト	textFont value buttonImportIcon getItemAt
util オブジェクト	printd

[event](#) オブジェクトに関する節が大幅に変更され、Acrobat JavaScript のイベントモデルがわかりやすくなりました。

Acrobat 5.0 での非推奨

次のプロパティとメソッドが非推奨になりました。

app オブジェクト	fullscreen numPlugIns getNthPlugInName
Doc オブジェクト	author creationDate creationDate keywords modDate numTemplates producer title getNthTemplate spawnPageFromTemplate
Field オブジェクト	hidden
TTS オブジェクト	soundCues speechCues

Acrobat 5.05 での変更

- Acrobat Approval で使用できないメソッドを示す新しいマークが、クイックバーに追加されました。
- [Doc](#) オブジェクトに、[disclosed](#) プロパティが追加されました。

Adobe Reader 5.1 での変更

Acrobat Reader 5.1 では、次のプロパティとメソッドへのアクセスが変更されました。

Annotation オブジェクト	プロパティ： alignment AP arrowBegin arrowEnd author contents doc fillColor hidden	modDate name noView page point points popupRect print	rect readOnly rotate strokeColor textFont type soundIcon width
	メソッド： destroy getProps setProps		
Doc オブジェクト	プロパティ： selectedAnnots		
	メソッド： addAnnot addField exportAsPDF exportAsXFDF getAnnot getAnnots getNthTemplate importAnFDF	importAnXFDF importDataObject mailDoc mailForm spawnPageFromTemplate submitForm syncAnnotScan	
Template オブジェクト	メソッド： spawn		