pdf_output

Apr 05, 2024 22:33

Supported Clients

SmartClient WebClient NGClient

Methods Summary

Array	addMetaData(data, metaData)	Add metadata to the PDF, like Author
Array	combinePDFDocuments (pdf_docs_bytearrays)	Combine multiple PDF docs into one.
Array	combineProtectedPDFDocuments (pdf_docs_bytearrays, pdf_docs_passwords)	Combine multiple protected PDF docs into one.
Array	convertPDFFormToPDFDocument(pdf_form, field_values)	Convert a PDF form to a PDF document.
Array	convertProtectedPDFFormToPDFDocument (pdf_form, pdf_password, field_values)	Convert a protected PDF form to a PDF document.
Array	encrypt(data, ownerPassword)	Add password protection and security options to the PDF
Array	encrypt(data, ownerPassword, userPassword)	Add password protection and security options to the PDF
Array	encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders)	Add password protection and security options to the PDF
Array	encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders, is128bit)	Add password protection and security options to the PDF
Array	encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders, is128bit, metaData)	Add password protection and security options to the PDF
Array	endMetaPrintJob()	Ends a previously started meta print job.
Object	getPDFPrinter()	Returns a PDF printer that can be used in print calls.
Object	getPDFPrinter(filename)	Returns a PDF printer that can be used in print calls.
Number	getPagesPrinted()	Returns the number of pages printed by the last print call done in the context of a meta print job.
Array	getThumbnailImage(data)	Create a thumbnail from the provided PDF
Array	getThumbnailImage(data, pageNumber)	Create a thumbnail from the provided PDF
Array	getThumbnailImage(data, pageNumber, dpi)	Create a thumbnail from the provided PDF
Number	getTotalPagesPrinted()	Returns the total number of pages printed in the context of a meta print job.
Number	insertFontDirectory(path)	Add a directory that should be searched for fonts.
Array	numberPages(data)	Add pages numbers to a PDF
Array	numberPages(data, fontSize, locationX, locationY, font, hexColor)	Add pages numbers to a PDF
Array	overlay(data, forOverlay)	Add some PDF based content over a PDF
Array	overlay(data, forOverlay, isOver)	Add some PDF based content over a PDF
Array	overlay(data, forOverlay, isOver, pages)	Add some PDF based content over a PDF
Array	overlay(data, forOverlay, pages)	Add some PDF based content over a PDF
Array	overlayText(data, text)	Add text over every page at a 45 degree angle
Array	overlayText(data, text, locationX, locationY, isOver, fontSize, font, hexColor)	Add text over every page at a 45 degree angle
Boolean	startMetaPrintJob()	Used for printing multiple things into the same PDF document.
Boolean	startMetaPrintJob(filename)	Used for printing multiple things into the same PDF document.
Array	watermark(data, image)	Add an image as a watermark on every page, or the pages specified as a parameter
Array	watermark(data, image, locationX, locationY, isOver)	Add an image as a watermark on every page, or the pages specified as a parameter
Array	watermark(data, image, locationX, locationY, isOver, pages)	Add an image as a watermark on every page, or the pages specified as a parameter

Methods Details

addMetaData(data, metaData)

```
Add metadata to the PDF, like Author
```

Parameters

Array data the PDF
Object metaData a JavaScript object (Scriptable) that contains the metadata as property/value pairs

Returns

Array

Supported Clients

SmartClient,WebClient

Sample

```
// Add metadata to the PDF, like Author
var pdf = plugins.file.showFileOpenDialog();
if (pdf) {
        var data = plugins.file.readFile(pdf);
        var metaData = { Author: 'Servoy' };
        pdfResult = %%elementName%%.addMetaData(data, metaData);
}
```

combinePDFDocuments(pdf_docs_bytearrays)

Combine multiple PDF docs into one.
Note: this function may fail when creating large PDF files due to lack of available heap memory. To compensate, please configure the application server with more heap memory via -Xmx parameter.

Parameters

Array pdf_docs_bytearrays the array of documents to combine

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
pdf_blob_column = combinePDFDocuments(new Array(pdf_blob1,pdf_blob2,pdf_blob3));
```

combineProtectedPDFDocuments(pdf_docs_bytearrays, pdf_docs_passwords)

Combine multiple protected PDF docs into one.
Note: this function may fail when creating large PDF files due to lack of available heap memory. To compensate, please configure the application server with more heap memory via -Xmx parameter.

Parameters

Array pdf_docs_bytearrays the array of documents to combine
Array pdf_docs_passwords an array of passwords to use

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
pdf_blob_column = combineProtectedPDFDocuments(new Array(pdf_blob1,pdf_blob2,pdf_blob3), new Array
(pdf_blob1_pass,pdf_blob2_pass,pdf_blob3_pass));
```

convertPDFFormToPDFDocument(pdf_form, field_values)

Convert a PDF form to a PDF document.

Parameters

Array pdf_form the PDF Form to convert
Object field_values the values to use

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var pdfform = plugins.file.readFile('c:/temp/1040a-form.pdf');
//var field_values = plugins.file.readFile('c:/temp/1040a-data.fdf');//read adobe fdf values or
var field_values = new Array()//construct field values
field_values[0] = 'f1-1=John C.J.'
field_values[1] = 'f1-2=Longlasting'
var result_pdf_doc = plugins.pdf_output.convertPDFFormToPDFDocument(pdfform, field_values)
if (result_pdf_doc != null)
{
        plugins.file.writeFile('c:/temp/1040a-flatten.pdf', result_pdf_doc)
}
```

convertProtectedPDFFormToPDFDocument(pdf_form, pdf_password, field_values)

Convert a protected PDF form to a PDF document.

Parameters

Array	pdf_form	the PDF Form to convert
String	pdf_password	the password to use
Object	field_values	the field values to use

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var pdfform = plugins.file.readFile('c:/temp/1040a-form.pdf');
//var field_values = plugins.file.readFile('c:/temp/1040a-data.fdf');//read adobe fdf values or
var field_values = new Array()//construct field values
field_values[0] = 'f1-1=John C.J.'
field_values[1] = 'f1-2=Longlasting'
var result_pdf_doc = plugins.pdf_output.convertProtectedPDFFormToPDFDocument(pdfform, 'pdf_password',
field_values)
if (result_pdf_doc != null)
{
        plugins.file.writeFile('c:/temp/1040a-flatten.pdf', result_pdf_doc)
}
```

encrypt(data, ownerPassword)

Add password protection and security options to the PDF

Parameters

Array	data	the PDF
String	ownerPassword	the owner password

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add password protection and security options to the PDF
// NOTE: Passwords are case sensitive
var unEncryptedFile = plugins.file.showFileOpenDialog();
if (unEncryptedFile) {
        var data = plugins.file.readFile(unEncryptedFile);
        encryptedResult = %%elementName%%.encrypt(data, 'secretPassword', 'secretUserPassword', false, false,
false, false, false, false, false, false, true);
}
```

encrypt(data, ownerPassword, userPassword)

Add password protection and security options to the PDF

Parameters

Array	data	the PDF
String	ownerPassword	the owner password
String	userPassword	the user password

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add password protection and security options to the PDF
// NOTE: Passwords are case sensitive
var unEncryptedFile = plugins.file.showFileOpenDialog();
if (unEncryptedFile) {
        var data = plugins.file.readFile(unEncryptedFile);
        encryptedResult = %%elementName%%.encrypt(data, 'secretPassword', 'secretUserPassword', false, false,
false, false, false, false, false, false, true);
}
```

encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders)

Add password protection and security options to the PDF

Parameters

Array	data	the PDF
String	ownerPassword	the owner password
String	userPassword	the user password
Boolean	allowAssembly	whether to set the allow assembly permission
Boolean	allowCopy	whether to set the allow copy permission
Boolean	allowDegradedPrinting	whether to set the allow degraded printing permission
Boolean	allowFillIn	whether to set the allow fill in permission
Boolean	allowModifyAnnotations	whether to set the allow modify annotations permission
Boolean	allowModifyContents	whether to set the allow modify contents permission
Boolean	allowPrinting	whether to set the allow printing permission
Boolean	allowScreenreaders	whether to set the allow screen readers permission

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add password protection and security options to the PDF
// NOTE: Passwords are case sensitive
var unEncryptedFile = plugins.file.showFileOpenDialog();
if (unEncryptedFile) {
        var data = plugins.file.readFile(unEncryptedFile);
        encryptedResult = %%elementName%%.encrypt(data, 'secretPassword', 'secretUserPassword', false, false,
false, false, false, false, false, false, true);
}
```

encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders, is128bit)

Add password protection and security options to the PDF

Parameters

Array	data	the PDF
String	ownerPassword	the owner password
String	userPassword	the user password
Boolean	allowAssembly	whether to set the allow assembly permission
Boolean	allowCopy	whether to set the allow copy permission
Boolean	allowDegradedPrinting	whether to set the allow degraded printing permission
Boolean	allowFillIn	whether to set the allow fill in permission
Boolean	allowModifyAnnotations	whether to set the allow modify annotations permission
Boolean	allowModifyContents	whether to set the allow modify contents permission
Boolean	allowPrinting	whether to set the allow printing permission
Boolean	allowScreenreaders	whether to set the allow screen readers permission
Boolean	is128bit	whether to use 128-bit encryption

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add password protection and security options to the PDF
// NOTE: Passwords are case sensitive
var unEncryptedFile = plugins.file.showFileOpenDialog();
if (unEncryptedFile) {
        var data = plugins.file.readFile(unEncryptedFile);
        encryptedResult = %%elementName%%.encrypt(data, 'secretPassword', 'secretUserPassword', false, false,
false, false, false, false, false, false, true);
}
```

encrypt(data, ownerPassword, userPassword, allowAssembly, allowCopy, allowDegradedPrinting, allowFillIn, allowModifyAnnotations, allowModifyContents, allowPrinting, allowScreenreaders, is128bit, metaData)

Add password protection and security options to the PDF

Parameters

Array	data	the PDF
String	ownerPassword	the owner password
String	userPassword	the user password
Boolean	allowAssembly	whether to set the allow assembly permission
Boolean	allowCopy	whether to set the allow copy permission
Boolean	allowDegradedPrinting	whether to set the allow degraded printing permission
Boolean	allowFillIn	whether to set the allow fill in permission
Boolean	allowModifyAnnotations	whether to set the allow modify annotations permission
Boolean	allowModifyContents	whether to set the allow modify contents permission
Boolean	allowPrinting	whether to set the allow printing permission
Boolean	allowScreenreaders	whether to set the allow screen readers permission
Boolean	is128bit	whether to use 128-bit encryption
Object	metaData	a JavaScript object (Scriptable) that contains the metadata as property/value pairs

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add password protection and security options to the PDF
// NOTE: Passwords are case sensitive
var unEncryptedFile = plugins.file.showFileOpenDialog();
if (unEncryptedFile) {
        var data = plugins.file.readFile(unEncryptedFile);
        encryptedResult = %%elementName%%.encrypt(data, 'secretPassword', 'secretUserPassword', false, false,
false, false, false, false, false, false, true);
}
```

endMetaPrintJob()

Ends a previously started meta print job. For meta print jobs that were stored in memory, not in a file on disk, also returns the content of the generated PDF document.

Returns

Array

Supported Clients

SmartClient,WebClient

Sample

```
//to print multiple forms to one pdf document (on file system).
var success = plugins.pdf_output.startMetaPrintJob('c:/temp/out.pdf')
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
plugins.pdf_output.endMetaPrintJob()

//to print multiple forms to one pdf document (to store in dataprovider).
var success = plugins.pdf_output.startMetaPrintJob()
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
mediaDataProvider = plugins.pdf_output.endMetaPrintJob()
```

getPDFPrinter()

Returns a PDF printer that can be used in print calls. Returns the last started meta print job.

Returns

Object

Supported Clients

SmartClient,WebClient

Sample

```
//to print current record without printdialog to pdf file in temp dir.
controller.print(true,false,plugins.pdf_output.getPDFPrinter());
```

getPDFPrinter(filename)

Returns a PDF printer that can be used in print calls. The PDF printer that generates a PDF into the specified file is returned.

Parameters

String filename the file name

Returns

Object

Supported Clients

SmartClient,WebClient

Sample

```
//to print current record without printdialog to pdf file in temp dir.
controller.print(true,false,plugins.pdf_output.getPDFPrinter('c:/temp/out.pdf'));
```

getPagesPrinted()

Returns the number of pages printed by the last print call done in the context of a meta print job.

Returns

Number

Supported Clients

SmartClient,WebClient

Sample

```
//to print multiple forms to one pdf document (on file system).
var success = plugins.pdf_output.startMetaPrintJob('c:/temp/out.pdf')
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
plugins.pdf_output.endMetaPrintJob()

//to print multiple forms to one pdf document (to store in dataprovider).
var success = plugins.pdf_output.startMetaPrintJob()
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
mediaDataProvider = plugins.pdf_output.endMetaPrintJob()
```

getThumbnailImage(data)

Create a thumbnail from the provided PDF

Parameters

Array data the PDF

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//read PDF file data
var pdf = plugins.file.readFile();
//get the thumbnail (default the first page is rendered with 72 dpi resolution)
var pngImg = plugins.pdf_output.getThumbnailImage(pdf);
//save PNG image to file
var thumbnailFile = plugins.file.convertToJSFile()
plugins.file.writeFile(thumbnailFile, pngImg);
```

getThumbnailImage(data, pageNumber)

Create a thumbnail from the provided PDF

Parameters

Array data the PDF

Number pageNumber PDF page to get thumbnail of. This parameter is zero based index.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//read PDF file data
var pdf = plugins.file.readFile();
//get the thumbnail (default the first page is rendered with 72 dpi resolution)
var pngImg = plugins.pdf_output.getThumbnailImage(pdf);
//save PNG image to file
var thumbnailFile = plugins.file.convertToJSFile()
plugins.file.writeFile(thumbnailFile, pngImg);
```

getThumbnailImage(data, pageNumber, dpi)

Create a thumbnail from the provided PDF

Parameters

Array	data	the PDF
Number	pageNumber	PDF page to get thumbnail of. This parameter is zero based index.
Number	dpi	resolution used to render the thumbnail image

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//read PDF file data
var pdf = plugins.file.readFile();
//get the thumbnail (default the first page is rendered with 72 dpi resolution)
var pngImg = plugins.pdf_output.getThumbnailImage(pdf);
//save PNG image to file
var thumbnailFile = plugins.file.convertToJSFile()
plugins.file.writeFile(thumbnailFile, pngImg);
```

getTotalPagesPrinted()

Returns the total number of pages printed in the context of a meta print job. Call this method before ending the meta print job.

Returns

Number

Supported Clients

SmartClient,WebClient

Sample

```
//to print multiple forms to one pdf document (on file system).
var success = plugins.pdf_output.startMetaPrintJob('c:/temp/out.pdf')
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
plugins.pdf_output.endMetaPrintJob()

//to print multiple forms to one pdf document (to store in dataprovider).
var success = plugins.pdf_output.startMetaPrintJob()
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
mediaDataProvider = plugins.pdf_output.endMetaPrintJob()
```

insertFontDirectory(path)

Add a directory that should be searched for fonts. Call this only in the context of an active meta print job.

Parameters

String path the path to use

Returns

Number

Supported Clients

SmartClient,WebClient

Sample

```
//Insert font directories for font embedding.
//You must create an MetaPrintJob before using it.
plugins.pdf_output.insertFontDirectory('c:/Windows/Fonts');
plugins.pdf_output.insertFontDirectory('c:/WinNT/Fonts');
plugins.pdf_output.insertFontDirectory('/Library/Fonts');
```

numberPages(data)

Add pages numbers to a PDF

Parameters

Array data the PDF

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add pages numbers to a PDF
var unNumberedFile = plugins.file.showFileOpenDialog();
if (unNumberedFile) {
        var data = plugins.file.readFile(unNumberedFile);
        pageNumberedPdf = %%elementName%%.numberPages(data, 12, 520, 30, 'Courier', '#ff0033');
}
```

numberPages(data, fontSize, locationX, locationY, font, hexColor)

Add pages numbers to a PDF

Parameters

Array	data	the PDF
Number	fontSize	the font size to use
Number	locationX	the x location of the numbers
Number	locationY	the y location of the numbers
String	font	the font to use
String	hexColor	the font color to use

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add pages numbers to a PDF
var unNumberedFile = plugins.file.showFileOpenDialog();
if (unNumberedFile) {
        var data = plugins.file.readFile(unNumberedFile);
        pageNumberedPdf = %%elementName%%.numberPages(data, 12, 520, 30, 'Courier', '#ff0033');
}
```

overlay(data, forOverlay)

Add some PDF based content over a PDF

Parameters

Array data the PDF
Array forOverlay a PDF to use as overlay

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add some PDF based content over a PDF
var pages = new Array();
pages[0] = '1';
pages[1] = '3';
pages[2] = '5';
var input1 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select source file');
if (input1) {
        var data = plugins.file.readFile(input1);
        var input2 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select file for overlay');
        if (input2) {
                var data2 = plugins.file.readFile(input2);
                overlayedPdf = %%elementName%%.overlay( data, data2, false, pages );
                //overlayedPdf = %%elementName%%.overlay( data, data2 );
                //overlayedPdf = %%elementName%%.overlay( data, data2, false, null );
                //overlayedPdf = %%elementName%%.overlay( data, data2, pages );
        }
}
```

overlay(data, forOverlay, isOver)

Add some PDF based content over a PDF

Parameters

Array data the PDF
Array forOverlay a PDF to use as overlay
Boolean isOver whether the overlay will be put over the content

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add some PDF based content over a PDF
var pages = new Array();
pages[0] = '1';
pages[1] = '3';
pages[2] = '5';
var input1 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select source file');
if (input1) {
        var data = plugins.file.readFile(input1);
        var input2 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select file for overlay');
        if (input2) {
                var data2 = plugins.file.readFile(input2);
                overlayedPdf = %%elementName%%.overlay( data, data2, false, pages );
                //overlayedPdf = %%elementName%%.overlay( data, data2 );
                //overlayedPdf = %%elementName%%.overlay( data, data2, false, null );
                //overlayedPdf = %%elementName%%.overlay( data, data2, pages );
        }
}
```

overlay(data, forOverlay, isOver, pages)

Add some PDF based content over a PDF

Parameters

Array	data	the PDF
Array	forOverlay	a PDF to use as overlay
Boolean	isOver	whether the overlay will be put over the content
Array	pages	an array of page numbers to put the overlay on

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add some PDF based content over a PDF
var pages = new Array();
pages[0] = '1';
pages[1] = '3';
pages[2] = '5';
var input1 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select source file');
if (input1) {
        var data = plugins.file.readFile(input1);
        var input2 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select file for overlay');
        if (input2) {
                var data2 = plugins.file.readFile(input2);
                overlayedPdf = %%elementName%%.overlay( data, data2, false, pages );
                //overlayedPdf = %%elementName%%.overlay( data, data2 );
                //overlayedPdf = %%elementName%%.overlay( data, data2, false, null );
                //overlayedPdf = %%elementName%%.overlay( data, data2, pages );
        }
}
```

overlay(data, forOverlay, pages)

Add some PDF based content over a PDF

Parameters

Array	data	the PDF
Array	forOverlay	a PDF to use as overlay
Array	pages	an array of page numbers to put the overlay on

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add some PDF based content over a PDF
var pages = new Array();
pages[0] = '1';
pages[1] = '3';
pages[2] = '5';
var input1 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select source file');
if (input1) {
        var data = plugins.file.readFile(input1);
        var input2 = plugins.file.showFileOpenDialog(1,null,false,'pdf',null,'Select file for overlay');
        if (input2) {
                var data2 = plugins.file.readFile(input2);
                overlayedPdf = %%elementName%%.overlay( data, data2, false, pages );
                //overlayedPdf = %%elementName%%.overlay( data, data2 );
                //overlayedPdf = %%elementName%%.overlay( data, data2, false, null );
                //overlayedPdf = %%elementName%%.overlay( data, data2, pages );
        }
}
```

overlayText(data, text)

Add text over every page at a 45 degree angle

Parameters

Array data the PDF
String text the text to use for the overlay

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add text over every page at a 45 degree angle\m
var pdf = plugins.file.showFileOpenDialog();
if (pdf) {
        var data = plugins.file.readFile(pdf);
        modifiedPdf = %%elementName%%.overlayText(data, 'DRAFT', 230, 430, true, 32, 'Helvetica', '#33ff33');
}
```

overlayText(data, text, locationX, locationY, isOver, fontSize, font, hexColor)

Add text over every page at a 45 degree angle

Parameters

Array	data	the PDF
String	text	the text to use for the overlay
Number	locationX	the x location of the overlay
Number	locationY	the y location of the overlay
Boolean	isOver	whether to put the overlay over the content
Number	fontSize	the font size to use
String	font	the font to use
String	hexColor	the font color to use

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add text over every page at a 45 degree angle\m
var pdf = plugins.file.showFileOpenDialog();
if (pdf) {
        var data = plugins.file.readFile(pdf);
        modifiedPdf = %%elementName%%.overlayText(data, 'DRAFT', 230, 430, true, 32, 'Helvetica', '#33ff33');
}
```

startMetaPrintJob()

Used for printing multiple things into the same PDF document. Starts a meta print job and all print calls made before ending the meta print job will be done into the same PDF document. The PDF document is stored in memory and can be retrieved when ending the meta print job and can be saved, for example, into a dataprovider.

Returns

Boolean

Supported Clients

SmartClient,WebClient

Sample

```
//to print multiple forms to one pdf document (on file system).
var success = plugins.pdf_output.startMetaPrintJob('c:/temp/out.pdf')
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
plugins.pdf_output.endMetaPrintJob()

//to print multiple forms to one pdf document (to store in dataprovider).
var success = plugins.pdf_output.startMetaPrintJob()
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
mediaDataProvider = plugins.pdf_output.endMetaPrintJob()
```

startMetaPrintJob(filename)

Used for printing multiple things into the same PDF document. Starts a meta print job and all print calls made before ending the meta print job will be done into the same PDF document. The PDF document is generated in a File specified by the filename.

Parameters

String filename the file name

Returns

Boolean

Supported Clients

SmartClient,WebClient

Sample

```
//to print multiple forms to one pdf document (on file system).
var success = plugins.pdf_output.startMetaPrintJob('c:/temp/out.pdf')
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
plugins.pdf_output.endMetaPrintJob()

//to print multiple forms to one pdf document (to store in dataprovider).
var success = plugins.pdf_output.startMetaPrintJob()
if (success)
{
	forms.form_one.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form one printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
	forms.form_two.controller.print(false,false,plugins.pdf_output.getPDFPrinter());
	application.output('form two printed ' + plugins.pdf_output.getPagesPrinted() + ' pages.');
}
application.output('total printed pages: ' + plugins.pdf_output.getTotalPagesPrinted());
mediaDataProvider = plugins.pdf_output.endMetaPrintJob()
```

watermark(data, image)

Add an image as a watermark on every page, or the pages specified as a parameter

Parameters

Array	data	the PDF
Object	image	the path of an image to use or array of bytes containing actual image

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Add an image as a watermark on every page, or the pages specified as a parameter.
var pdf = plugins.file.showFileOpenDialog();
if (pdf) {
        var data = plugins.file.readFile(pdf);
        var image = plugins.file.showFileOpenDialog();
        modifiedPdf = %%elementName%%.watermark(data, image);
}
```

watermark(data, image, locationX, locationY, isOver)

Add an image as a watermark on every page, or the pages specified as a parameter

Parameters

Array	data	the PDF
Object	image	the path of an image to use or array of bytes containing actual image
Number	locationX	the x location of the image
Number	locationY	the y location of the image
Boolean	isOver	whether to put over the content

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

watermark(data, image, locationX, locationY, isOver, pages)

Add an image as a watermark on every page, or the pages specified as a parameter

Parameters

Array	data	the PDF
Object	image	the path of an image to use or array of bytes containing actual image
Number	locationX	the x location of the image
Number	locationY	the y location of the image
Boolean	isOver	whether to put over the content
Array	pages	an array of pages where to apply the watermark

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample