White Paper pdfServMachine

Isarweg 6
D-42697 Solingen

fon support +49.208.780.38.18
fon sales +49.178.780.38.18
fax +49.208.780.38.19

info@broadgun.de
www.broadgun.de

Overview

pdfServMachine is a Software Development Kit (SDK) for PDF conversion and manipulation.

• Easily convert your HTML, text and Microsoft Office files to PDF.

• Integrate PDF generation into virtually any application that can print.

• Manipulate existing PDF documents.

• pdfServMachine runs on Windows NT/2000/XP/2003 and exposes a COM API.

Prerequisites

• pdfMachine 10.2 or latter must be installed. The latest version is available at http://pdfmachine.de
• Internet Explorer 6

Installation - Out of Process COM Server

1. Download
Down and unzip the SDK to a suitable directory.
e.g. c:\pdfServMachine

2. Register the COM server
From the command line, run:

```
pdfServMachine /RegServer
```

Depending upon the your security settings, you may be able to now use pdfServMachine or you may need use dcomcnfg.exe to add your users to the access list. See the section on dcomcnfg.exe in the service installation instructions if required.

Installation - Service

Before installing the service, think very hard as to whether this is the way to go for you, as it adds a lot of extra configuration steps. In most cases it is unnecessary to install pdfServMachine as a service.

1. Download
Down and unzip the SDK to a suitable directory.
e.g. c:\pdfServMachine

2. Install the service
From the command line, run:

```
pdfServMachine /Service
```

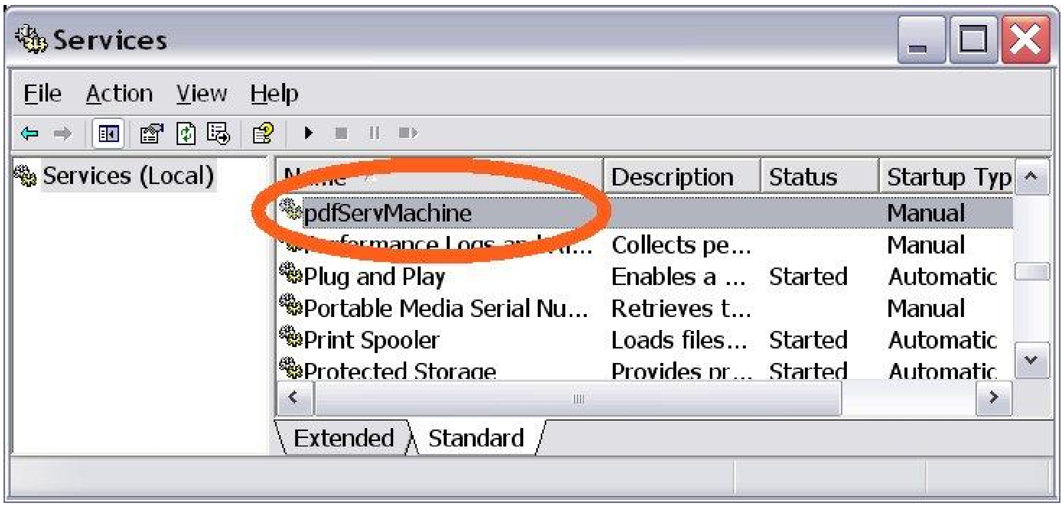
3. Set the logon account for the service

Run the services applet.

Launch the services applet. e.g. Start->run then type:

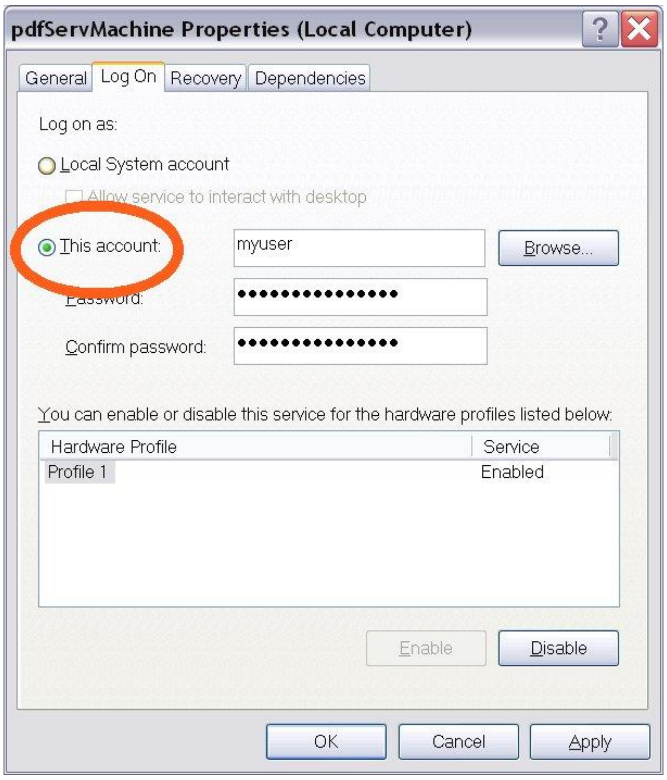
```
services.msc
```

Select the pdfServMachine from the list of services.



Name	Description	Status	Startup Typ
pdfServMachine			Manual
Performance Logs and Al...	Collects pe...		Manual
Plug and Play	Enables a ...	Started	Automatic
Portable Media Serial Nu...	Retrieves t...		Manual
Print Spooler	Loads files...	Started	Automatic
Protected Storage	Provides pr...	Started	Automatic

Double click on the service, then switch to the logon tab.
Select "This Account" and enter in a user with Administrator rights.
Click OK.



Log on as:

Local System account

Allow service to interact with desktop

[✓ This account:] myuser

Password: •••••••••••••••

Confirm password: •••••••••••••••

You can enable or disable this service for the hardware profiles listed below:

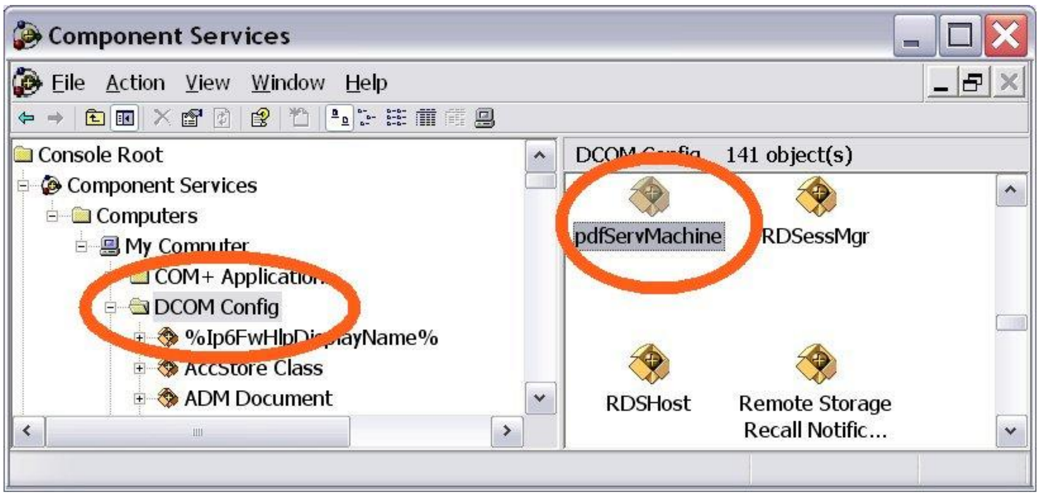
Hardware Profile	Service
Profile 1	Enabled

4. Configure security with dcomcnfg.exe

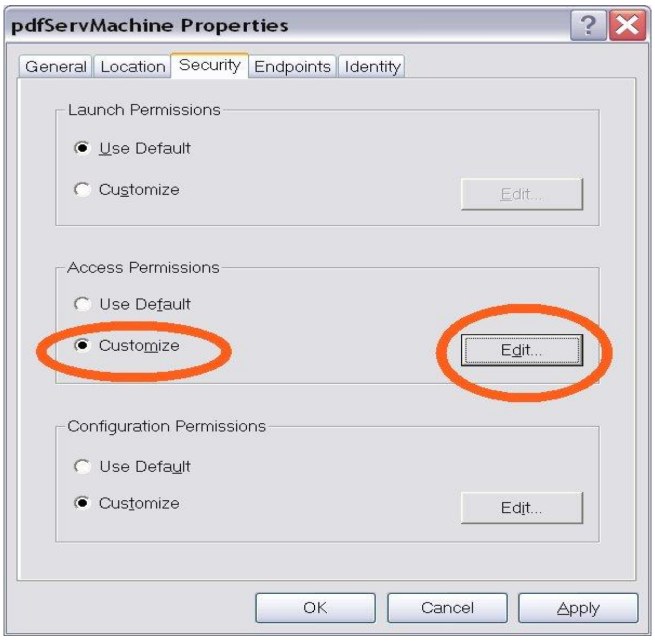
e.g. Start->run then type:

dcomcnfg

Find the pdfServMachine entry, as shown below.



Right click on the pdfSerMachine icon, select Properties, than select the "Security" tab, as shown below.


pdfServMachine Properties

General | Location | Security | Endpoints | Identity

Launch Permissions
- [✓ Use Default]
- Customize
- Edit...

Access Permissions
- Use Default
- [✓ Customize]
- Edit...

Configuration Permissions
- Use Default
- [✓ Customize]
- Edit...

OK | Cancel | Apply

Edit the "Access Permissions" to include the user account or group that matches the account the service will be run under. Apply the changes and exit dcomcnfg.

5. Start the service.

From within the services.msc view, optionally set the "start up" type of the service to Automatic, than start the service.

Installation - Service

Before installing the service, think very hard as to whether this is the way to go for you, as it adds a lot of extra configuration steps. In most cases it is unnecessary to install pdfServMachine as a service.

1. Download

Down and unzip the SDK to a suitable directory.
 e.g. c:\pdfServMachine

2. Install the service

From the command line, run:

```
pdfServMachine /Service
```

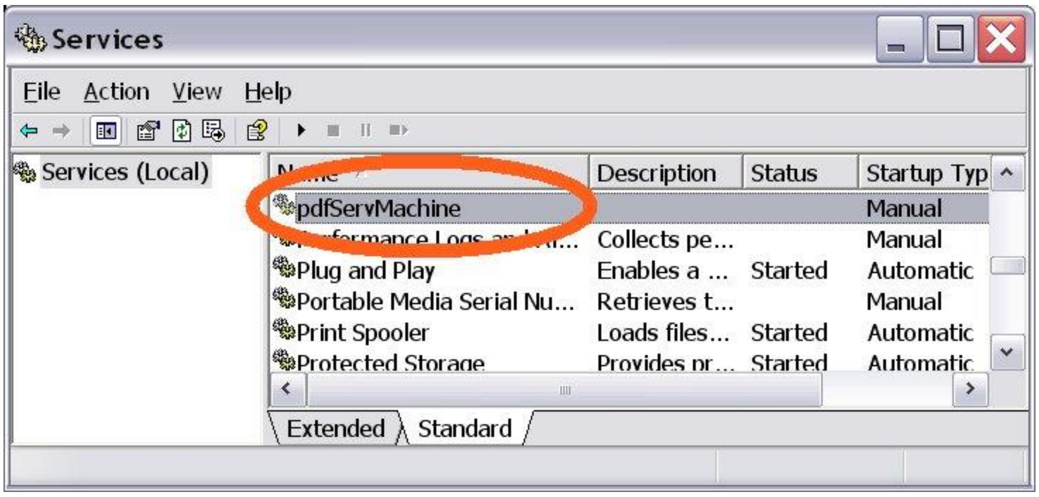
3. Set the logon account for the service

Run the services applet.

Launch the services applet. e.g. Start->run then type:

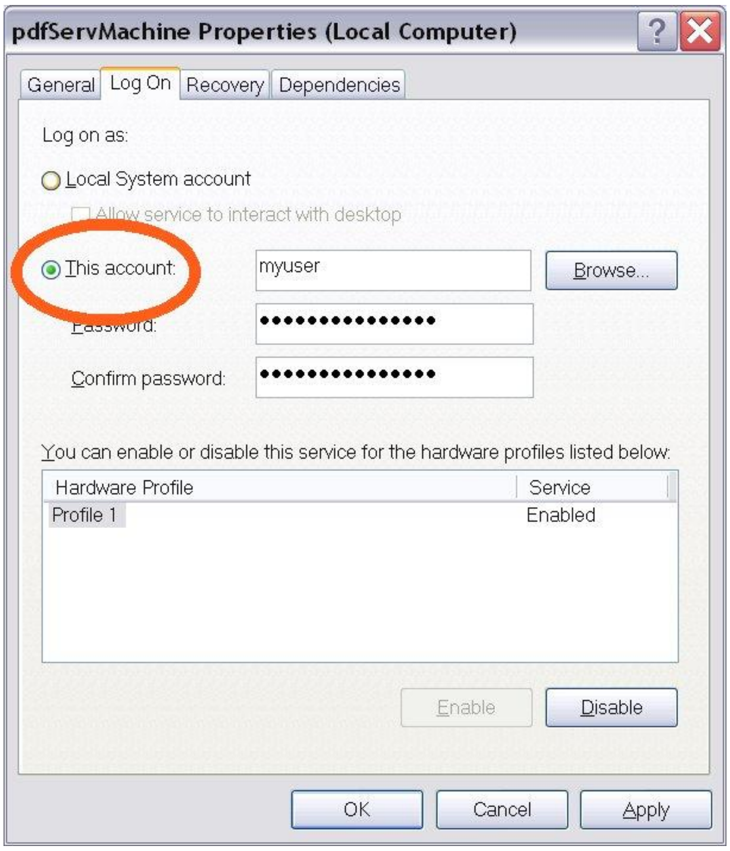
```
services.msc
```

Select the pdfServMachine from the list of services.



Name	Description	Status	Startup Typ
pdfServMachine			Manual
Performance Logs and A...	Collects pe...		Manual
Plug and Play	Enables a ...	Started	Automatic
Portable Media Serial Nu...	Retrieves t...		Manual
Print Spooler	Loads files...	Started	Automatic
Protected Storage	Provides pr...	Started	Automatic

Double click on the service, then switch to the logon tab.
Select "This Account" and enter in a user with Administrator rights.
Click OK.


pdfServMachine Properties (Local Computer)

General | Log On | Recovery | Dependencies

Log on as:

○ Local System account

☐ Allow service to interact with desktop

● This account: myuser [Browse...]

Password: •••••••••••••••

Confirm password: •••••••••••••••

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Profile 1	Enabled

[Enable] [Disable]

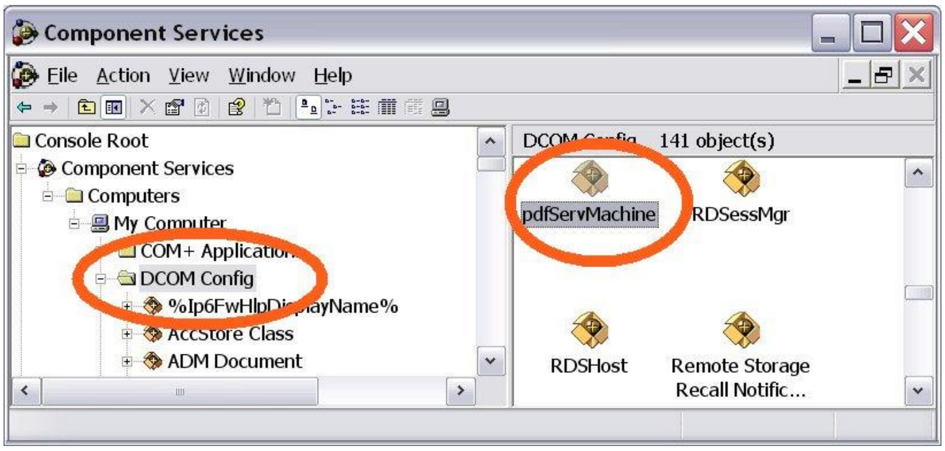
[OK] [Cancel] [Apply]

4. Configure security with dcomcnfg.exe

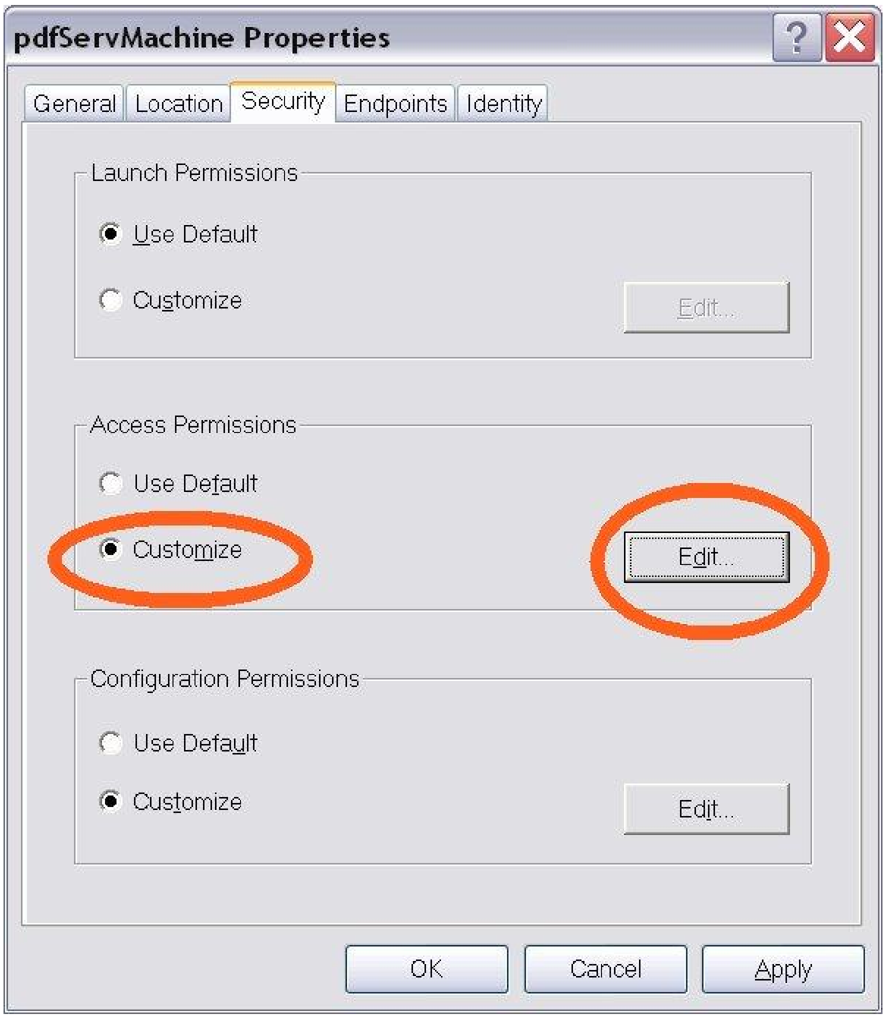
e.g. Start->run then type:

```
dcomcnfg
```

Find the pdfServMachine entry, as shown below.



Right click on the pdfSerMachine icon, select Properties, then select the "Security" tab, as shown below.



pdfServMachine Properties

General | Location | Security | Endpoints | Identity

Launch Permissions
- (●) Use Default
- ( ) Customize — Edit...

Access Permissions
- ( ) Use Default
- (●) Customize — Edit...

Configuration Permissions
- ( ) Use Default
- (●) Customize — Edit...

OK | Cancel | Apply

Edit the "Access Permissions" to include the user account or group that matches the account the service will be run under. Apply the changes and exit dcomcnfg.

5. Start the service.

From within the services.msc view, optionally set the "start up" type of the service to Automatic, then start the service.

Un-Installation

From the command line, run:

pdfServMachine.exe /UnregServer

Examples

Convert to PDF example

Look at the following sample javascript to see how easy it is to use pdfServMachine:

```
// Example javascript script to convert the
// www.google.com web page to a PDF file

var conv = new ActiveXObject("pdfServMachine.converter");
conv.convert("http://www.google.com", "c:\\google.pdf");
WScript.Echo("finished conversion");
```

Convert to PDF example - calling app does print

This time around the calling application starts the windows print job. The file c:\x.txt is converted to the PDF file c:\x.pdf, but Notepad.exe does the printing.

```
var conv = new ActiveXObject("pdfServMachine.converter");
var shell = new ActiveXObject("WScript.Shell");

conv.printJobStart("c:\\x.pdf", true);
shell.Run("notepad.exe /p c:\\x.txt");
conv.printJobEnd(true, false);
WScript.Echo("finished conversion");
```

Append PDF example

```
// Example javascript script to that merges
// several PDF's into one big PDF.

var pdf = new ActiveXObject("pdfServMachine.Pdf");
pdf.append("c:\\afile.pdf");
pdf.append("c:\\bfile.pdf");
pdf.append("c:\\cfile.pdf");
pdf.saveAs("c:\\mergedFile.pdf");
```

C++ Example

```
// Example that generates a PDF file by explicitly
// printing to the pdfMachine printer.

// Compiled with msvc ++ 2003
// Compiler command line used: cl /EHsc print2pdf.cpp


#include <stdio.h>
#include <tchar.h>
```

```
#include <time.h>

// generates all the code for the smart pointers
#import "c:/dev/pdfservmachine/release/pdfservmachine.exe"

const TCHAR *errMsg = 0;

// initializes a devmode for printing
void initDevmode(DEVMODE *dm)
{
memset(dm, 0, sizeof(DEVMODE));

dm->dmSize = sizeof(DEVMODE);
dm->dmOrientation = DMORIENT_PORTRAIT ;
_tcscpy((TCHAR*)dm->dmFormName, TEXT("A4"));
dm->dmFields = DM_ORIENTATION | DM_FORMNAME;
}


// Prints "numPages" of boring text to the printer "printerName".
bool doPrint(const TCHAR *printerName, int numPages)
{
if (printerName == 0 || _tcslen(printerName) == 0 )
printerName = _T("Broadgun pdfMachine");

DEVMODE dm;
initDevmode(&dm);

HDC dc = CreateDC(0, printerName, 0,&dm);
if (!dc)
{
errMsg = "CreateDC failed";
return false;
}

DOCINFO di = {0};
di.cbSize = sizeof(di);
di.lpszDocName = "doc name";
int rc = StartDoc(dc, &di);
if (rc == SP_ERROR)
{
errMsg = "StartDoc failed";
return false;
}

char str[100];
sprintf(str, "On Page %d", 0);

SIZE sz;
GetTextExtentPoint32(dc, str, (int)strlen(str), &sz);

for (int i = 0; i < numPages; i++)
{
rc = StartPage(dc);
if (rc <= 0)
{
errMsg = "StartPage failed";
return false;
}
SetMapMode(dc, MM_TEXT);
LOGFONT lf = {0};
```

```
lf.lfHeight = 50;
strcpy(lf.lfFaceName, "Arial");
HFONT font = CreateFontIndirect(&lf);
HFONT oldFont = (HFONT)SelectObject(dc, font);

sprintf(str, "On Page %d", i);
for (int j = 0; j < 2; j++)
{
TextOut(dc, 0, sz.cy*j, str, (int)strlen(str));
}
rc = EndPage(dc);
if (rc <= 0)
{
errMsg = "EndPage failed";
return false;
}
SelectObject(dc, oldFont);
DeleteObject(font);
}

rc = EndDoc(dc);
if (rc <= 0)
{
errMsg = "EndDoc failed";
return false;
}
DeleteDC(dc);
return true;
}

// Generates a PDF files.
// The number of files generated is controlled by the "numFiles".
void doTest(int numFiles)
{
TCHAR filename[MAX_PATH];

pdfServMachineLib::IconverterPtr conv;
conv.CreateInstance(L"pdfServMachine.converter");
if (conv == 0)
{
_tprintf(_T("Error: could not create pdfServMachine.converter\n"));
return;
}

srand(time(0));
int randNum = rand();

for (int i = 0; i < numFiles; i++)
{
_stprintf(filename, _T("C:/tmp/Ptest_%d_%d.pdf"), randNum, i);
if (!conv->printJobStart(_bstr_t(filename), false))
goto fail;

if (!doPrint(_T("Broadgun pdfMachine"), 10))
{
_tprintf(_T("print doc failed: %s\n"), errMsg);
break;
}

if (!conv->printJobEnd(false, false))
goto fail;
```

```

}
_tprintf(_T("Created %d files. Last file [%s].\n"), i, filename);
return;

fail:
{
_bstr_t err = conv->getErrorMessage();
_tprintf(_T("Error: %s\n"), (const TCHAR *)err);
_tprintf(_T("Created %d files. Last file [%s].\n"), i, filename);
}


}


void _tmain(int argc, TCHAR *argv[])
{
// read in command line parameter for number of files to generate
int numFiles = 1;
if (argc > 1)
numFiles = _ttoi(argv[1]);

CoInitialize(0);
try
{
doTest(numFiles);
}
catch (_com_error &e)
{
_tprintf(_T("COM Error: %d. %s\n"), e.Error(),
        (const TCHAR*)e.ErrorMessage());
}
CoUninitialize();
}
```

COM API for "pdfServMachine.converter"

convert

Converts a document or web page to a PDF file.

```
BOOL convert(   BSTR inFileName,
                BSTR outFileName,
                BOOL postProcess )
```

Parameters

BSTR inFileName

The full path of the document to be converted to PDF. This can be a document that can be printed with the Shellexec api, such as a MS Word file or a text file or it can be the url of a web page.

BSTR outFileName

The resulting PDF filename.

BOOL postProcess

If TRUE, normal pdfMachine processing of the PDF file occurs applying features such as encryption, N-Up, stationery and doc. info. - if they are enabled. These features can be enabled within the pdfMachine options dialog, or by directly setting registry entries.

OEM & SDKs for Developers

Please email hjnolden@broadgun.de if you wish to obtain a license to integrate pdfMachine into your application.

Registry Settings

Certain behaviour of pdfMachine can be controlled via Windows registry settings. This should only be done by people who are comfortable with editing the registry.

The registry key for versions prior to 7.4 :

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors\PDF Port
Monitor\Ports\PDFPORT1:
```

In versions 7.4 and up, the registry settings are now stored under the key:

```
HKEY_CURRENT_USER\Software\pdfMachine\BroadGun pdfMachine
```

Registry Value Name	Registry Type	Description
SaveFilenameAction	REG_DWORD	Controls the filename of the resulting PDF file. **Valid Values** / **Description** 0 — The user is prompted for filename. 1 — Use the values for "DefaultSaveDir" and "DefaultSaveFilename" to generate a file name. This file will be overwritten without any prompting. 2 — Use the document title for the filename. The file will be saved in the "DefaultSaveDir" directory. The document title is obtained from the pDocName member of the DOC_INFO_1 print job structure. This gives developers the opportunity to control the filename programmatically. Note that is value may be automatically converted into a legal filename if it contains illegal filename characters. 3 — Same as '2' and also display the 'wait' dialog

		box with the gear animation. 4 Generate a random filename in the "DefaultSaveDir"..
DefaultSaveDir	REG_SZ	The directory that is used to place the file when "SaveFilenameAction" is equal to 1 or 4.
DefaultSaveFilename	REG_SZ	The default filename that is used when "SaveFilenameAction" is equal to 1.
NextAction	REG_DWORD	Controls what happens after the PDF file has been generated. **Valid Values** **Action** 0 Display the small "What do you want to do next" dialog. 1 Do nothing. No next action takes place. 2 Launch default PDF file viewer. 3 Execute the command line that is defined by the registry value "ExecCmdLine". 4 Send email
ExecCmdLine	REG_SZ	When "NextAction" is equal to 3, this is the command that is executed. It may contain a string such as "%s" which will be substituted with the full path of the generated PDF file. e.g. ExecCmdLine = c:\winnt\notepad.exe "%s" This would launch the notepad.exe program upon completion of file generation. The notepad program would open the PDF file.
EmailOneShot	REG_DWORD	If set and not zero, an email will be sent, regardless of other settings. The value is then set to zero after use. This is useful for an application that wants to send an email but not upset any other settings. This feature was introduced in version 8.9.
SaveFileOneShot	REG_SZ	If set, it contains the full path and name of the PDF that will be generated. No user interface will be displayed. This is useful for an application that wants to save a PDF file but not upset any other settings. This feature was introduced in version 8.9.
EmailTo	REG_SZ	If set provide default values for the default mail client when it is launched. If the first character is an '@' then the rest of the text is

		taken to be a filename and the text is read from the file. e.g if the "Email Body" value was "@c:\body.txt" then the file c:\body.txt would be read in and this would be placed in the body of the email. For the TO, CC, BCC address fields, several addresses may appear if the addresses are comma or white space separated. Some email applications require the "SMTP:" prefix on mail addresses to send SMTP style mail.
EmailCC	REG_SZ	When an email is to be sent, this value is used.
EmailBCC	REG_SZ	When an email is to be sent, this value is used.
EmailSubject	REG_SZ	When an email is to be sent, this value is used.
EmailBody	REG_SZ	When an email is to be sent, this value is used.
UserConfirmAttachmentName	REG_DWORD	If set to 1, then the user is prompted for the attachment name prior to sending an email.
AutomatedMapiSend	REG_DWORD	If set to 1 and a EmailTo entry is present, then the email is sent automatically, without user intervention.
StationeryEnableStationeryAllPages	REG_DWORD	If set to 1, the file pointed to by "stationeryPathStationeryAllPages" is used as a stationery file for every page.
StationeryEnableStationeryFirstPage	REG_DWORD	If set to 1, the file pointed to by "stationeryPathStationeryFirstPage" is used as a stationery file for the first page.
StationeryOnTopStationeryAllPages	REG_DWORD	If set to 1, stationery is drawn last.
StationeryOnTopStationeryFirstPage	REG_DWORD	If set to 1, stationery on first page is drawn last.
StationeryPathStationeryAllPages	REG_SZ	The fullpath of the PDF file used for stationery.
StationeryPathStationeryFirstPage	REG_SZ	The fullpath of the PDF file used for stationery on the first page.
sd_enabled	REG_DWORD	If set to 1, security is enabled.
sd_keyLen	REG_DWORD	set to either 5 for 40 bit encryption, or 16 for 128 bit encryption.

Calling pdfMachine from an application

The easiest way to generate a PDF file from an application is to first set either registry entries EmailOneShot or SaveFileOneShot . Then print your document to the "Broadgun pdfMachine" printer.

Returns

TRUE if success, otherwise FALSE.

printJobStart

Used when the calling application is starting the Windows print job. Must be followed by a printJobEnd() after the print job has been started.

```
BOOL printJobStart(BSTR outFileName, BOOL setDefPrinter)
```

Parameters

BSTR outFileName

The resulting PDF filename.

BOOL setDefPrinter

If TRUE, the Windows default printer is set to the current pdfMachine printer, usually "Broadgun pdfMachine". The default printer prior to this call can be restored by setting the "restoreDefPrinter" flag in the corresonding printJobEnd call.

Returns

TRUE if success, otherwise FALSE.

printJobEnd

Ends a print job previously opened with printJobStart(). This call blocks until the print spooler has finished printing.

```
BOOL printJobEnd(BOOL restoreDefPrinter, BOOL postProcess)
```

Parameters

BOOL restoreDefPrinter

If TRUE, the default windows printer is restored to what it was previously, otherwise it is left alone.

BOOL postProcess

If TRUE, normal pdfMachine processing of the PDF file occurs applying features such as encryption, N-Up, stationery and doc. info. - if they are enabled. These features can be enabled within the pdfMachine options dialog, or by directly setting registry entries.

OEM & SDKs for Developers

Please email hjnolden@broadgun.de if you wish to obtain a license to integrate pdfMachine into your application.

Registry Settings

Certain behaviour of pdfMachine can be controlled via Windows registry settings. This should only be done by people who are comfortable with editing the registry.

The registry key for versions prior to 7.4 :

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors\PDF Port
Monitor\Ports\PDFPORT1:
```

In versions 7.4 and up, the registry settings are now stored under the key:

```
HKEY_CURRENT_USER\Software\pdfMachine\BroadGun pdfMachine
```

Registry Value Name	Registry Type	Description
SaveFilenameAction	REG_DWORD	Controls the filename of the resulting PDF file. **Valid Values** / **Description** 0 — The user is prompted for filename. 1 — Use the values for "DefaultSaveDir" and "DefaultSaveFilename" to generate a file name. This file will be overwritten without any prompting. 2 — Use the document title for the filename. The file will be saved in the "DefaultSaveDir" directory. The document title is obtained from the pDocName member of the DOC_INFO_1 print job structure. This gives developers the opportunity to control the filename programmatically. Note that is value may be automatically converted into a legal filename if it contains illegal filename characters. 3 — Same as '2' and also display the 'wait' dialog box with the gear animation. 4 — Generate a random filename in the "DefaultSaveDir"..

DefaultSaveDir	REG_SZ	The directory that is used to place the file when "SaveFilenameAction" is equal to 1 or 4.
DefaultSaveFilename	REG_SZ	The default filename that is used when "SaveFilenameAction" is equal to 1.
NextAction	REG_DWORD	Controls what happens after the PDF file has been generated. **Valid Values** — **Action** 0 — Display the small "What do you want to do next" dialog. 1 — Do nothing. No next action takes place. 2 — Launch default PDF file viewer. 3 — Execute the command line that is defined by the registry value "ExecCmdLine". 4 — Send email
ExecCmdLine	REG_SZ	When "NextAction" is equal to 3, this is the command that is executed. It may contain a string such as "%s" which will be substituted with the full path of the generated PDF file. e.g. ExecCmdLine = c:\winnt\notepad.exe "%s" This would launch the notepad.exe program upon completion of file generation. The notepad program would open the PDF file.
EmailOneShot	REG_DWORD	If set and not zero, an email will be sent, regardless of other settings. The value is then set to zero after use. This is useful for an application that wants to send an email but not upset any other settings. This feature was introduced in version 8.9.
SaveFileOneShot	REG_SZ	If set, it contains the full path and name of the PDF that will be generated. No user interface will be displayed. This is useful for an application that wants to save a PDF file but not upset any other settings. This feature was introduced in version 8.9.
EmailTo	REG_SZ	If set provide default values for the default mail client when it is launched. If the first character is an '@' then the rest of the text is taken to be a filename and the text is read from the file. e.g if the "Email Body" value was "@c:\body.txt" then

		the file c:\body.txt would be read in and this would be placed in the body of the email. For the TO, CC, BCC address fields, several addresses may appear if the addresses are comma or white space separated. Some email applications require the "SMTP:" prefix on mail addresses to send SMTP style mail.
EmailCC	REG_SZ	When an email is to be sent, this value is used.
EmailBCC	REG_SZ	When an email is to be sent, this value is used.
EmailSubject	REG_SZ	When an email is to be sent, this value is used.
EmailBody	REG_SZ	When an email is to be sent, this value is used.
UserConfirmAttachmentName	REG_DWORD	If set to 1, then the user is prompted for the attachment name prior to sending an email.
AutomatedMapiSend	REG_DWORD	If set to 1 and a EmailTo entry is present, then the email is sent automatically, without user intervention. .
StationeryEnableStationeryAllPages	REG_DWORD	If set to 1, the file pointed to by "stationeryPathStationeryAllPages" is used as a stationery file for every page.
StationeryEnableStationeryFirstPage	REG_DWORD	If set to 1, the file pointed to by "stationeryPathStationeryFirstPage" is used as a stationery file for the first page.
StationeryOnTopStationeryAllPages	REG_DWORD	If set to 1, stationery is drawn last.
StationeryOnTopStationeryFirstPage	REG_DWORD	If set to 1, stationery on first page is drawn last.
StationeryPathStationeryAllPages	REG_SZ	The fullpath of the PDF file used for stationery.
StationeryPathStationeryFirstPage	REG_SZ	The fullpath of the PDF file used for stationery on the first page.
sd_enabled	REG_DWORD	If set to 1, security is enabled.
sd_keyLen	REG_DWORD	set to either 5 for 40 bit encryption, or 16 for 128 bit encryption.

Calling pdfMachine from an application

The easiest way to generate a PDF file from an application is to first set either registry entries EmailOneShot or SaveFileOneShot . Then print your document to the "Broadgun pdfMachine" printer.

Returns

TRUE if success, otherwise FALSE.

COM API for "pdfServMachine.pdf"

open

Opens a PDF file ready for manipulation.

```
BOOL open(BSTR fileName)
```

Parameters

BSTR fileName

The full path of the PDF file to be opened.

Returns

TRUE if success, otherwise FALSE.

saveAs

Creates a PDF file containing the result of the PDF manipulation.

```
BOOL saveAs(BSTR fileName)
```

Parameters

BSTR fileName

The full path of the PDF file to be written to.

Returns

TRUE if success, otherwise FALSE.

append

Opens the PDF file 'fileName' and appends the contents to our internal representation.

```
BOOL append(BSTR fileName)
```

Parameters

BSTR fileName

The full path of the PDF file that will be read in.

Returns

TRUE if success, otherwise FALSE.

deletePage

Deletes a page.

```
BOOL deletePage(int pageNum)
```

Parameters

int pageNum

The number of the page to be deleted.

Returns

TRUE if success, otherwise FALSE.

getNumPages

Deletes a page.

```
int getNumPages()
```

Returns

The number of pages.

setStationery

Applies stationery. The first page of the 'stationeryFile' is drawn on each page specified.

```
BOOL setStationery(BSTR stationeryFile,
       int startPage, int endPage, BOOL ontop)
```

Parameters

BSTR stationeryFile

The path of the file to be read in and used as stationery.

int startPage

The first page to have stationery applied.

int endPage

The last page to have stationery applied is endPage -1. If endPage is -1, then all pages will have stationery applied.

BOOL ontop

If TRUE, the stationery is drawn after the existing page has been drawn, otherwise it is drawn first.

Returns

TRUE if success, otherwise FALSE.

applyNup

Applies N-Up processing, where 2, 4 or 8 pages are shrunk and drawn on the one page.

```
BOOL applyNup(BSTR pageSize, int numPagesOnEachPage,
       int margin, int borderThickness)
```

Parameters

BSTR pageSize

A common page name, such as "A4" or "Letter". Valid values are: 4A, 2A, A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, 4B, 2B, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, SRA0, SRA1, SRA2, SRA3, SRA4, RA0, RA1, RA2, C0, C1, C2, C3, C4, C5, C6, C7/6, C7, DL, A3, extra, A4, extra, Letter, Legal, Executive, Ledger, Tabloid.

int numPagesOnEachPage

Either 1, 2, 4 or 8.

int margin

The size of the margin between pages, measured in points.

int borderThickness

The size of the border drawn around each page measured in points. -1 means no border, 0 means as thin as possible.

Returns

TRUE if success, otherwise FALSE.

getErrorMessage

Applies N-Up processing, where 2, 4 or 8 pages are shrunk and drawn on the one page.

```
BSTR getErrorMessage()
```

Returns

A description of what went wrong. This is set if any of the functions return FALSE.

saveWithSecurity

Saves a PDF file with PDF standard security and encryption.

```
BOOL saveWithSecurity(BSTR fileName, BSTR userPass, BSTR ownerPass,
        int flags, int keysizeInBits)
```

Parameters

BSTR fileName

The full path of the PDF file to be written to.

BSTR userPass

The password for opening the PDF file. May be empty.

BSTR ownerPass

The password for changing permissions in the PDF file. .

int flags

Flags as returned by *makeEncryptFlags128()* or *makeEncryptFlags40()*.

int keysizeInBits

The key size. The values of 40 or 128 are allowed.

Returns

TRUE if success, otherwise FALSE.

makeEncryptFlags40

Creates 40 bit encryption flags for use by saveWithSecurity().

```
int makeEncryptFlags40(BOOL print, BOOL change,
        BOOL copy, BOOL addnotes)
```

Parameters

BOOL print

If true, printing will be allowed.

BOOL change

If true, changing of the document will be allowed.

BOOL copy

If true, copying will be allowed.

BOOL addnotes

If true, annotations will be allowed.

Returns

The flags to be passed to saveWithSecurity().

makeEncryptFlags128

Creates 128 bit encryption flags for use by saveWithSecurity().

```
int makeEncryptFlags128(BOOL print, BOOL change,
        BOOL copyAndExtract, BOOL addnotes,
        BOOL formFillOrSign, BOOL extractAccessibility,
        BOOL assemble, BOOL hiResPrint)
```

Parameters

BOOL print

If true, printing will be allowed.

BOOL change

If true, changing of the document will be allowed.

BOOL copyAndExtract

If true, copying and extraction will be allowed.

BOOL addnotes

If true, annotations will be allowed.

BOOL formFillOrSign

If true, filling our of forms and document signing will be allowed.

BOOL extractAccessibility

If true, extraction for accessibility purposes is allowed.

BOOL assemble

If true, document assembly is allowed.

BOOL hiResPrint

If true, hi-res printing is allowed.

Returns

The flags to be passed to saveWithSecurity().

setDocInfo

Sets document information properties.

```
BOOL setDocInfo(BSTR name, BSTR value)
```

Parameters

BSTR name

The name of the value to be set. Allowed values are: Author, Title, Subject, Keywords.

BSTR value

If corresponding value.

Returns

TRUE if success.

addPage

Adds a blank new page to the current PDF document.

```
BOOL addPage(long width, long height)
```

Parameters

long width

The width of the page in points.

long height

The height of the page in points.

Returns

TRUE if success.

addSignature

Adds a digital signature to the PDF. Uses Windows certificate management. To see what certificates are installed, open up Control Panel, select "Internet Options", then "Content", then "Certificates".

```
BOOL addSignature(BOOL isVisibile, BSTR storeName,
        BSTR certName, BSTR issuerName, BSTR serialNum,
        BSTR keyBlob, BSTR location, BSTR reason, BSTR imageFile,
        LONG textDisplayFlags, LONG pageNum, BSTR anchorPosition,
        LONG x, LONG y, LONG width, LONG height)
```

Parameters

bool isVisible

If true the signature will be visible on the page, otherwise there will be no appearance, but the signature will be present in the signature tab of acrobat reader.

BSTR storeName

Optional. The name of the Windows certificate store. Defaults to "My".

BSTR certName

The common name of the certificate. If the certificate is issued to a person, it is usually the persons name.

BSTR issuerName

Optional. The name of the issuer. Used when there are multiple certificates with the same common name. The combination of issuerName and serialNum is unique.

BSTR serialNum

Optional. A hex encoded string that is the serial number of the certificate to use. Used when there are multiple certificates with the same common name. The combination of issuerName and serialNum is unique.

BSTR keyBlob

Optional. A hex encoded key used for signing. This is returned from exportKeysByCertificateName. It is used to avoid Windows showing the "accessing a private key" dialog box, which would be problematic for unattended applications/services.

exportKeysByCertificateName

Exports the keys associated with a certificate as a hex string. The value returned can be used in the addSignature method. The returned keys are not encrypted.
This method will cause Windows to display the "accessing a private key" dialog box, however when the result is passed to addSignature, no dialog will be shown.

```
BSTR exportKeysByCertificateName(BSTR certName)
```

Parameters

BSTR certName

The certificate name.

Returns

Unencrypted hexencoded key string

BSTR location

Optional. The location string in the signature. e.g. "City of Melbourne".

BSTR reason

Optional. The reason string in the signature. e.g. "I agree".

BSTR imageFile

Optional. The full path of an image file that will appear in the signature. JPEG, BMP and GIF files can be used.

long textDisplayFlags

A bitmask that controls what text appears on the signature appearance.
The following hex values may be OR'ed together.

0x01 - Certificate Name
0x02 - Location
0x04 - Reason
0x08 - Time/Date
0x20 - Big certificate Name

long pageNum

The page to place the signature on. 1 is the first page, -1 means the last page.

BSTR anchorPosition

Effects the position of the signature by controlling what the X and Y values are anchored to. Can be either: topleft, topright, bottomleft, bottomright, center, centerleft, centerright, centertop, centerbottom.

long X

Used to combination with "anchorPosition" to set the X coordinate. Units are in points.

long Y

Used to combination with "anchorPosition" to set the Y coordinate. Units are in points.

long width

The signature width. Units are in points.

long height

The signature height. Units are in points.

Returns

TRUE if success.

exportKeysByCertificateName

Exports the keys associated with a certificate as a hex string. The value returned can be used in the addSignature method. The returned keys are not encrypted.
This method will cause Windows to display the "accessing a private key" dialog box, however when the result is passed to addSignature, no dialog will be shown.

```
BSTR exportKeysByCertificateName(BSTR certName)
```

Parameters

BSTR certName

The certificate name.

Returns

Unencrypted hexencoded key string to be used in the "keyBlob" parameter of addSignature.

createSelfSignedCertificate

Creates a self signed certificate. The certificate is created in the Windows Certificate "My" store.

```
BOOL createSelfSignedCertificate(BSTR certName,
               BSTR orgUnit, BSTR orgName,
               BSTR email, BSTR localeName,
               BSTR stateName, BSTR countryCode)
```

Parameters

BSTR certName

The certificate name.

BSTR orgUnit

Optional. The organizational unit.

BSTR orgName

Optional. The organizational name.

BSTR email

Optional. The email address.

BSTR localeName

Optional. The locale.

BSTR stateName

Optional. The state.

BSTR countryCode

Optional. The 2 char country code.

Returns

TRUE if success.