Intégration d'une imprimante PDF pour éditions automatisées avec GhostScript et Redmon.

Par Farscape

Le but de cet article est de vous montrer comment générer de manière automatisée dans un programme (**MFC** par exemple) une édition au format **PDF.**
Pour cela nous utiliserons une Imprimante PDF que nous allons assembler de toutes pièces.

1. A qui s'adresse cet article :

- A priori à tous les utilisateurs Windows désirant une solution simple et gratuite de génération d'édition au format PDF.
- A tous les programmeurs Win32 : C/C++ etc .. qui voudraient tirer parti de cette imprimante dans leur programme.

La première partie de l'article est consacrée à l'installation et la configuration des éléments logiciels :
- Installation de l'imprimante PDF.
- Réglage du pilote d'impression.
- Paramétrages du redirecteur de port.

Ensuite je vous présenterai les modifications que j'ai apportées sur le redirecteur de port **RedMon** Afin de contrôler au mieux le nom du fichier PDF de sortie, et la possibilité d'appeler un programme à la fin de la génération du fichier PDF.

On trouve sur le net différentes solutions payantes ou gratuites (PdfCreator, cutePdf etc..) pour traiter ce sujet.
Mais elles ne permettent pas toujours de contrôler le nom du document en sortie, ou impose à l'utilisateur de composer ce nom.
Ce dernier point est rédhibitoire pour automatiser un traitement d'impression PDF dans un logiciel.

A travers cet article je vais donc vous montrer comment utiliser le logiciel open source **GhostScript** pour parvenir à nos fins …

2. Présentation de la solution :

GhostScript est un logiciel permettant la conversion de fichiers au format PostScript vers le format PDF (entre autre).
Cette utilitaire fonctionne en ligne de commande dans une console, ce qui n'est pas pratique lorsque l'on veut automatiser le processus d'impression.
En outre **GhostScript** fournit un certain nombre de batch pour réaliser des taches de conversions en ligne de commande.

Redmon est un logiciel de redirection de port d'impression, il récupère ce que va générer l'imprimante pour le rediriger sur **GhostScript.**

Le processus de génération du document PDF sera le suivant :
- L'utilisateur imprime en sélectionnant l'imprimante virtuelle PostScript
- Le flux d'impression est transformé en fichier PostScript
- **Redmon** intercepte ce fichier et l'envoi à **GhostScript** pour le convertir au format PDF

3. Installation et configuration des logiciels

Les logiciels:

Il nous faudra donc installer les logiciels **Ghostscript et Redmon**
Que vous pouvez télécharger aux adresses suivantes :

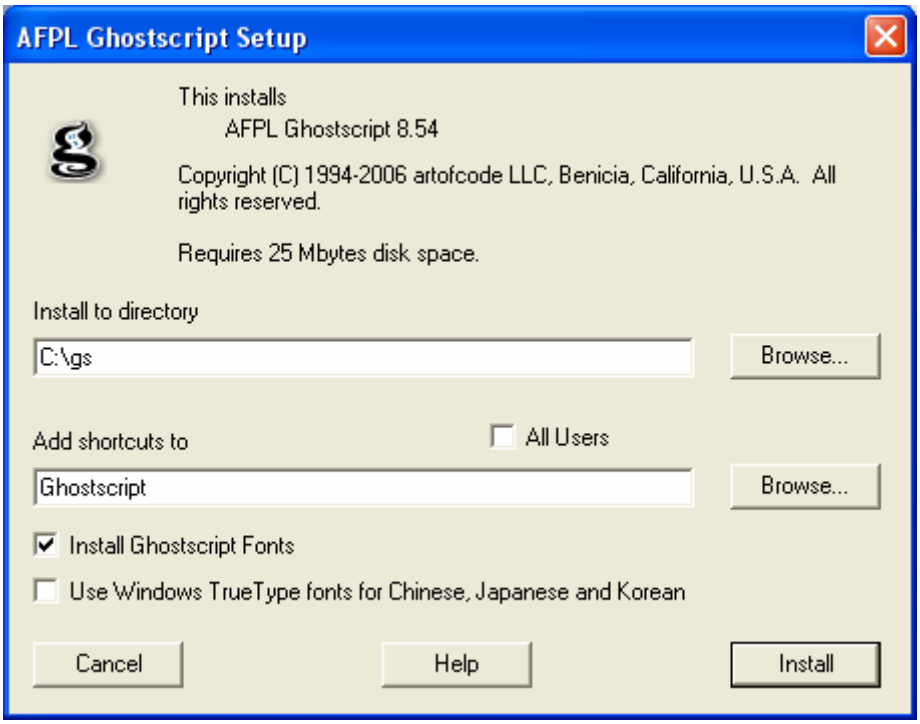
Ghostscript :
http://www.cs.wisc.edu/~ghost/doc/AFPL/gs854.htm

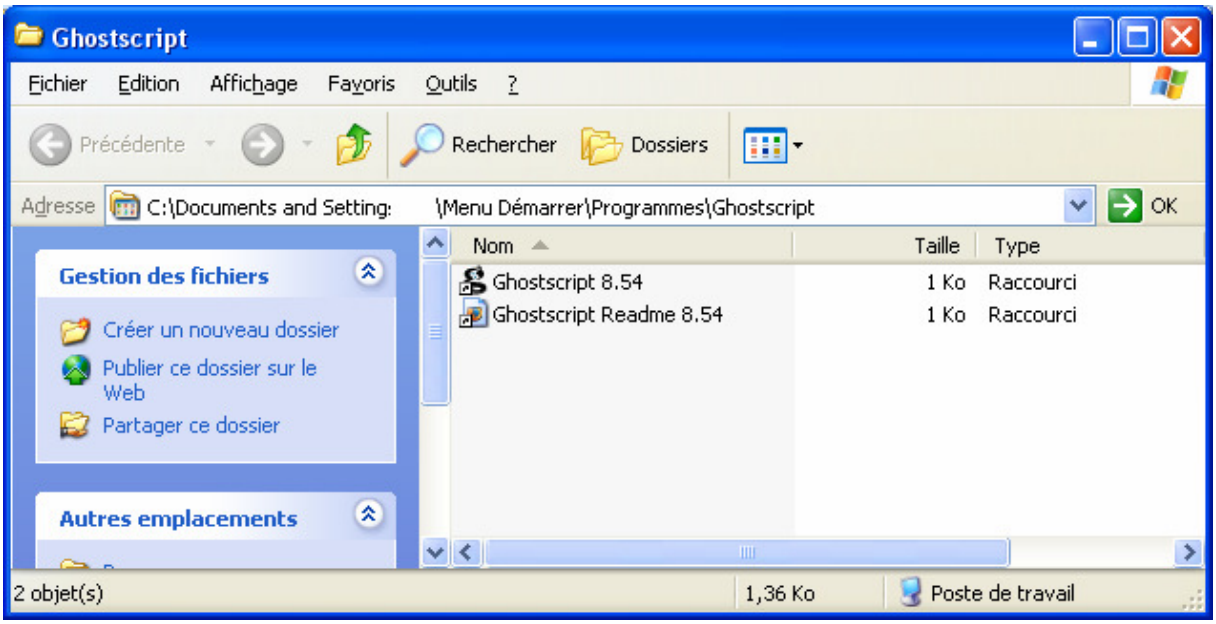
Redmon :
http://www.cs.wisc.edu/~ghost/redmon/index.htm
ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/redmon17.zip

Il existe une version française de **Redmon**
ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/redmon17fr.zip
C'est la version que j'utiliserai par la suite.

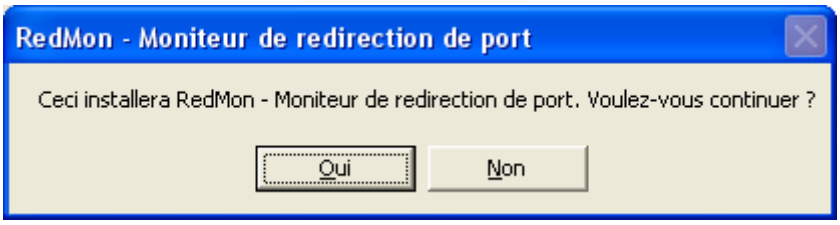
Procédez à l'installation du logiciel **Ghostscript** en lançant son setup.
Il n'y a pas de consignes particulières, sauf pour le répertoire d'installation :

Je vous conseille de remplacer **C:\Program Files\GS** par **C:\GS** pour éviter les problèmes de chemin par la suite…





Procédez ensuite à l'installation de **Redmon** .



C'est tout !

4. Installation de l'imprimante virtuelle sous Windows XP

Il nous reste à installer notre imprimante postscript virtuelle.
La distribution **GhostScript** distribue un fichier **C:\gs\gs8.54\lib\ghostpdf.inf** pour le driver postscript,
Mais ce pilote n'est pas signé.

Autres solutions possibles :

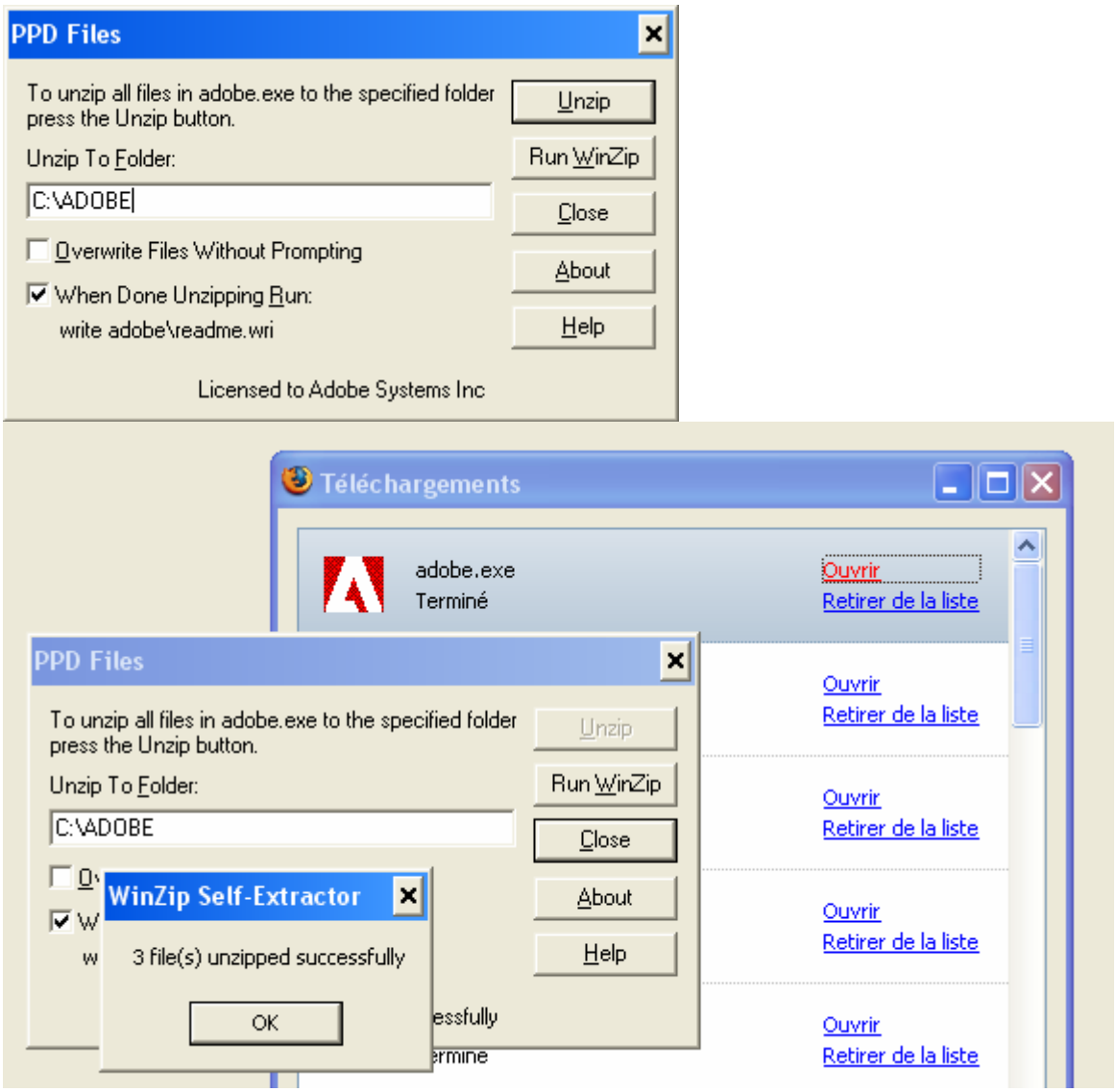
Utiliser une imprimante **HP Laserjet 4000 Series PS**
Ou encore utiliser l'imprimante postscript **AdobePS**
C'est cette dernière solution que j'ai retenue.

Télécharger les éléments suivants sur le site d'Adobe :

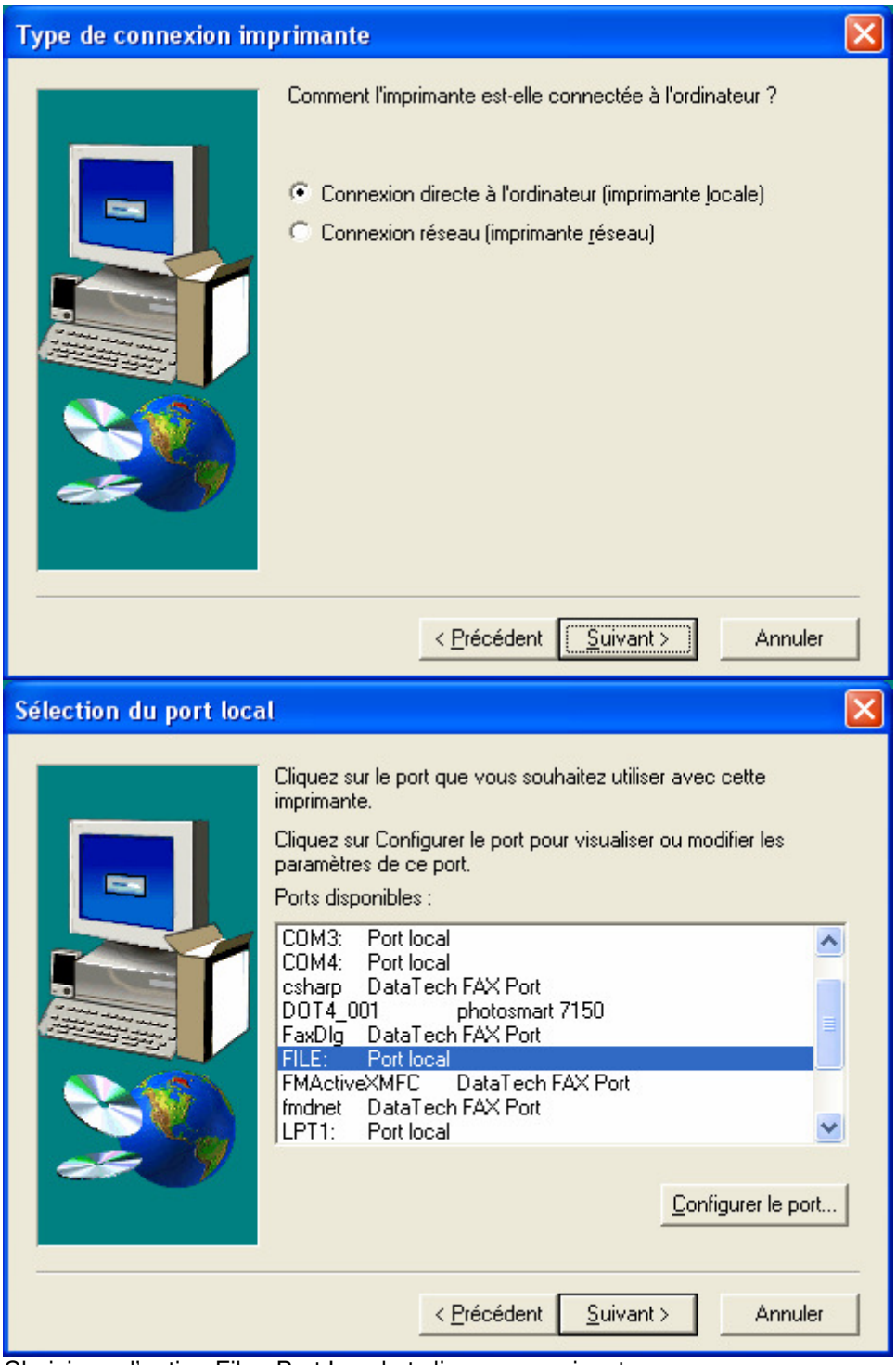
Adobe acrobat postcript :
http://www.adobe.com/support/downloads/thankyou.jsp?ftpID=1503&fileID=1441

Adobe fichier PPD acrobat distiller:
http://download.adobe.com/pub/adobe/printerdrivers/win/all/ppdfiles/adobe.exe

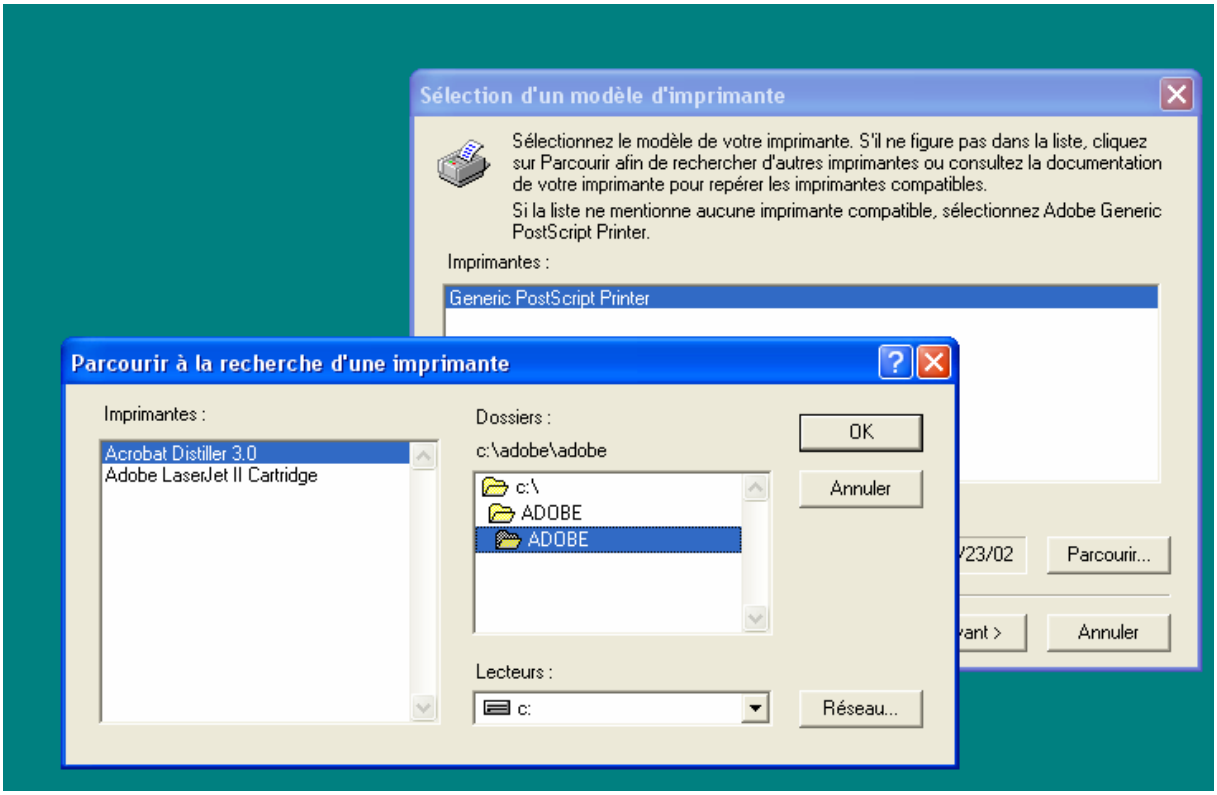
Exécutez en premier **Acrobat Distiller** en plaçant les fichiers (par exemple) dans le répertoire **C:\ADOBE**



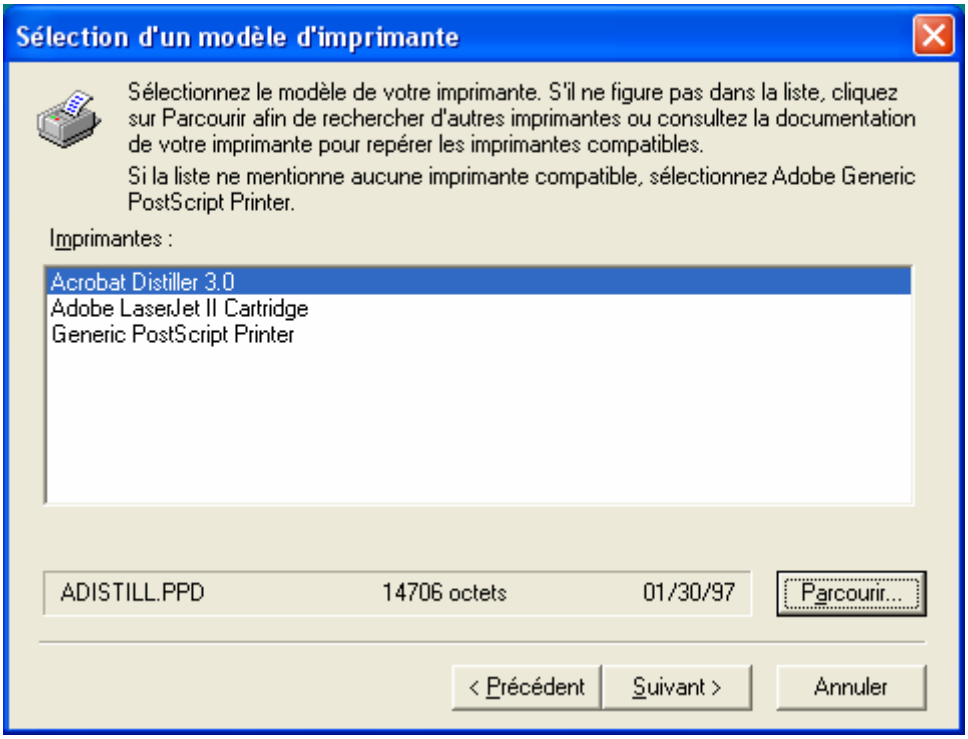
Lancez ensuite le setup d'**Adobe Acrobat PostScript**.



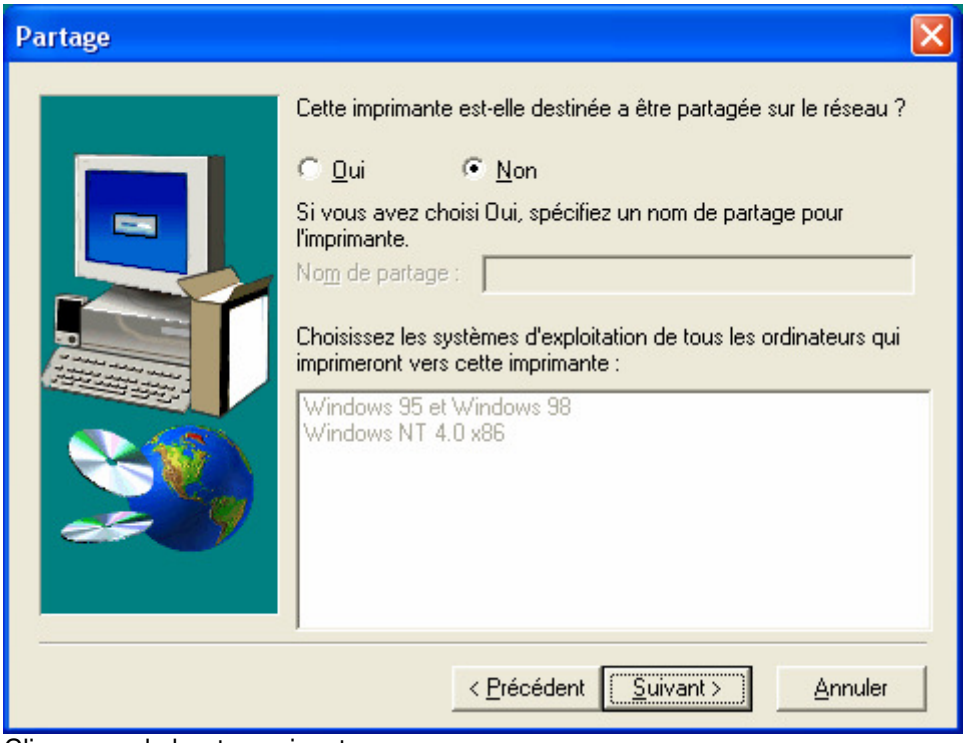
Choisissez l'option File : Port Local et cliquez sur suivant.



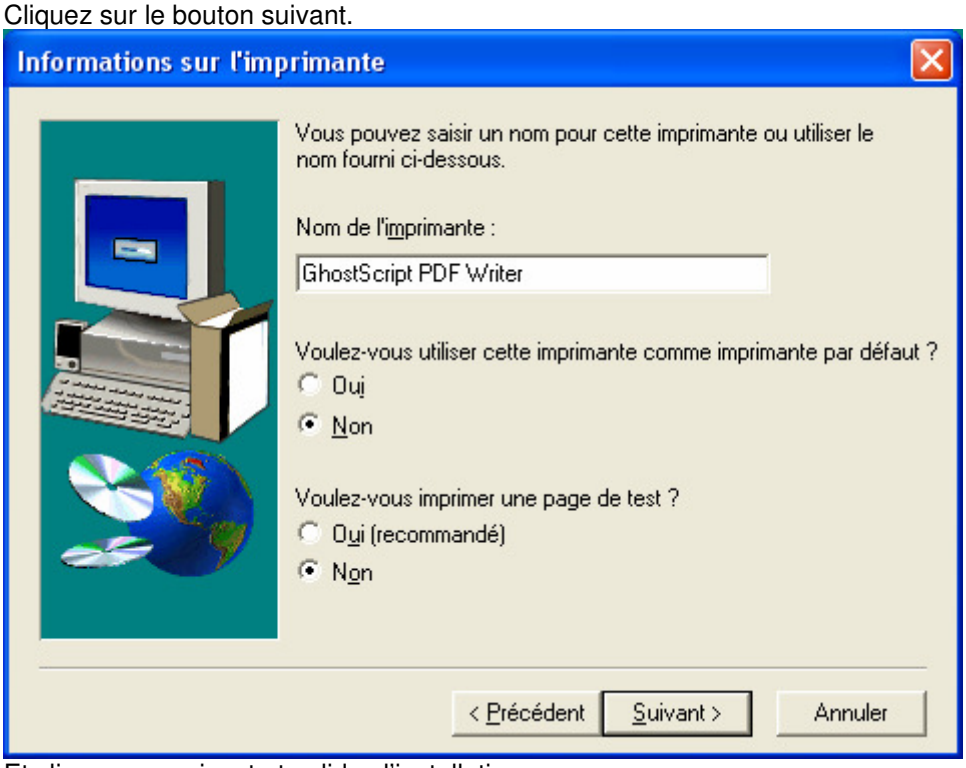
Sur la fenêtre sélection d'un modèle d'imprimante cliquez sur le bouton parcourir et sélectionnez l'option Acrobat Distiller 3.0.



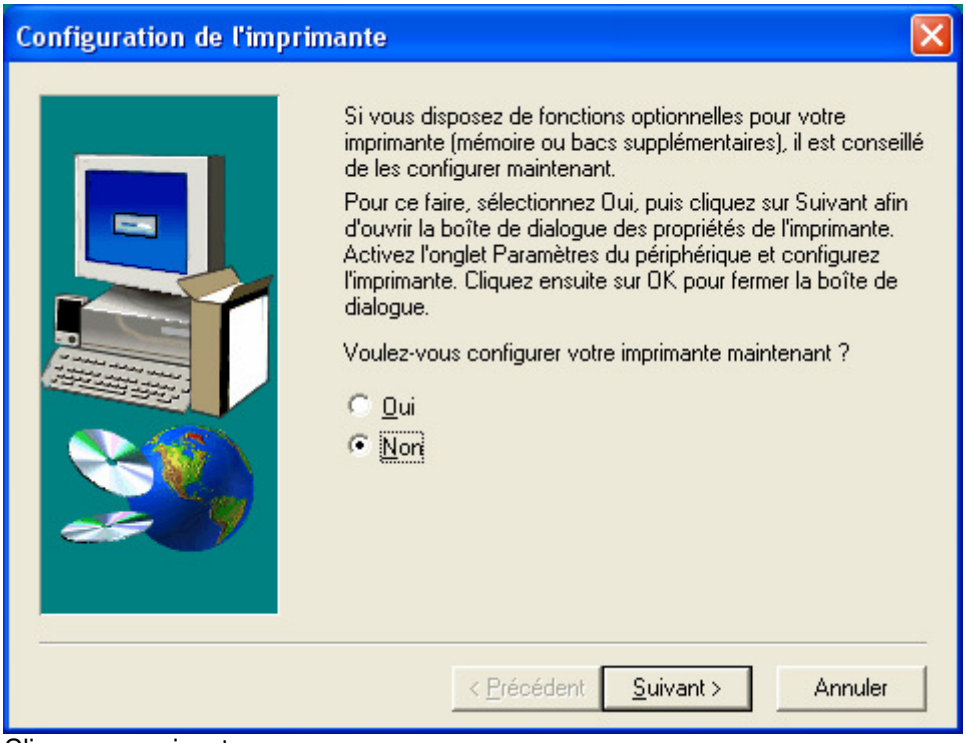
Cliquez ensuite sur suivant.



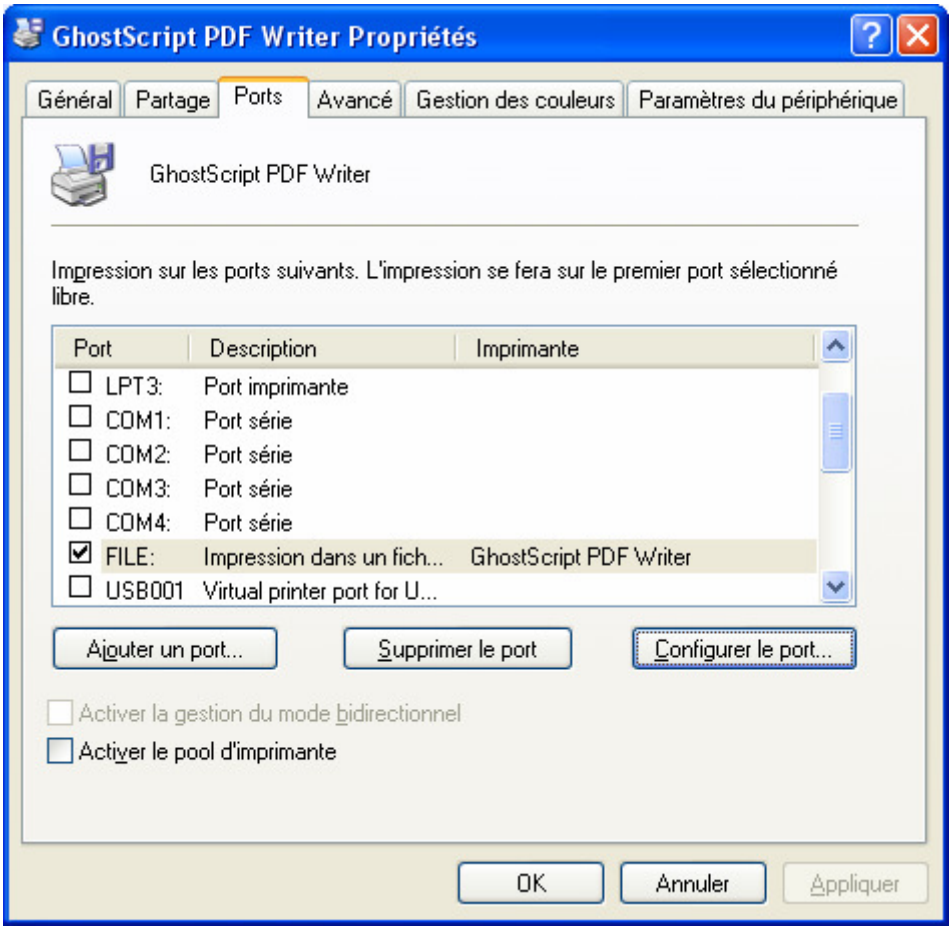
Cliquez sur le bouton suivant.



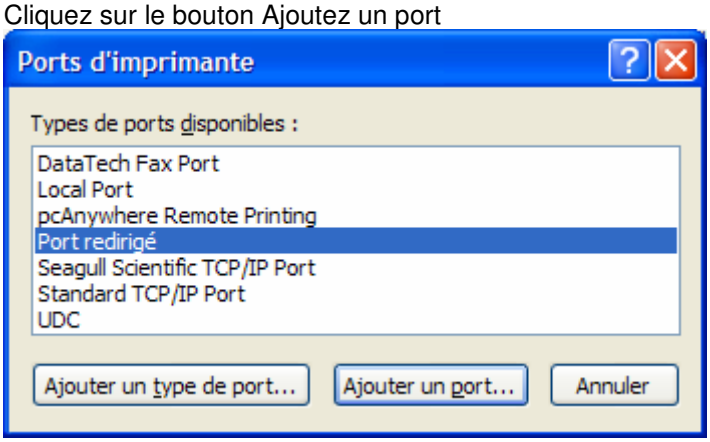
Et cliquez sur suivant et validez l'installation.



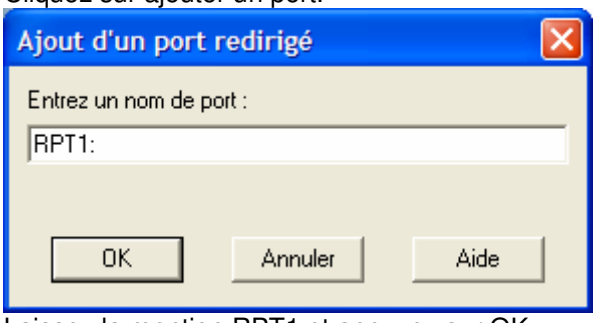
Cliquez sur suivant.
Dans le panneau des imprimantes cliquez sur clic droit propriétés sur la nouvelle imprimante.
Sélectionnez l'onglet ports :



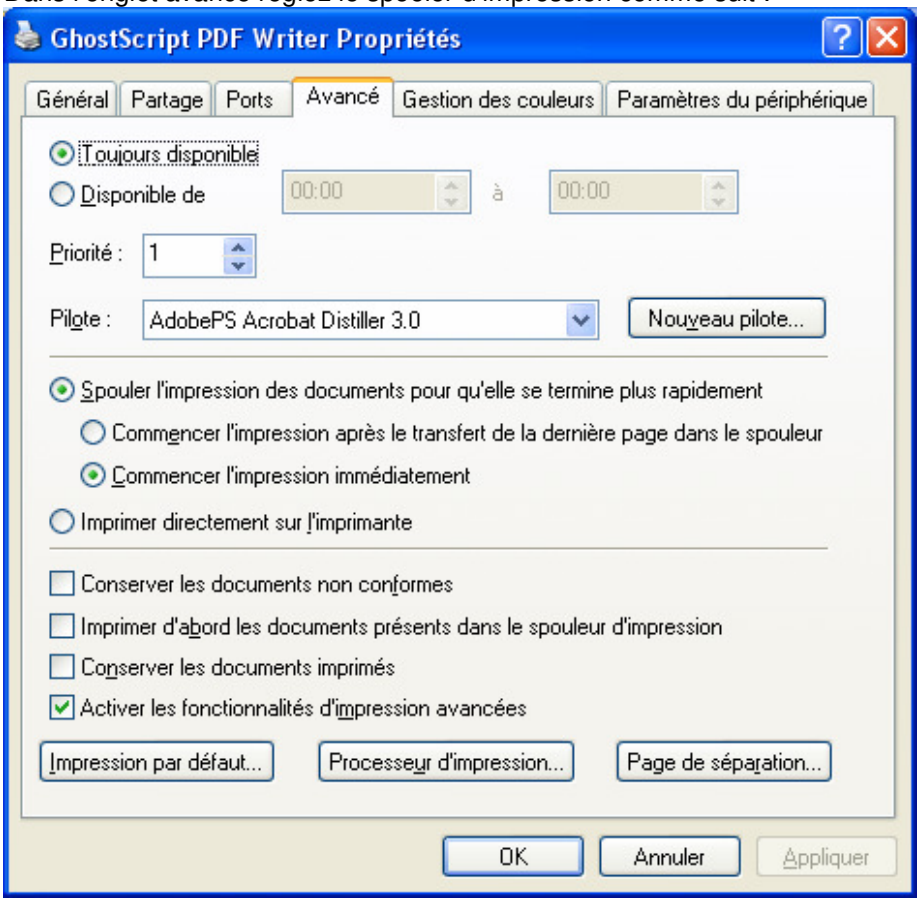
Cliquez sur le bouton Ajoutez un port



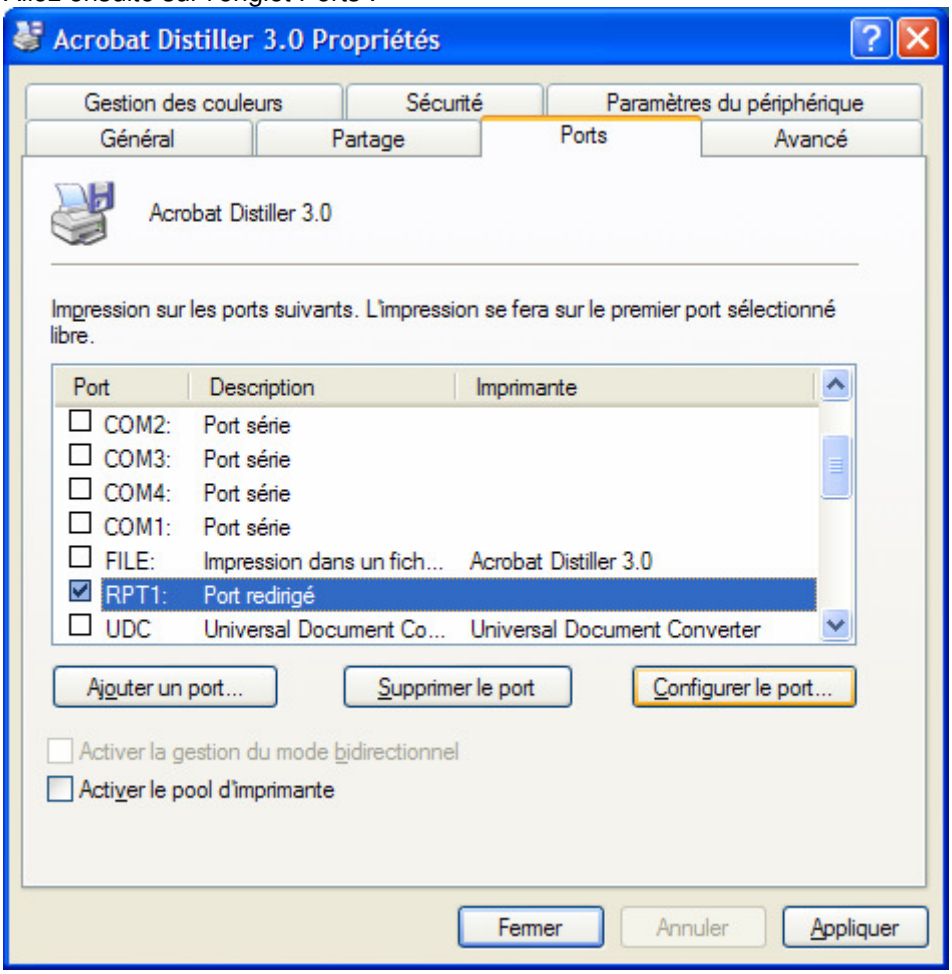
Cliquez sur ajouter un port.



Laissez la mention RPT1 et appuyez sur OK
Dans l'onglet avancé réglez le spooler d'impression comme suit :



Allez ensuite sur l'onglet Ports :



Avant de configurer le port nous allons créer à l'aide de Notepad un fichier **pdfwriter.rsp**
Dont le contenu est le suivant :

-IC:\gs\gs8.54\lib;C:\gs\fonts
-sDEVICE=pdfwrite
-r300
-dPDFSETTINGS=/prepress
-dNOPAUSE
-dSAFER
-dAutoRotatePages=/PageByPage

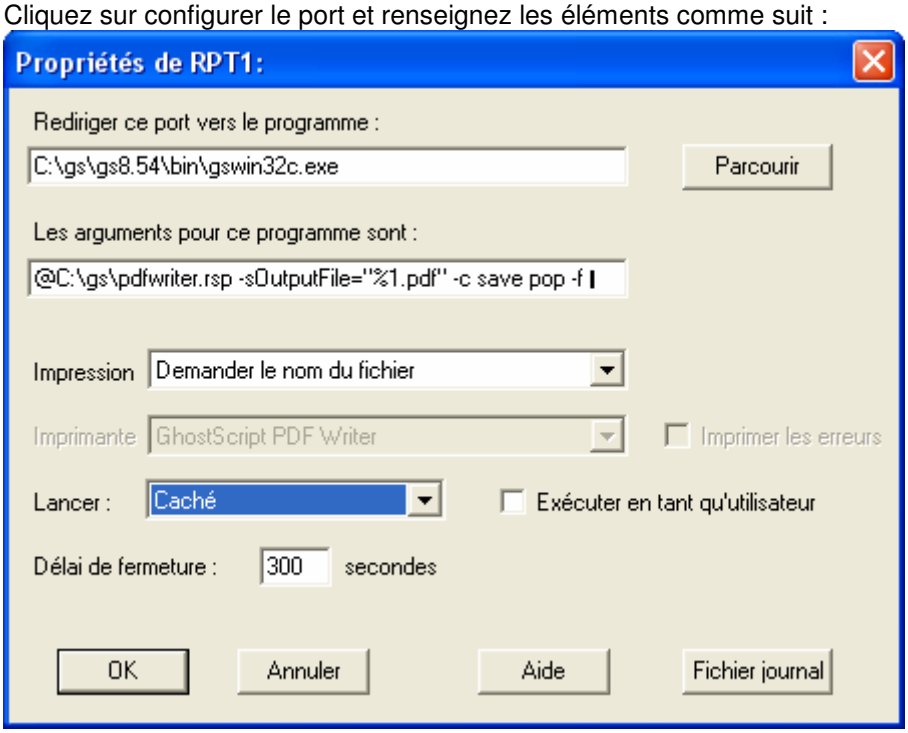
Explications :

-IC:\gs\gs8.54\lib;C:\gs\fonts :
Permet de spécifier les chemins des différents modèles et fontes employées.
-sDEVICE=pdfwrite :
Permet de sélectionner le périphérique de sortie **pdfwrite** correspond au format pdf, d'autres formats sont disponibles comme le format **jpeg** ou **tiff** .
Pour plus d'information consultez la rubrique **C:/gs/gs8.54/doc/Devices.htm**
-r300 :
Sélection de la résolution ici 300 dpi .
-dPDFSETTINGS:
Permet de fixer le niveau de compatibilité pdf **,/prepress** sélectionne une sortie similaire a l'option « prepress Optimized » de Acrobat Distiller.
-dNOPAUSE :
Permet d'éviter les attentes clavier de **Ghostscript**
-dSAFER :
Paramètre souvent conseillé pour cette utilisation, il verrouille sa configuration PostScript prévenant tous changements sur le périphérique de sortie.
-dAutoRotatePages :
Permet la rotation automatique de page si nécessaire..

Les explications sont disponibles dans la documentation : **C:\gs\gs8.54\doc\Use.htm**

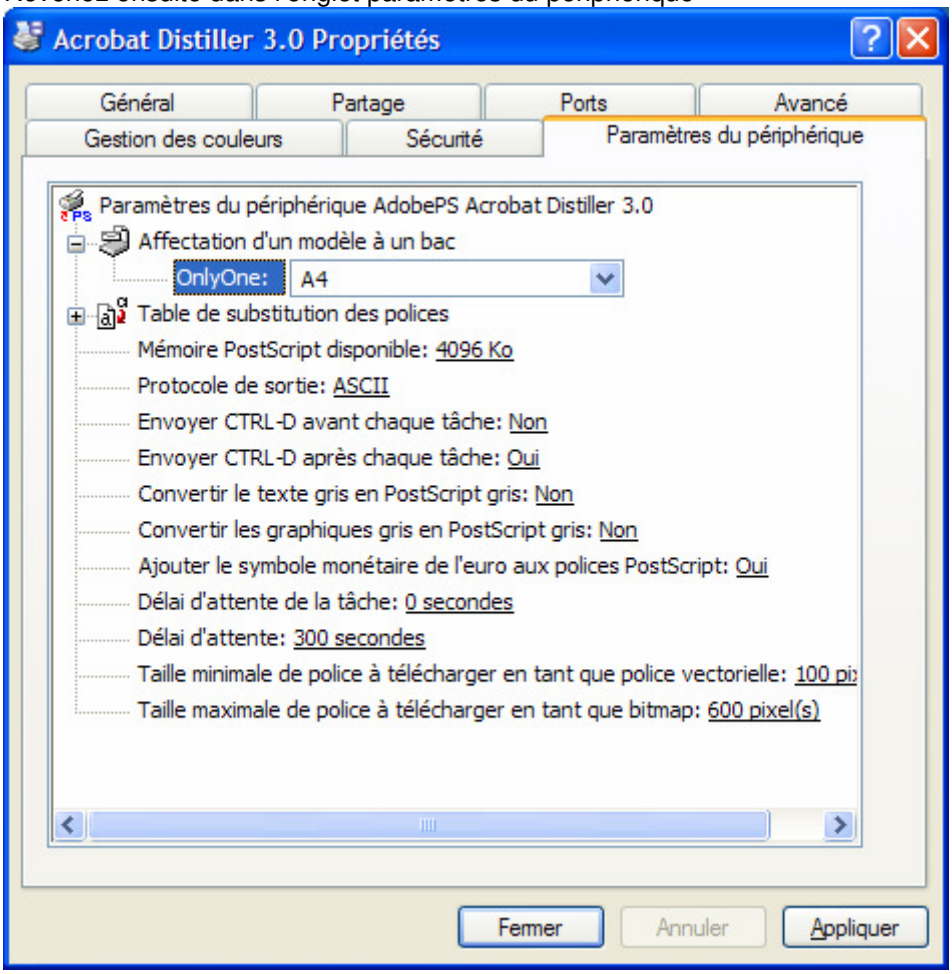
Vérifiez bien après l'enregistrement que le fichier porte le nom **PdfWriter.rsp**, et placez le dans le répertoire **C:\GS**

Cliquez sur configurer le port et renseignez les éléments comme suit :



La ligne pour la redirection :C:\gs\gs8.54\bin\gswin32c.exe
L'argument du programme : @C:\gs\pdfwriter.rsp -sOutputFile="%1.pdf" -c save pop -f -

Revenez ensuite dans l'onglet paramètres du périphérique



Vérifiez que la ligne Envoyer CTRL-D après chaque tâche soit à OUI.

Test de l'imprimante :

Voila l'installation de l'imprimante est terminée.
Il reste plus qu'à la tester.
Pour cela un document Word fera très bien l'affaire.

Imprimez votre document avec la nouvelle imprimante.
Une boîte de dialogue apparaît pour saisir le nom du document de sortie
Saisissez le nom sans l'extension de fichier.
Vous devez retrouver le fichier PDF doit être a l'emplacement choisi.

Si le fichier n'est pas généré :

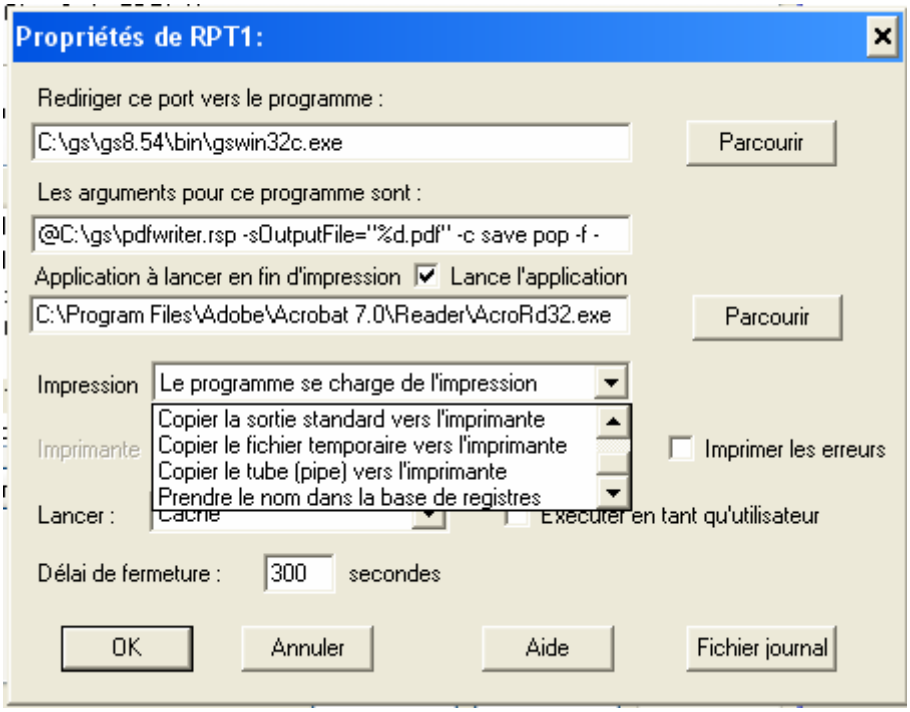
Vérifier les différents chemins pour **GhostScrip**.
Les chemins inscrits dans le fichier **pdfWriter.rsp**.
Le chemin d'accès du fichier **pdfWriter.rsp** et la ligne de commande dans la configuration du port.

5. Conclusions sur l'installation

Voila pour l'installation manuelle des différents éléments logiciels.
La conversion PDF fonctionne bien, j'ai fait le test sur différents documents Word et sur des tableaux Excel compliqués.
Contrairement à d'autres produits gratuits que j'ai pu tester, le résultat est impeccable.

6. Modifications de Redmon

J'ai apporté quelques modifications à **Redmon**,
En voici un aperçu graphique sur la fenêtre de paramétrage du port :



J'ai rajouté les modes de fonctionnement suivants :

Mode d'impression :
Prendre le nom dans la base de registres :
Dans ce mode le programme appelant place le nom du fichier de sortie dans la base de registre à l'emplacement suivant :
HKEY_CURRENT_USER\Software\Ghostgum\RedMon\FileName
Le nom est récupéré et la zone est vidée.
Si un autre programme comme Word demande une édition dans ce mode, le pilote rebascule automatiquement dans le mode **Demander le nom du fichier** si cette zone est vide.
Pour l'exemple j'ai paramétré le lancement d'**Acrobat Reader.**

La deuxième modification consiste à préciser un programme qui devra être lancé à la fin du processus d'impression et qui recevra en argument le nom du fichier PDF.

Ce mode est actif avec les modes d'impressions suivants :

- **Demander le nom du fichier**
- **Prendre le nom dans la base de registres**
- **Le programme se charge de l'impression.**

Enfin une option intéressante que j'ai découverte en étudiant les sources :

La possibilité d'utiliser le nom du job d'impression pour générer le fichier au format PDF.

Il suffit de remplacer **–sOutputFile= « %1.pdf »** par **« %d.pdf »**
Le 'd' signifiant document.

Dans le cas ou le mode d'impression sélectionné est **Prendre le nom dans la base de registres**
La priorité du nom généré sera celui contenu dans la base de registre si elle existe.

Autre déviation de fonctionnement, si **%d** est sélectionné et que le nom de la base de registre est vide, le mode de d'impression basculera sur **Le programme se charge de l'impression.**
Ce fonctionnement panaché permettra d'utiliser l'imprimante de deux manières différentes avec un seul réglage.
Voila pour les modifications de comportements.

Concernant la programmation :

La distribution de **Redmon** fournit un zip contenant les sources.
J'ai un peu modifié le .mak existant pour pouvoir compiler avec **Visual 6.0**
La version courante de **Redmon** étant la 1.7 j'ai annoté mes modifications et ajouts de code avec la mention **//1.7a**.
Dans un premier temps je n'ai apporté que des modifications sur la version française des ressources.
J'ai testé mes modifications uniquement dans un environnement **Windows XP**.

Mise à jour de la dll :

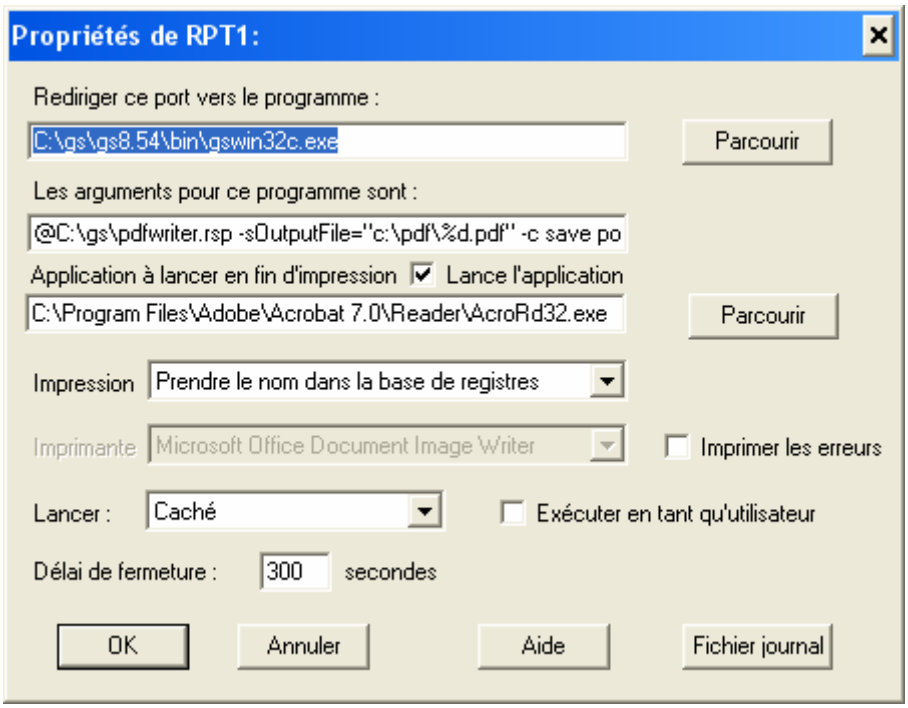
Pour mettre à jour la dll de **Redmon** il suffit de stopper le spooler d'impression avec la commande suivante :
Net stop spooler
De copier la nouvelle DLL **redmonntn.dll** dans le répertoire C:\Windows\System32
Et de taper la commande **Net start spooler** pour relancer le spooler.

L'ensemble des ressources nécessaires est disponible sur mon site à l'emplacement suivant :

http://farscape.developpez.com/Samples/GhostPdfWriter/

8. Utilisation de la solution :

Pour illustrer la solution j'ai écrit un petit programme de test mettant en œuvre les deux méthodes pour fixer le nom du fichier PDF en sortie.
Pour cet essai le port de l'imprimante est paramétré comme suit :



Les fichiers générés seront placés dans le répertoire **C:\PDF** qu'il vous faudra créer.
L'option par défaut pour le nom sera le nom du document édité grâce à l'option « %d »
Enfin le mode d'impression est réglé sur prendre le nom dans la base de registres.

L'interface du programme de test :

Ce petit programme quelque soit la méthode utilisée vous demandera de choisir un fichier .txt ou autre en ASCII et l'éditera cinq fois dans une boucle.
La sélection de l'imprimante est transparente pour l'utilisateur.
Le lancement du programme associé en fin d'édition est désactivé en début de traitement et réactivé à la fin.
A la fin du traitement les cinq fichiers sont dans le répertoire **C:\PDF**
La ListBox permet de voir l'avancement des jobs d'impression.

En dehors du traitement d'édition le programme implémente deux classes :
Une permettant de lister les jobs d'impressions pour une imprimante donnée.
L'autre permettant de gérer les options de **RedMon** dans la base de registres.

Détails :

```
namespace Printer
{
struct JOB_INFO
	{
	   JOB_INFO():nJobId(-
1),nStatus(0),nPosition(0),nTotalPages(0),nPagesPrinted(0){}
	   int     nJobId;
	   CString strPrinterName;
	   CString strMachineName;
	   CString strUserName;
	   CString strDocumentName;
	   int    nStatus;
	   int    nPosition;
	   int     nTotalPages;
	   int     nPagesPrinted;
	   CString strDocumentDate;
	   CString strDocuementTime;
	};

	typedef CArray<JOB_INFO ,JOB_INFO > CJobArray;

	class CJob
	{
	public:
		CJob():m_hPrinter(INVALID_HANDLE_VALUE){};

		~CJob();

		void   ClosePrinter();// fermeture du handle de l'imprimante.
		// fixe le nom de l'imprimante de travail
		void   SetPrinterName(const char
*szPrinterName){m_strPrinterName=szPrinterName;}
		// renvoie le nom de l'imprimante de travail.
		CString GetPrinterName()const {return m_strPrinterName;}

		// recherche d'un job dans le spooler d'impression.
		BOOL         FindJob(const char *szjobname);

		// recherche d'un job dans le spooler d'impression par position.
		BOOL         NextJob(int &nlocation) ;

		// recuperation des jobs en cours.
		BOOL         GetAllJob(CJobArray &rJobArray) ;

		// renvoie la structure JOB_INFO ,initialisée avec les fonctions
FindJob,NextJob.
		JOB_INFO     GetJobInfos() const { return m_JobInfo;}
	private:
		JOB_INFO m_JobInfo;
		CString m_strPrinterName;
		HANDLE  m_hPrinter;
	};
}
```

La classe de Gestion des données pour **Redmon** :

```
namespace DvpPdf
{
	class SetOptions
	{
		public:
		SetOptions(){};

		// selection de l'imprimante PDF dont le port est égal a szPort
		// le retour est NULL si echec de creation du DC.
		CDC * SelectPrinter(const char *szPort="RPT1:");

		// renvoie le nom de l'imprimante en cours d'utilisation
		// fixée par SelectPrinter.
		CString GetPrinterName() const {return m_strPrinterName;}

	// Active/desactive le lancement du programme à la fin du document.
		void  SetPdfAppRun(BOOL bEnable=FALSE);

		// fixe le nom pour la génération du fichier.
		void  SetPdfFileName(const char *szFilename);

		// renvoie le dernier nom fixé par SetPdfFileName
		CString GetLastPdfFileName() const {return m_strLastPdfName;}

		private:
		CString m_strPrinterName;
		CString m_strLastPdfName;
	};
};
```

Implications sur la méthode utilisée :

Quand le nom du fichier correspond au nom du document la seule contrainte que l'on aura au niveau du traitement c'est d'attendre que tous les jobs soient terminés, pour réaliser ensuite un traitement d'envoi par E-mail par exemple.

Extrait du traitement :

```
void CTestPDFDlg::PrintPdfByDocumentName()
{
	CFileDialog dlg( TRUE, _T( "txt" ), _T( "*.txt" ) );
	if( dlg.DoModal() != IDOK ) return;

	CStdioFile File;
	if( !File.Open(dlg.GetPathName(), CFile::modeNoTruncate    | CFile::modeRead
| CFile::typeText ) )
	{
		#ifdef _DEBUG
		afxDump << "erreur d'ouverture fichier" << "\n";
		#endif
		return;
	}
	CString strTitle;

	DvpPdf::SetOptions OptionsPDF;
// pas de lancement de l'application sur l'edition du document.
	OptionsPDF.SetPdfAppRun();
// 5 fois l'edition avec un nom de document different.
	for(int nDoc=0;nDoc<5;nDoc++)
	{
		strTitle.Format("%s-%d",static_cast<const char
*>(dlg.GetFileTitle()),nDoc+1);
		CDC *pDC=OptionsPDF.SelectPrinter();// selection imprimante PDF
		if(!pDC)
		{
			AfxMessageBox("Probleme de sélection d'imprimante");
			return;
		}
		DOCINFO docInfo;
		memset(&docInfo, 0, sizeof(DOCINFO));
		docInfo.cbSize = sizeof(DOCINFO);
		docInfo.lpszDocName = strTitle;
		docInfo.lpszOutput = NULL;
		PrintPdf(pDC,File,docInfo);
	}
	// attente de fin de traitement de l'imprimante
	Printer::CJob job;
	Printer::CJobArray arJobInfos;
	job.SetPrinterName(OptionsPDF.GetPrinterName());
	CString str;
	do
	{
		job.GetAllJob(arJobInfos);// recupere les jobs en cours
		CTestPDFApp ::PumpMessages();// pompe a messages voir faq
		m_ListBoxJobs.ResetContent();
		CTestPDFApp ::PumpMessages();
		for(int i=0;i<arJobInfos.GetSize();i++) // affichage des jobs
		{
			if(m_bCancel) break;
			str.Format("%d:%d/%d Pages - Nom:
",arJobInfos[i].nJobId,arJobInfos[i].nPagesPrinted,arJobInfos[i].nPagesPrinted);
			str+=arJobInfos[i].strDocumentName;
			m_ListBoxJobs.AddString(str);
			CTestPDFApp ::PumpMessages();
		}
		if(m_bCancel) break;
		_sleep(1000);
	}
	while(arJobInfos.GetSize());
	m_ListBoxJobs.AddString("Traitement Terminé");
	OptionsPDF.SetPdfAppRun(TRUE); // retablit le lancement de l'application
apres l"edition d'un document.

}
```

Quand c'est le programme qui impose le nom du fichier dans la base de registre, le décalage de traitement entre le fait de placer le nom dans la base et la lecture de celui-ci par **Redmon** va causer un problème de synchronisation dû au temps de traitement du spooler de Windows.

On risque en effet de placer un nouveau nom dans la base alors que le précédent n'a pas été encore lu par **Redmon**.

Pour gérer ce problème il faudra intercaler un traitement d'attente sur le job d'impression entre chaque soumission d'édition.

Extrait du Traitement :

```
void CTestPDFDlg::PrintPdfByName()
{
	// TODO: Add your control notification handler code here
	CFileDialog dlg( TRUE, _T( "txt" ), _T( "*.txt" ) );
	if( dlg.DoModal() != IDOK ) return;
	CStdioFile File;
	if( !File.Open(dlg.GetPathName(), CFile::modeNoTruncate| CFile::modeRead |
CFile::typeText ) )
	{
		#ifdef _DEBUG
		afxDump << "erreur d'ouverture fichier" << "\n";
		#endif
		return;
	}
	CString strTitle;

	DvpPdf::SetOptions OptionsPDF;
// pas de lancement de l'application sur l'edition du document.
	OptionsPDF.SetPdfAppRun();
	Printer::CJob job;
	Printer::CJobArray arJobInfos;
	CString str;
// 5 fois l'edition avec un nom de fichier different.
	for(int nDoc=0;nDoc<5;nDoc++)
	{
		strTitle=dlg.GetFileTitle();	// le nom du fichier uniquement
		CDC *pDC=OptionsPDF.SelectPrinter();// selection imprimante PDF
		job.SetPrinterName(OptionsPDF.GetPrinterName());

		if(!pDC)
		{
			AfxMessageBox("Probleme de sélection d'imprimante");
			return;
		}
		DOCINFO docInfo;
		memset(&docInfo, 0, sizeof(DOCINFO));
		docInfo.cbSize = sizeof(DOCINFO);
		docInfo.lpszDocName = strTitle;
		docInfo.lpszOutput = NULL;

		// le nom du fichier généré .
		str.Format("%s-%d",static_cast<const char
*>(dlg.GetFileTitle()),nDoc+1);
		OptionsPDF.SetPdfFileName(str);

		PrintPdf(pDC,File,docInfo);
	// attente de fin de traitement de l'imprimante	pour le job en cours..
		while(job.FindJob(strTitle))
		{
			CTestPDFApp ::PumpMessages();// pompe a messages voir faq:
			m_ListBoxJobs.ResetContent();
			CTestPDFApp ::PumpMessages();

			if(m_bCancel) break;
```

```
                    str.Format("%d:%d/%d Pages - Doc: ",
                           job.GetJobInfos().nJobId,
                           job.GetJobInfos().nPagesPrinted,
                           job.GetJobInfos().nPagesPrinted);

                    str+=job.GetJobInfos().strDocumentName;
                    str+=" Fichier: ";
                    str+=OptionsPDF.GetLastPdfFileName();

                    m_ListBoxJobs.AddString(str);
                    CTestPDFApp ::PumpMessages();
                    if(m_bCancel) break;
                    _sleep(1000);
                }
        }
        m_ListBoxJobs.AddString("Traitement Terminé");
// retablit le lancement de l'application apres l"edition d'un document.
        OptionsPDF.SetPdfAppRun(TRUE); }
```

9. Conclusions

Je pense avoir complété le fonctionnement de **Redmon** pour disposer de suffisamment de souplesse à l'utilisation.
La possibilité de lancer un programme à la fin de l'édition offre des perspectives intéressantes comme l'envoi par email avec un programme dédié, un peu comme **PdfCreator**.
Le fait de maîtriser le nom de sortie permettra au programmeur de maitriser l'utilisation du fichier après l'édition :
Comme par exemple l'envoi par mail automatisé de facture etc..

A vous de jouer …

Remerciements :

Je remercie toute l'équipe du forum **Visual C++** pour sa relecture attentive du document .
Je remercie également **nico-pyright(c)** pour la correction orthographique.