Acquérir une image à partir d'un scanner avec la bibliothèque de fonctions WIAAUT.DLL avec Delphi

Table des matières

I Introduction

Dans ce tutoriel, vous allez apprendre à utiliser la bibliothèque de fonctions WIAAUT.DLL version 2.0 en utilisant Delphi. Je vais vous montrer comment acquérir une image en utilisant un scanner (imprimante HP PhotoSmart 6510). Nous allons détailler avec le plus de précision possible les objets Device et Image de même que nous allons utiliser les principales boîtes de dialogue communes mises à notre disposition par cette bibliothèque. Je dois vous dire que cet exemple a été écrit avec la version XE Starter de Delphi, mais il est possible d'appliquer ces quelques lignes de code dans n'importe quelle version même antérieure.

II création du projet

Nous devons dans un premier temps créer un nouveau projet. Dans votre éditeur créer un nouveau projet que vous appellerez comme vous le désirez. Nous allons devoir importer bibliothèques de type. Pour ce faire il suffit de sélectionner la commande de menu « importer un composant ». Et de sélectionner ensuite l'option « importer une bibliothèque de type ». Dans la liste de bibliothèques de type qui vous est présentée, sélectionnez celle qui qui porte le nom : « Microsoft Windows image Acquisition Library v2.0 ». Validez la création d'une unité dans votre répertoire de de développement. Une nouvelle fiche est créée et ajoutée à votre projet, elle se nomme : « WIA_TLB.pas». Vous êtes maintenant prêt à développer votre application. Nous allons maintenant voir comment utiliser cette bibliothèque pour interagir avec vos périphériques d'acquisition d'images.

III Quelques codes utiles

1. Sélection d'un périphérique par l'intermédiaire d'une boîte de dialogue

La bibliothèque de fonctions met à notre disposition une série de boîtes de dialogue permettant d'effectuer une série d'actions les plus communes en seulement quelques lignes de programmation, ce qui allège bien le travail des développeurs. En effet suffit juste de taper les lignes de code suivantes pour afficher la boîte de sélection d'un périphérique compatible avec WIA Version 2.0. A noter que les périphériques compatibles avec l'ancienne version ne le sont plus avec la version 2.0. C'est pour cette raison que j'ai choisi mon scanner qui est compatible avec la version 2. 0. Voici le code qui permet d'afficher la boîte de dialogue de sélection d'un périphérique :

```
cmnDlg := CoCommonDialog.Create;
connectedDev := cmnDlg.ShowSelectDevice(scannerDeviceType, true, true);
```

Comme vous pouvez le constater on crée d'abord un objet de type ***CoCommonDialog***. Cet objet nous permet ensuite d'avoir accès aux différentes boîtes de dialogue communes. Dans votre cas nous utilisons la fonction qui permet d'afficher la boîte de sélection et qui retourne un objet de type ***IDevice***. Ceci nous permet de garder en mémoire le périphérique sélectionné. Ainsi nous n'aurons plus besoin de rouvrir cette boîte par la suite.

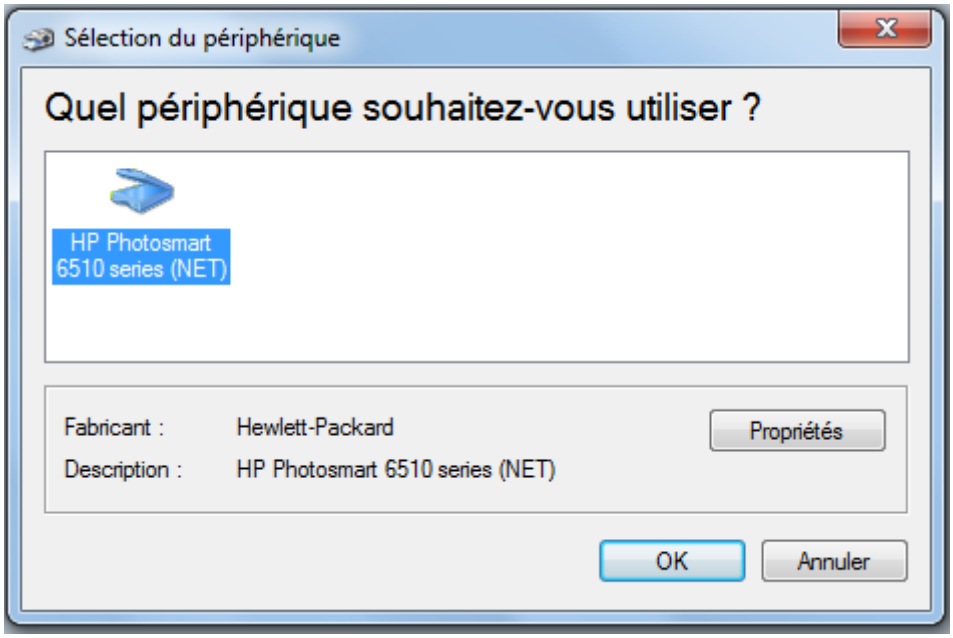
La fonction prend en argument :

1. Le type de périphérique que l'on souhaite utiliser ;
2. Le deuxième paramètre de type boolean qui permet d'ouvrir ou non la boîte de dialogue de sélection d'un périphérique.
3. Le troisième paramètre lui aussi de type booléen permet d'annuler ou non les erreurs survenues lorsque l'utilisateur clique sur le bouton Annuler.

Concernant les types de périphérique, nous avons :

1. ***ScannerDeviceType*** : pour sélectionner un périphérique de type scanner ;
2. ***CameraDeviceType*** : pour sélectionner un périphérique de type appareil photo numérique ;
3. ***VideoDeviceType*** : pour sélectionner un périphérique de type caméra vidéo ;
4. ***UnspecifiedDeviceType*** : pour sélectionner un périphérique ne répondant pas aux trois derniers types (type inconnu).

Voici un aperçu de la boîte de dialogue qui apparaît à l'écran :



2. Obtenir des informations sur le périphérique

Vous pouvez, par l'intermédiaire de votre objet de type ***IDevice*** (connectedDev ici), récupéré des informations concernant le périphérique que vous utilisez. Pour ce faire c'est très simple, nous avons à notre disposition des types d'objet ***IProperties*** et ***IProperty*** qui sont rattaché à notre objet de type ***IDevice***. Il suffit d'énumérer ces propriétés pour accéder à leur valeur. Voici le bout de code qui nous permet de récupérer les valeurs des propriétés Name, Type, Description, Port, Horizontal Optical Resolution, et Vertical Optical Resolution et pour finir WIA Version :

```
procedure TWIAConnectedDevice.getDeviceProperties;
var count, i : integer;
   index : OleVariant;
   prop : IProperty;
   propValue, propName : WideString;
begin
  Properties := connectedDev.Properties;
  count := Properties.Count;
  for i:=1 to count do
  begin
    index := Olevariant(i);
    prop :=  Properties.Item[index];
    propValue := String(prop.Get_Value);
    propName := prop.Name;
   case AnsiIndexStr(propName, ['Description', 'Port', 'Name', 'Type', 'Vertical Optical Resolution',
'Horizontal Optical Resolution', 'WIA Version']) of
       0 : setDescription(propValue);
       1 : setPort(propValue);
       2 : setName(propValue);
       3 : setDeviceType(propValue);
       4 : setHorizontalOpticalRes(propValue);
       5 : setVerticalOpticalRes(propValue);
       6 : setWIAVersion(propValue);
    end;
  end;
end;
```

avec Properties de type ***IProperties*** et prop de type ***IProperty***.

Dans un premier temps on récupère le nombre de propriétés grâce à la propriété count de l'objet ***IProperties***. On « scanne » ensuite toutes les propriétés par l'intermédiaire de notre boucle (for to do). Dans cette boucle il est à noter que l'on doit faire un changement de type pour l'index qui est de type ***OleVariant*** dans la bibliothèque de fonctions. Autre chose à noter pour notre case of qui n'accepte que des valeurs ordinales nous sommes obligés de passer par la fonction ***AnsiIndexStr*** et de lui passer en paramètre les noms des propriétés auxquelles nous voulons accéder pour obtenir leurs valeurs associées. C'est un peu lourd, mais on ne peut pas faire autrement.

Maintenant que l'on a sélectionné un « device » et que l'on a pu récupérer certaines de ces données, il est temps d'acquérir une image à partir de celui-ci.

3. Boîte de dialogue d'acquisition d'une image à partir d'un scanner

Là encore la bibliothèque nous aide énormément en nous mettant à disposition une fonction permettant d'afficher une boîte de dialogue permettant d'acquérir une image scannée.

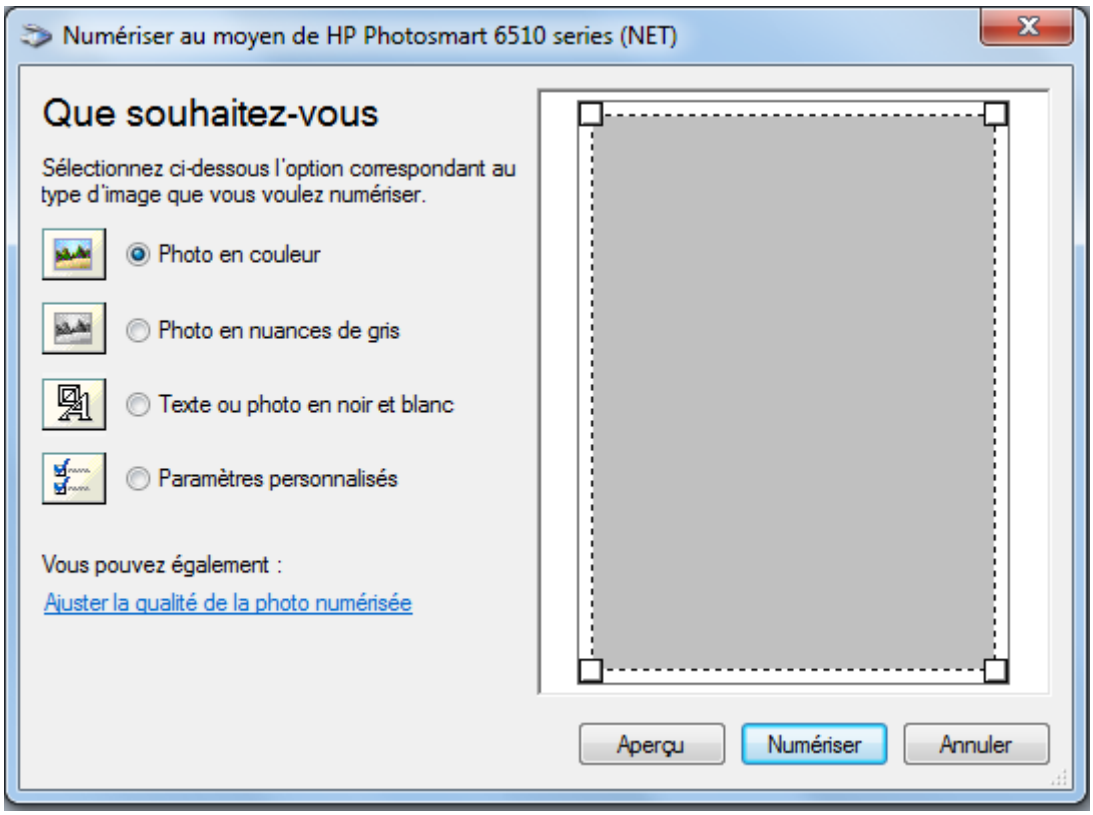
Ceci s'effectue toujours par l'intermédiaire de l'objet de type ***ICommonDialog***. Voici le code a inséré pour effectuer cette action :

```
image := cmnDlg.ShowAcquireImage(ScannerDeviceType,
                                 UnspecifiedIntent,
                                 MaximizeQuality,
                                 wiaFormatBMP,
                                false,
                                false,
                                true);
```

Nous pouvons voir que cette fonction retourne un objet de type ***IImageFile***. Comme vous pouvez le constater cette fonction nécessite quelques paramètres :

1. Le type de périphérique utilisé ;
2. Le deuxième paramètre permet de spécifier si l'on veut faire une acquisition d'image en niveaux de gris, en couleurs, ou de texte seulement. Lorsque la valeur est initialisée à « UnspecifiedIntent » la boîte s'ouvre avec toutes ses options.
3. Le troisième paramètre permet de spécifier le niveau de qualité de l'image scannée.
4. Le quatrième paramètre permet de spécifier le format de l'image, ici on acquiert une image u format bmp.
5. Le cinquième paramètre permet de spécifier si on doit rouvrir la boîte de sélection du périphérique.
6. Le sixième paramètre permet de spécifier si on veut afficher la boîte de dialogue de paramétrages de l'acquisition d'image.
7. Le dernier permet de spécifier si l'on prend en compte l'annulation des erreurs.

Voici l'aperçu de la boîte de dialogue qui apparaît à l'écran :



Une fois l'objet ***IImageFile*** créé il suffit d'utiliser sa fonction ***SaveFile*** pour l'enregistrer sur le disque, comme ceci :

```
image.SaveFile(sFileName);
```

C'est aussi simple que cela.

4. Obtenir des informations sur une image

Maintenant que nous avons créé notre objet de type ***IImageFile***, nous pouvons avoir accès à certaines de ces propriétés qui vont nous permettre d'obtenir des informations intéressantes comme sa hauteur, sa largeur, ses résolution horizontales et verticales, son extension de fichiers, ainsi que son « Pixel depth ». Il suffit d'appeler les propriétés correspondantes de l'objet :

```
setHeight(image.Height);
setWidth(image.Width);
setVerticalResolution(image.VerticalResolution);
setHorizontalResolution(image.HorizontalResolution);
setFileExtension(image.FileExtension);
setPixelDepth(image.PixelDepth);
```

A noter que l'objet de type ***IImageFile*** peut très bien être créé pour charger une image située localement sur le disque dur. Pour ce faire, nous devons appeler la fonction ***LoadFile*** qui prend en paramètre le chemin physique du fichier image à ouvrir :

```
image := CoImageFile.Create;
image.LoadFile(sFileName);
```

On prendra soin bien sûr de créer l'objet par l'intermédiaire du constructeur de la classe CoImageFile.

Une fois l'objet créé on peut demander les mêmes renseignements comme vu précédemment par l'intermédiaire de ses propriétés.

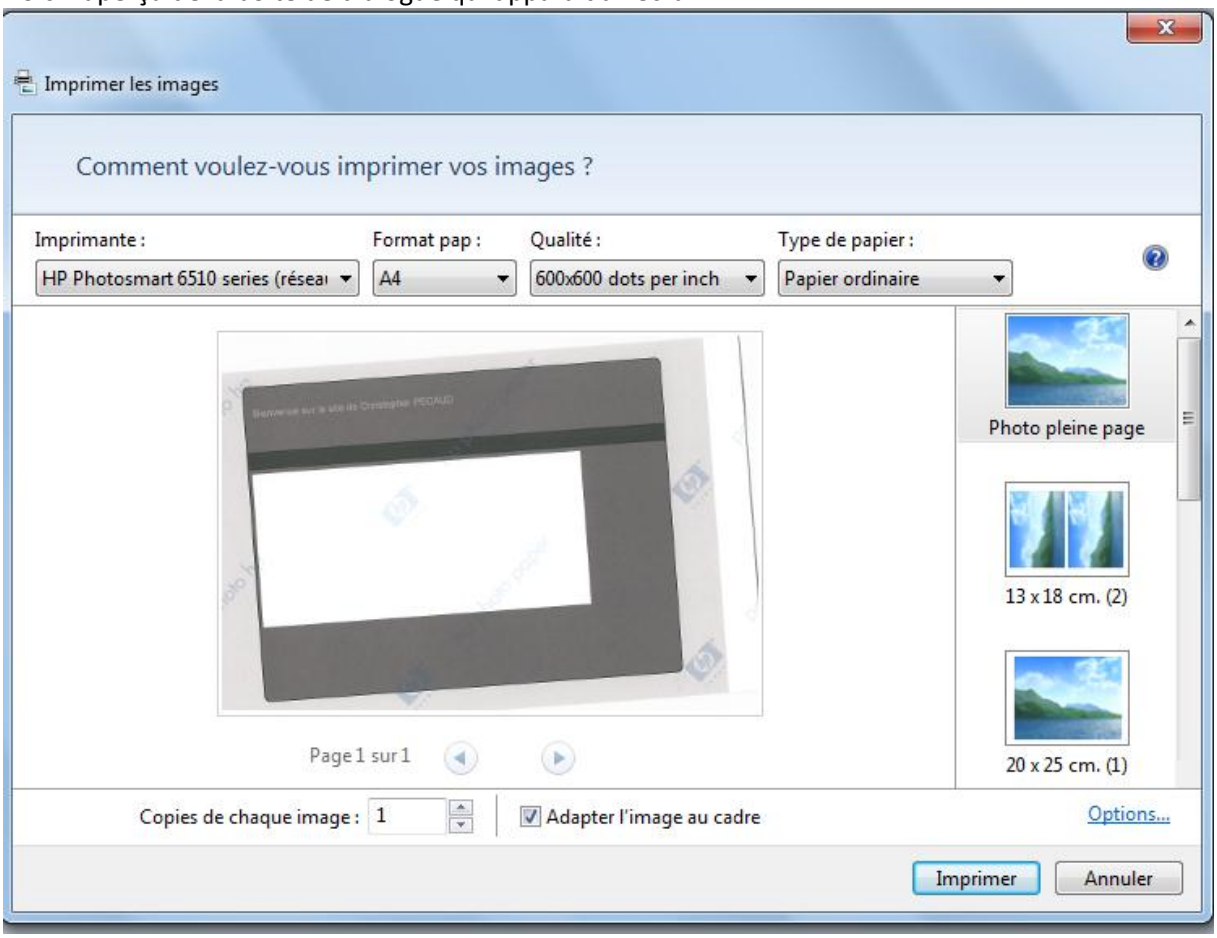
Nous allons voir dans un dernier temps comment imprimer une image, à partir d'une boîte de dialogue encore mise à notre disposition par la bibliothèque.

5. Imprimer une image en utilisant l'objet de type ICommonDialog

Là encore notre tâche est encore grandement facilitée par l'intermédiaire de cet objet. Il suffit d'utiliser la fonction ***showPhotoPrintingWizard*** qui prend en argument le nom du fichier de l'image à imprimer. Attention toutefois il y a une petite subtilité, il faut convertir le nom en ***OleVariant***. Voici le code nécessaire pour imprimer une image en utilisant ce procédé :

```
procedure TWIAScanner.printImage(sFileName : string);
var OleFileName : OleVariant;
begin
OleFileName := sFileName;
cmnDlg.ShowPhotoPrintingWizard(OleFileName);
end;
```

Voici l'aperçu de la boîte de dialogue qui apparaît à l'écran :



IV Conclusion

Je vous ai proposé par l'intermédiaire de ce tutoriel un aperçu des fonctions offertes par la bibliothèque de fonctions « Windows Image Acquisition Library ». Elle est fournie avec un ensemble d'objet et de fonctions facilitant vraiment la tâche de développement d'applications orientées gestion de l'acquisition de l'image. Mais celle-ci ne s'arrête pas là en effet elle propose de même un objet permettant de faire du traitement d'image. Nous verrons ceci d'un peu plus près dans un prochain tutoriel.