

L'UAC et Delphi

par Pascal Fonteneau ([Accès Site](#))

Date de publication : 19/03/2008

Dernière mise à jour :

Avec l'arrivée de VISTA, nous avons eu pour la plupart d'entre nous quelques ennuis de compatibilité avec nos programmes ou ceux que nous utilisons. La faute à qui me direz-vous! A l'UAC (Contrôle de compte utilisateur), c'est du moins ce que l'on entend un peu partout sur les forums. Depuis quelques temps, pas mal d'infos sont disponibles sur les solutions à ces problèmes de compatibilité. Le but de cet article est de centraliser les informations disponibles et d'expliquer le plus simplement possible ce que fait l'UAC, pourquoi et comment elle le fait. En découlera, pour nous programmeurs, ce qu'il est nécessaire d'implémenter dans nos programmes Delphi, pour une prise en compte correcte de l'UAC et ainsi contribuer à la sécurité du système de nos utilisateurs.

I - Pourquoi UAC

I-A - Emplacement des données

I-B - Confirmation d'exécution

II - Quelques règles de bonne conduite

III - Comment faire pour

III-A - Déplacer les données vers un répertoire autre que ceux protégés.

III-B - Rendre rapidement opérationnel un programme dont vous n'avez pas le source

III-B-1 - Ajout d'un fichier manifest interne:

III-C - Lancement d'un exe en mode administrateur depuis un autre programme Delphi.

III-C-1 - Via du code directement depuis Delphi

III-C-2 - La solution "OBJET COM"

III-C-3 - Solutions extrêmes

III-D - Ajouter un bouclier sur un bouton TButton

III-E - Ajout du bouclier à un bouton d'un TTaskDialog

IV - Pour finir

V - Epilogue

I - Pourquoi UAC

La plateforme Windows a toujours été la cible de nombreux virus et autres joyeusetés. Placer judicieusement un verrou lorsque nécessaire est une action de nature à limiter l'exposition du système à du code malveillant.

La philosophie est assez simple. Elle se résume principalement en deux points :

I-A - Emplacement des données

Les données n'ont plus leur place dans certains répertoires comme program files, windows et system32 et dans une bonne partie de la registry.

Tout programme existant ne remplissant pas ce premier point ne se comportera pas sous VISTA de manière similaire à XP. Même avec le lancement du programme en mode administrateur vous n'avez aucune garantie que le code s'exécutera correctement, puisque :

Les écritures à destination d'un répertoire protégé par l'UAC seront redirigées vers un répertoire virtuel situé sous **CompteUtilisateur\appdata\Local\VirtualStore**

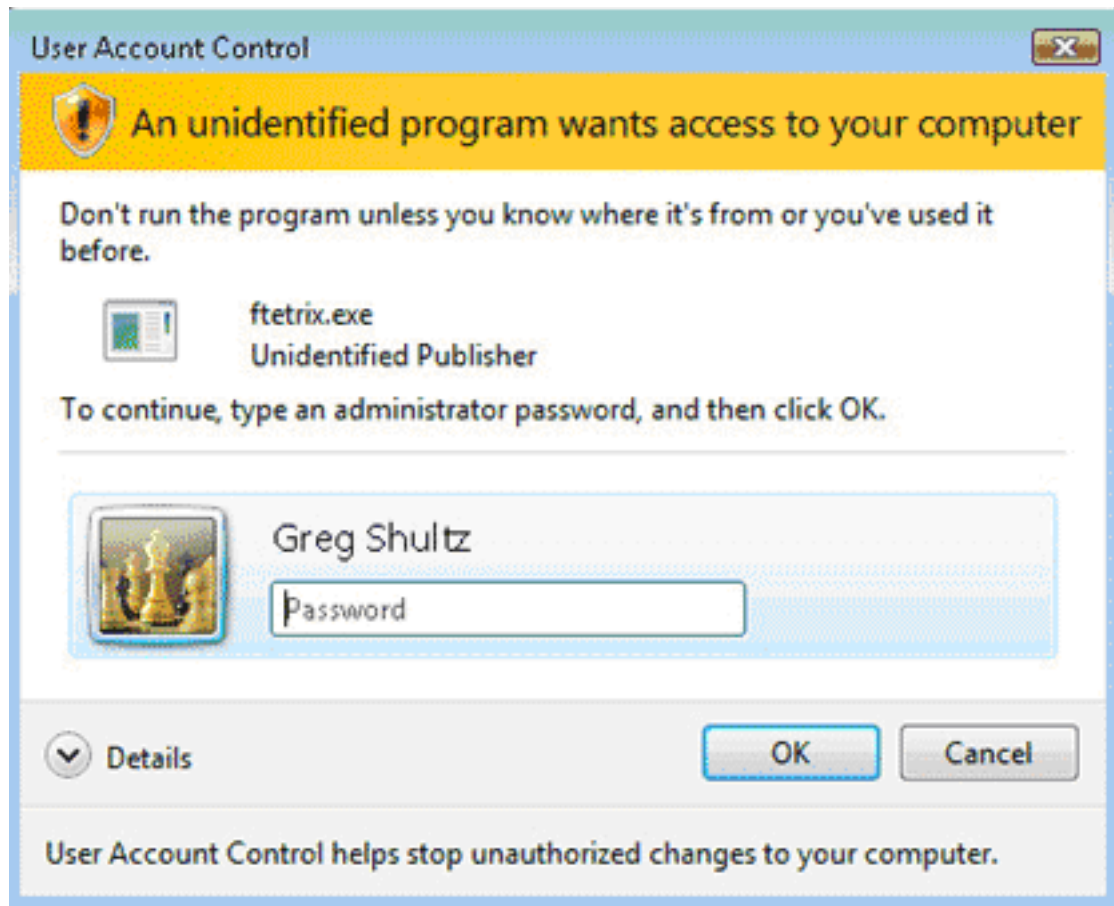
ET

L'écriture des données à destination de la Registry subit le même sort. Elle est redirigée vers **HKEY_CURRENT_USER\Software\Classes\VirtualStore\MACHINE\SOFTWARE**.

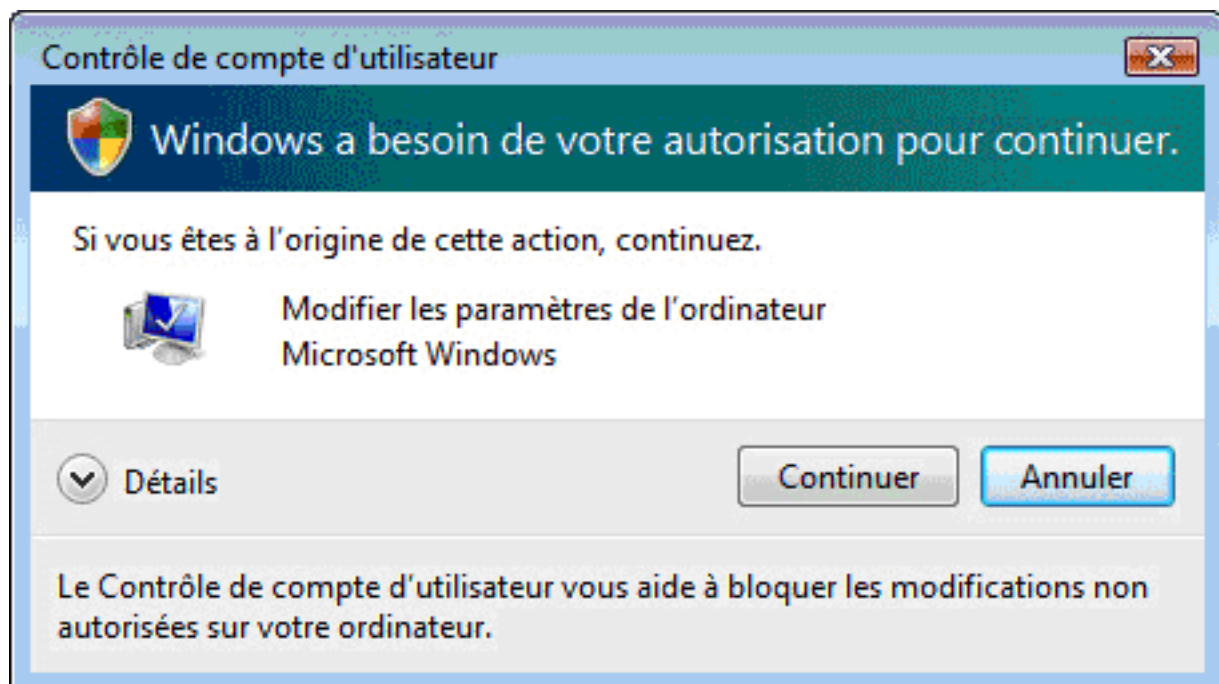
Ces deux emplacements ont été introduit pour assurer un certain niveau de compatibilité, force est de constater que même si la démarche est louable, dans les faits il existe beaucoup trop de limites à ce dispositif pour être utilisé avec efficacité...

I-B - Confirmation d'exécution


Que vous soyez utilisateur standard ou administrateur, tout lancement d'un processus devant accéder au système doit provoquer l'appel d'un écran de confirmation d'exécution. Deux boites de dialogues différentes assurent cette tâche.



En mode utilisateur, demande du mot de passe pour une élévation de privilège.



En mode administrateur, une simple confirmation.

 *C'est le processus (l'exe ou le COM) dans sa globalité qui fonctionne en mode administrateur ou utilisateur. N'envisagez donc pas d'avoir dans le même programme une partie du code demandant l'élévation de privilège administrateur. C'est au lancement du programme qu'il vous faut déterminer si le mode utilisateur standard suffit ou si le mode administrateur sera nécessaire.*

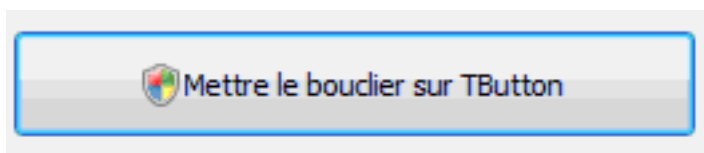
II - Quelques règles de bonne conduite

Un ensemble de règles pour les nouveaux développements et bien sûr pour la reprise des anciens programmes peuvent facilement être appliquées.

- 1 Placer les données utilisateurs dans le répertoire ou sous-répertoire du compte de l'utilisateur ou dans un répertoire commun.
- 2 Séparation du code en deux exécutables (ou 1 exécutable + 1 composant COM). Toutes fonctions ne nécessitant pas une élévation de privilège seront placées dans le programme principal. Les fonctions réservées à l'administrateur seront regroupées dans le second programme. Le premier ou le second processus devra disposer d'un mécanisme d'élévation de privilège pour les fonctions administrateurs.
- 3 Définir le niveau de sécurité que requiert votre programme et créer un fichier manifest avec un des trois niveaux suivants (voir comment plus loin, pour la création du fichier manifest).
 - **AsInvoker** = le programme s'exécute avec le même niveau de droit que le parent.
 - **HighestAvailable** = l'application s'exécute avec les privilèges les plus hauts que l'utilisateur puisse obtenir.
 - **RequireAdministrator** = l'application ne peut être exécutée que par un administrateur.

Dans les 3 cas, il n'y aura plus de redirection vers VirtualStore.

- Dans vos sources, ne pas ouvrir les fichiers situés dans les répertoires protégés ou dans la registry avec un Flag WRITE ou dans un mode autorisant l'écriture.
- Ne pas utiliser le dossier de redirection (VirtualStore) pour vos nouveaux programmes et faites en sorte que ceux existants ne l'utilisent plus, car il n'est pas garanti que ce dossier et sa philosophie d'utilisation soient maintenus dans les prochaines versions de Windows.
- Est-il nécessaire de le préciser, ne jamais désactiver l'UAC, même, si cela permet de rendre à nouveau opérationnel un programme dans l'urgence. Il y a d'autres solutions.
- Informer l'utilisateur de la nécessité d'une élévation des droits par le symbole Bouclier (Shield)



Bouclier (Shield)

Pour le moment, nous en savons suffisamment pour commencer à travailler, mais je vous conseille néanmoins la lecture de l'excellent article de Jacques MASSA sur l'UAC:

<http://msdn2.microsoft.com/fr-fr/library/bb469893.aspx>

Cet article n'est pas orienté Delphi (et pour cause), mais vous y trouverez un complément d'informations et disposerez ainsi d'une base de connaissance solide.

III - Comment faire pour

III-A - Déplacer les données vers un répertoire autre que ceux protégés.

Pour cela, utilisez les fonctions habituelles de localisation des répertoires particuliers

<http://delphi.developpez.com/faq/?page=repertoire#specialdirectory>

III-B - Rendre rapidement opérationnel un programme dont vous n'avez pas le source

 *Opérationnel ne veut pas dire compatible VISTA au sens UAL du terme.*

Par l'ajout d'un fichier manifest externe: Voici le source d'un fichier manifest édité avec le notepad et enregistré sous le nom *NonExecutable.Exe.Manifest* et dans le même répertoire que l'exécutable. Ligne 8, choisissez le niveau de protection entre *AsInvoker*, *HighestAvailable*, et *RequireAdministrator*.

Exemple de code par l'ajout d'un fichier manifest externe

```
<?xml version="1.0" encoding="utf-8" ?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity version="1.0.0.0" processorArchitecture="X86" name="Project3" type="win32" />
  <description> essai Application </description>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="highestAvailable" /> ICI
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Au prochain lancement, vous serez invité à passer en mode administrateur. A partir de ce moment, le programme se comportera comme sous XP, c'est à dire sans redirection.

III-B-1 - Ajout d'un fichier manifest interne:

Créez le même fichier manifest que précédemment. Nommez-le *MonManifest.manifest* par exemple.

Toujours avec notepad, créez un second fichier nommé *MonManifest.rc* et contenant cette simple ligne

```
1 24 "MonManifest.manifest"
```

Depuis le répertoire contenant ces 2 fichiers, passez en mode commande DOS (cmd) Puis compilez la ressource en tapant :

```
brcc32 MonManifest.rc
```

Vous obtiendrez un fichier de ressource nommé *MonManifest.RES*. Ajoutez cette ligne **{\$R 'MonManifest.RES' 'MonManifest.rc' }** au fichier *project.dpr* pour l'inclure en tant que ressources lors de la compilation Delphi. ATTENTION, compilez mais ne lancez pas le programme car vous ne pourrez plus créer de processus depuis Delphi dès lors que l'inclusion du manifest dans le programme sera prise en compte. Vous devrez désormais lancer le programme depuis l'explorateur Windows.

```
program
{$R 'MonManifest.RES' 'MonManifest.rc' } // ajouter cette ligne ici dans le source du projet
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}
```

III-C - Lancement d'un exe en mode administrateur depuis un autre programme Delphi.

III-C-1 - Via du code directement depuis Delphi

Cas typique de l'updater, le programme principal lance un second programme avant de mourir. Ce second programme se charge du download, de la décompression éventuelle, de l'installation et de la relance du programme principal en nouvelle version. Dans ce cas, utilisez simplement ce code (pas besoin de manifest)

```
procedure RunAsAdmin(hWnd : HWND; aFile : String; aParameters : String);
var
  sei : TShellExecuteInfoA;
begin
  Fillchar(sei, SizeOf(sei), 0);
  sei.cbSize := SizeOf(sei);
  sei.Wnd := hWnd;
  sei.fMask := SEE_MASK_FLAG_DDEWAIT or SEE_MASK_FLAG_NO_UI;
  sei.lpfile := PChar(aFile);
  sei.lpVerb := 'runas';
  sei.lpParameters := PChar(aParameters);
  sei.nShow := SW_SHOWNORMAL;
  if not ShellExecuteEx(@sei) then
    RaiseLastOSError;
end;

// choisir et lancer un executable en mode administrateur
if OpenFileDialog1.Execute then
  RunAsAdmin(self.Handle, OpenFileDialog1.FileName, '');
```

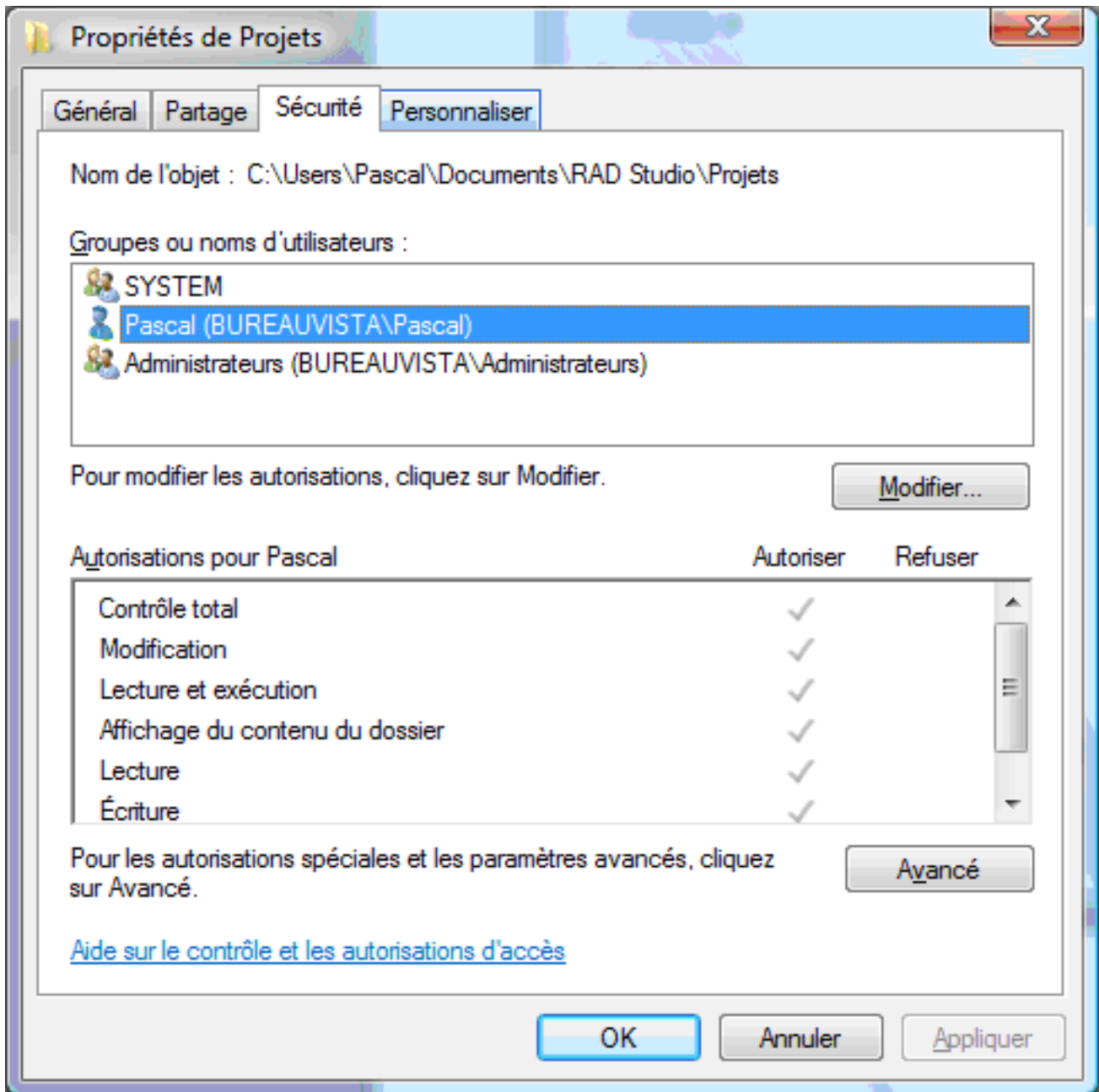
III-C-2 - La solution "OBJET COM"

Cette solution n'est pas développée ici, cependant vous pouvez consulter [ce lien](#)

III-C-3 - Solutions extrêmes

Si vos utilisateurs n'ont pas accès au mode administrateur et que vous n'avez pas les sources, il reste encore les possibilités suivantes :

- 1 Déplacer le programme et ses données dans un répertoire non protégé comme par exemple C:\ChezMoi\
- 2 Changer les droits du répertoire racine du programme sur la machine de l'utilisateur



Propriétés de Projet

- Si vous développez des Freeware ou des Shareware, vous n'avez pas accès au système de l'utilisateur. Mais vous pouvez en même temps changer les droits du répertoire dans lequel votre programme sera installé.

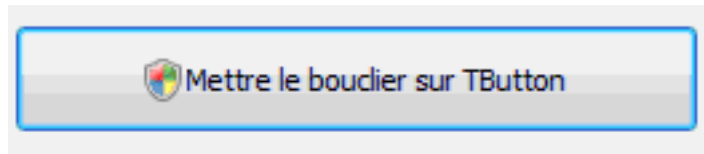
Exemple avec InnoSetup

```
[Dirs]
Name: "{app}";Permissions:users-full authusers-ful
```

- ⚠ Attention, ces possibilités vont à l'encontre de la sécurité proposée par l'UAC. Elles sont donc à utiliser en dernier recours.**

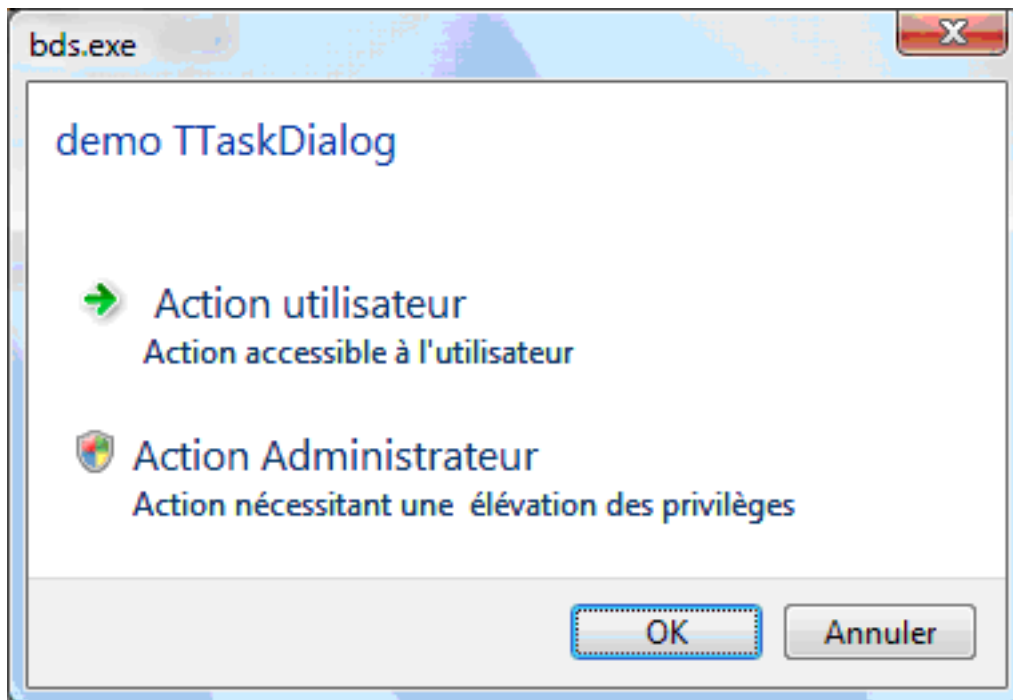
III-D - Ajouter un bouclier sur un bouton TButton

```
// necessite uses commctrl;  
Button_SetElevationRequiredState(Button1.Handle, True);
```



III-E - Ajout du bouclier à un bouton d'un TTaskDialog

Mettre simplement la propriété ElevationRequired à True pour le bouton.



IV - Pour finir

Lorsque l'UAC repère un des mots Install, Update ou Setup dans le nom d'un exécutable, la boîte de contrôle de compte utilisateur s'affichera au lancement du programme. Ceci peut poser un problème de debug de vos exe sous Delphi voir l'article suivant :

<http://dn.codegear.com/article/33493>

V - Epilogue

Lors de la recherche du fond pour cet article, je me suis aperçu que bien souvent des personnes recherchent un bug dans le moteur du BDE, la registry ou des opération E/S (DeleteFile, CreateFile, Registry) alors que c'est simplement la redirection des opérations d'écriture vers le dossier virtualstore qui fausse l'exécution. Alors, je rajoute volontairement ce petit épilogue, afin que le moteur de recherche du site indexe aussi cet article pour les mots BDE, Registry, DeleteFile et CreateFile.

