L'utilisation des composants ActiveX sans inscription préalable(1ère partie)

Par omen999

Date de publication : 19 janvier 2010

Certains composants ActiveX permettent d'étendre considérablement les capacités des langages VBScript/JScript mais au prix d'un enregistrement préalable dans la base de registre. Il existe néanmoins une solution qui lève cette contrainte.

Vous pouvez également poser vos questions et laisser vos commentaires et suggestions sur le forum : **Commentez**

1 - Introduction....3
2 - Les solutions identifiées....3
3 - Mise en oeuvre....3
3-1 - Application manifest - wscript.exe.manifest....4
3-2 - Assembly manifest - dynwrapx.sxs.manifest....4
3-3 - Démonstration avec un script....5
4 - Limitations et inconvénients....6
5 - Liens....6
6 - Remerciements....6

Copyright ® 2010 omen999. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/

1 - Introduction

VBScript comme JScript possèdent nativement la capacité de créer - ou de récupérer - des objets exposés par des composants ActiveX munis d'une interface IDispatch respectant la norme COM/OLE définie par Microsoft.

Cette norme impose notamment l'inscription préalable du composant, c'est-à-dire la déclaration de toutes ses interfaces avec leurs caractéristiques, dans la base de registre. Le client COM, en l'occurence le script, retrouve en consultant cette dernière les points d'entrée des différentes classes exposées par ce composant. Toute tentative d'instancier un objet à partir d'un composant non enregistré se traduit par un message d'erreur bien connu (VBS) : "*Un composant ActiveX ne peut pas créer un objet : ...*"

Si cette inscription s'effectue simplement au moyen de l'utilitaire **regsvr32.exe**, on peut quelques fois souhaiter s'affranchir de cette contrainte, tout particulièrement lorsqu'on ne dispose pas des droits d'administrateur. Cet article se propose de décrire une solution applicable non seulement aux langages VBScript et JScript exécutés dans le contexte Windows Script Host, mais aussi à tous les langages Active Scripting dès lors qu'ils ont la capacité de gérer les objets COM/OLE. L'exemple choisi pour illustrer cette première partie concerne un composant russe très simple - DynamicWrapperX - sans ressource typelib, qui n'expose qu'une seule classe et dont j'ai traduit par ailleurs la documentation. Ce wrapper permet aux VBScript/JScript d'appeler directement les fonctions des bibliothèques dll.

2 - Les solutions identifiées

Classiquement, deux techniques existent pour contourner cette limitation :

- L'émulation de la fonction CoCreateInstance

Elle consiste à appeler directement la fonction DllGetClassObject présente dans chaque composant. Malheureusement, elle ne nous dispense pas de l'obligation d'inscrire le composant qui aura pour tâche d'assurer l'émulation qui vise justement à nous éviter d'y procéder...
C'est donc une impasse qui renvoie, comme chacun l'aura compris, à l'angoissante problématique de l'incomplétude ontologique. Une piste à explorer consisterait à la combiner avec la deuxième technique (*cf infra*), mais c'est un autre sujet.

- L'utilisation de la technologie SxS

A partir de la version XP SP2 (et serveur 2K3), Microsoft a introduit un nouveau mécanisme permettant à une application de gérer des composants COM sans avoir besoin de les enregistrer. A l'initialisation, chaque nouveau process recherche désormais la présence, dans le répertoire de l'exécutable, d'un fichier *application manifest* contenant les métadonnées qui vont lui permettre de créer un contexte d'activation. Ce contexte fournira au client les mêmes éléments que ceux obtenus de la base de registre. Ce fichier, qui porte l'extension *.manifest*, est un simple fichier texte au format xml. Cette solution fait l'objet du présent article.

3 - Mise en oeuvre

En premier lieu, un fichier application manifest doit être créé et placé dans le même répertoire que l'exécutable. Comme le script s'exécute dans le contexte Windows Script Host, l'exécutable concerné est **wscript.exe** normalement situé dans le répertoire system32. Ce fichier sera donc logiquement nommé :

wscript.exe.manifest

L'*assembly*, c'est-à-dire l'ensemble des modules formant un tout unique et indivisible, est constitué du ou des fichiers du composant et de son *assembly manifest*. Comme il s'agira d'un *private assembly*, il doit être placé dans le répertoire de l'exécutable. Pour être tout à fait exact, il peut également être placé dans un sous-répertoire selon les modalités décrites dans le chapitre "*Assembly Searching Sequence*" de la documentation en ligne fournie par Microsoft [1].

Copyright ® 2010 omen999. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/

Le composant DynamicWrapperX se présente sous la forme d'un petit fichier dénommé dynwrapx.dll ainsi notre *assembly* sera constitué des 2 fichiers suivants :

dynwrapx.dll
dynwrapx.sxs.manifest

Nous possédons les binaires, il ne reste plus qu'à écrire les *manifests*.

3-1 - Application manifest - wscript.exe.manifest

wscript.exe.manifest
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">

 <assemblyIdentity
  type="win32"
  name="wscript.exe"
  version="1.0.0.0"/>

  <dependency>
  <dependentAssembly>
         <assemblyIdentity
                 type="win32"
                 name="dynwrapx.sxs"
                 version="1.0.0.0"/>
  </dependentAssembly>
  </dependency>

  <dependency>
         <dependentAssembly>
               <assemblyIdentity
                 type="win32"
                 name="Microsoft.Windows.Common-Controls"
                 version="6.0.0.0"
                 publicKeyToken="6595b64144ccf1df"
                 language="*"
                 processorArchitecture="x86"/>
         </dependentAssembly>
  </dependency>

</assembly>
```

assemblyIdentity comporte les trois attributs minimums requis : *type*, *name* et *version* qui permettent d'identifier **wscript.exe**.
dependency et *dependentAssembly* définissent les composants liés à l'exécutable. Là encore, les trois attributs minimums ont été utilisés pour lier le composant DynamicWrapperX, ou plus exactement son *assembly manifest* que nous verrons ci-dessous.

La deuxième dépendance concerne la bibliothèque des contrôles XP qui permet d'appliquer le style XP à tous les contrôles visuels (les fonctions VBS *MsgBox* ou *InputBox* permettent un test simple).
Il s'agit d'un *shared assembly* que l'on retrouve dans le répertoire %windows%\WinSxS. Les plus curieux constateront que l'attribut version est inopérant et que la bibliothèque effectivement chargée sera normalement celle la plus récente, conformément aux redirections définies dans le fichier policy (répertoire %windows%\WinSxS\Policies).

3-2 - Assembly manifest - dynwrapx.sxs.manifest

dynwrapx.sxs.manifest
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">

 <assemblyIdentity
     type="win32"
```


Copyright ® 2010 omen999. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/

dynwrapx.sxs.manifest
```
      name="dynwrapx.sxs"
      version="1.0.0.0"/>

 <file name="dynwrapx.dll">
     <comClass
         description="DynamicWrapperX Class"
         clsid="{89565275-A714-4a43-912E-978B935EDCCC}"
         threadingModel="Both"
         progid="DynamicWrapperX"/>
 </file>

</assembly>
```

Concernant l'élément *assemblyIdentity*, mêmes observations que pour l'*application manifest*.

Pour l'élément *file*, seul l'attribut *name* est requis. On peut y ajouter les attributs *hash* et *hashalg* pour contrôler l'intégrité du composant mais cet exemple est, volontairement, le plus simple possible.

Un composant peut exposer plusieurs classes qui doivent être décrites dans des sous-éléments *comClass* séparés. Seul l'attribut *clsid* du sous-élément *comClass* est normalement requis, mais il est souhaitable d'y ajouter les attributs *description*, *progid* et *threadingModel* afin d'identifier clairement les différentes classes disponibles.

Après la création des *manifests*, il reste une dernière étape : choisir l'emplacement de nos différents fichiers.
La solution la plus simple mais pas nécessairement la plus commode consiste à placer tous les fichiers dans le répertoire system32. Une solution plus souple (et qui est également la moins intrusive) consiste à placer, dans un même répertoire séparé, les *manifests*, le ou les fichiers du composant, une copie de **wscript.exe** ainsi que le script. Un simple batch lancera alors le script à partir de ce répertoire.

3-3 - Démonstration avec un script

Pour ne pas déroger à la tradition, nous allons illustrer cette initiation avec un script écrit en VBScript qui affichera le message *Hello world !* et que nous appellerons **hello.vbs**.

hello.vbs
```
Option Explicit
Const _
MB_ICONWARNING = &H30, _
MB_YESNO = &H4

Dim oWrap,iRep
Set oWrap = CreateObject("DynamicWrapperX")
oWrap.Register "user32.dll", "MessageBoxW", "i=hwwu","r=l"
iRep = oWrap.MessageBoxW(0, "Hello world !", "RegFreeSxS par omen999", MB_YESNO + MB_ICONWARNING)
```


*Si l'on souhaite regrouper l'ensemble des fichiers dans un répertoire séparé, il importe de ne pas oublier les conditions de lancement du script. Les fichiers d'extension .vbs sont habituellement associés à l'exécutable **wscript.exe** placé dans le sous-répertoire system32. Pour obtenir la lecture des fichiers manifests, il est nécessaire de forcer le lancement de l'hôte WSH à partir de notre répertoire. La façon la plus simple d'y procéder est de créer un petit fichier batch appelé **hello.bat**.*
(La commande start /B est purement cosmétique et force la fermeture de la fenêtre de commande)

hello.bat
```
start /B wscript hello.vbs
```


Copyright ® 2010 omen999. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/

4 - Limitations et inconvénients

Comme indiqué ci-dessus, cette méthode n'est applicable qu'à partir de la version XP SP2. Chaque composant impose l'écriture d'un *assembly manifest* qui doit décrire l'ensemble des classes exposées, outre la mise à jour de l'*application manifest* de wscript. Un explorateur d'objets est indispensable pour déterminer les données du composant ainsi que pour créer ce fichier manuellement. Il arrive cependant que certains composants présentent des interfaces non standards qui peuvent réserver des surprises (ex : pas d'implémentation de GetTypeInfo).

Pour les moins motivés, il existe des utilitaires qui permettent de créer ces fichiers *manifests* :

- Regsvr42.exe - opensource [2]
- Side-by-Side Manifest Maker - commercial [3]

5 - Liens

Documentation Microsoft Assembly Searching Sequence

Site regsvr42

Site Side-by-Side Manifest Maker

Blog décrivant un exemple javascript (russe)

6 - Remerciements

Je tiens à remercier **bbil**, **eusebe19** et **Caro-Line** pour leur relecture attentive et leurs observations pertinentes.

Copyright ® 2010 omen999. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.
https://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/