L'utilisation des composants ActiveX sans inscription préalable dans la base de registre(2ème partie)

Par omen999

Date de publication : 12 novembre 2011

Cette deuxième partie détaille la syntaxe des fichiers *SxS manifests*, leur intégration comme ressource dans un fichier PE (*Portable Executable*) et enfin les autres solutions qui ne font pas appel nécessairement aux *Side-by-Side assemblies*. Un exemple d'écriture est également fourni.

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

1 - Introduction....4
2 - Syntaxe des fichiers SxS manifests....4
2-0 - Format des fichiers....4
2-1 - Application manifest....4
2-1-1 - assembly (requis)....4
2-1-1-a - manifestVersion (requis)....4
2-1-2 - noInherit....4
2-1-3 - assemblyIdentity (requis)....5
2-1-3-a - type (requis)....5
2-1-3-b - name (requis)....5
2-1-3-c - language....5
2-1-3-d - processorArchitecture....5
2-1-3-e - version (requis)....5
2-1-3-f - publicKeyToken....5
2-1-4 - dependency....5
2-1-5 - dependentAssembly....5
2-1-6 - file....6
2-1-6-a - name....6
2-1-6-b - hashalg....6
2-1-6-c - hash....6
2-2 - Assembly manifest....6
2-2-1 - assembly (requis)....6
2-2-1-a - manifestVersion (requis)....6
2-2-2 - noInheritable....6
2-2-3 - assemblyIdentity....6
2-2-3-a - name (requis)....7
2-2-3-b - language....7
2-2-4 - dependency....7
2-2-5 - dependentAssembly....7
2-2-6 - file....7
2-2-7 - comClass....7
2-2-7-a - description....7
2-2-7-b - clsid (requis)....7
2-2-7-c - threadingModel....7
2-2-7-d - tlbid....8
2-2-7-e - progid....8
2-2-7-f - miscStatus....8
2-2-7-g - miscStatusIcon....8
2-2-7-h - miscStatusContent....8
2-2-7-i - miscStatusDocPrint....8
2-2-7-j - miscStatusThumbnail....8
2-2-8 - typelib....8
2-2-8-a - tlbid (requis)....8
2-2-8-b - version (requis)....8
2-2-8-c - helpdir (requis)....9
2-2-8-d - resourceid....9
2-2-8-e - flags....9
2-2-9 - comInterfaceExternalProxyStub....9
2-2-9-a - iid (requis)....9
2-2-9-b - baseInterface....9
2-2-9-c - numMethods....9
2-2-9-d - name....9
2-2-9-e - tlbid....9
2-2-9-f - proxyStubClsid32....10
2-2-10 - comInterfaceProxyStub....10
2-2-10-a - iid (requis)....10
2-2-10-b - name (requis)....10
2-2-10-c - tlbid....10

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-10-d - baseInterface.....10
2-2-10-e - numMethods.....10
2-2-10-f - proxyStubClsid32.....10
2-2-10-g - threadingModel.....10
2-2-11 - windowClass.....10
2-2-11-a - versioned.....11
3 - Écriture d'un exemple.....11
3-1 - Application manifest - wscript.exe.manifest.....11
3-2 - Assembly manifest - msado15.sxs.manifest.....11
3-3 - Le script testRegFreeAdo.vbs.....17
4 - Inclusion sous la forme d'une ressource - utilitaire mt.exe.....18
5 - Les autres solutions.....19
5-1 - SxS dynamique.....19
5-1-1 - API SxS Assembly.....19
5-1-2 - Objet Microsoft.Windows.ActCtx.....19
5-2 - WSC et moniker.....20
5-3 - Émulation de CoCreateInstance.....20
5-3-1 - Composant ActiveX Pack1 - newObjects.....21
5-3-2 - Bibliothèque RegFreeX.dll - omen999.....21
6 - Liens.....21
7 - Remerciements.....22

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

1 - Introduction

La **première partie** a été consacrée à la description générale du mécanisme SxS (*Side-by-Side assemblies*) introduit par Microsoft à partir de Windows XP SP2. Toutefois, le contenu des fichiers *manifests* a été survolé puisque seuls les éléments et attributs nécessaires au bon fonctionnement du script donné en exemple ont été examinés. Une étude exhaustive de cette syntaxe accompagnée d'un commentaire sur la signification de chaque item nous permettra de maitriser totalement le contenu de ces fichiers *manifests* et de l'adapter en fonction des caractéristiques du composant concerné et des besoins réels du projet envisagé.

Seule la mise en œuvre des *private SxS assemblies* sera détaillée, excluant ainsi les *shared assemblies* qui sont, par nature, susceptibles d'être utilisés par plusieurs applications. Dans un souci de clarté, un exemple plus complet de création de *manifests* est donné et porte sur le composant MS *ActiveX Data Objects* (ADO) qui permet la gestion de différents types de fichiers par VBScript/JScript. En outre, ces *manifests* peuvent également être intégrés comme ressource de tous fichiers au format PE (*Portable Executable*) et ce, au moyen d'un éditeur approprié comme celui fourni par Microsoft. Enfin, il existe des solutions qui ne s'appuient pas exclusivement sur les *SxS manifests* pour exploiter les composants non inscrits et nous les évoquerons dans une dernière section.

2 - Syntaxe des fichiers SxS manifests

2-0 - Format des fichiers

Les fichiers *manifests* doivent respecter la spécification XML 1.0 et sont donc enregistrés au format *Unicode*. Leur arborescence est simple et il n'est pas indispensable de disposer d'un éditeur XML spécifique même s'il est toujours plus confortable de pouvoir vérifier rapidement la conformité du document.

2-1 - Application manifest

Ce fichier peut contenir six éléments dont deux sont normalement requis et un total de dix attributs. **La casse doit être respectée pour les noms mais également pour la valeur de l'attribut type**. Les éléments et attributs requis sont signalés.

2-1-1 - assembly (requis)

C'est l'élément racine de la structure et il doit appartenir à l'espace de nommage "urn:schemas-microsoft-com:asm.v1". Un unique attribut.

2-1-1-a - manifestVersion (requis)

Une seule valeur autorisée : "1.0".

2-1-2 - noInherit

Cet élément est un flag qui inhibe l'héritage du contexte d'activation au profit des *threads* qui pourraient être créés par le même processus. Le composant non-inscrit devient alors inaccessible pour ces *threads*. Il est fortement conseillé d'omettre ce flag qui entraine généralement des dysfonctionnements lorsque l'application n'a pas été spécifiquement créée pour gérer le suivi de son propre contexte d'activation au moyen de l'API idoine.

Il doit toujours être placé immédiatement avant l'élément *assemblyIdentity* concerné par cette inhibition.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-1-3 - assemblyIdentity (requis)

Cet élément identifie de façon unique deux notions distinctes :
1° l'application à laquelle s'applique le *manifest* ;
2° inclus dans un élément *dependentAssembly*, le fichier *assembly manifest* qui identifie le composant non-enregistré que l'on souhaite exploiter.

Cette identification est assurée par six attributs.

2-1-3-a - type (requis)

Toujours "win32" en caractères minuscules.

2-1-3-b - name (requis)

Soit le nom de l'application, soit le nom de l'*assembly manifest* (sans l'extension .manifest). Pour les limitations imposées au choix des noms des *assembly manifests* (cf *infra* [1]).

2-1-3-c - language

Code DHTML du langage spécifique utilisé par l'application ou le composant. Si ceux-ci sont génériques, l'attribut peut être omis pour l'application et contenir la valeur "*" pour l'*assembly*.

2-1-3-d - processorArchitecture

Soit "x86", "ia64" ou "amd64" en fonction de l'application autorisée à utiliser l'*assembly*. À n'utiliser que lorsque le code pose des problèmes de compatibilité de plate-forme. Peut être ignoré le plus souvent.

2-1-3-e - version (requis)

Un numéro de version sous la forme "n.n.n.n" où n est une valeur numérique 16 bits non signée.

2-1-3-f - publicKeyToken

Une chaine représentant les huit derniers octets sous forme de deux caractères hexadécimaux du condensat SHA-1 de la clé publique avec laquelle l'application ou le composant est signé. Cette valeur, qui permet d'éviter les substitutions de fichiers, est facultative pour les *private assemblies*.

2-1-4 - dependency

Cet élément est en principe facultatif mais en pratique obligatoire puisqu'il va contenir un *assembly* qui permettra de définir le composant que l'on souhaite inclure dans le contexte d'activation. Chaque composant donnera lieu à l'écriture d'un élément *dependency* séparé.

2-1-5 - dependentAssembly

Cet élément, obligatoirement inclus dans un élément *dependency* et un seul, contiendra un élément *assemblyIdentity* tel que décrit ci-dessus.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-1-6 - file

Cet élément permet de définir les fichiers privés de l'application. Ces fichiers sont identifiés par trois attributs.

2-1-6-a - name

Nom du fichier avec son extension.

2-1-6-b - hashalg

Algorithme de hachage. Sa valeur doit toujours être "SHA1".

2-1-6-c - hash

Une chaine contenant des valeurs hexadécimales représentant le condensat du fichier.

Note : personnellement, je n'ai pas trouvé d'utilité particulière à cet élément lorsqu'il figure dans un *application manifest*.

2-2 - Assembly manifest

Ce fichier XML contient les données de l'*assembly* référencé dans l'*application manifest*, c'est-à-dire l'ensemble des fichiers et ressources du composant. Il peut comporter jusqu'à 11 éléments et 39 attributs.

2-2-1 - assembly (requis)

Ce premier élément est identique à celui de l'application *manifest* ci-dessus.

2-2-1-a - manifestVersion (requis)

Une seule valeur autorisée : "1.0".

2-2-2 - noInheritable

Cet élément doit être utilisé en coordination avec l'élément *noInherit* lorsque celui-ci figure dans l'application *manifest* et précéder immédiatement l'élément *assemblyIdentity* concerné. Voir l'élément *noInherit supra*[2] pour plus d'explications.

2-2-3 - assemblyIdentity

Cet élément présente des caractéristiques analogues à celui disponible dans l'*application manifest*. Il décrit toutefois deux situations différentes :
- lorsqu'il est le premier sous-élément de l'élément *assembly*, il décrit l'*assembly* "propriétaire" de l'*assembly manifest*. Il est unique et Microsoft le qualifie de *DEF-context assembly* ;
- lorsqu'il est un sous-élément d'un élément *dependentAssembly*, il définit alors un *REF-context assembly* qui décrit de façon unique l'*asssembly* utilisé par le *DEF-context assembly*. À chaque *REF-context assembly* doit correspondre un *DEF-context assembly*.

On retrouve également les mêmes attributs (voir ci-dessus) avec les précisions suivantes :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-3-a - name (requis)

Pour les limitations imposées au choix des noms des *assembly manifests* (cf *infra* [1]).

2-2-3-b - language

Dans le cas d'un *DEF-context assembly*, cet attribut doit être omis si le composant n'est pas localisé. Dans le même cas, s'il s'agit d'un *REF-context assembly*, l'attribut prendra la valeur "*".

En pratique, il n'est pas très courant qu'un composant ait lui-même à dépendre d'autres ActiveX. Le REF-context assembly ainsi que les deux éléments suivants seront donc rarement rencontrés.

2-2-4 - dependency

Il doit contenir au moins un élément *dependentAssembly* qui doit figurer en premier.

2-2-5 - dependentAssembly

Cet élément doit contenir, en premier élément inclus, un *assemblyIdentity*. Chacun de ces éléments *dependentAssembly* doit être inclus dans un élément *dependency*.

2-2-6 - file

Cet élément dont l'utilité est hypothétique dans un *application manifest*, est en revanche indispensable pour définir les caractéristiques propres de l'*assemblyIdentity* propriétaire du *manifest* autrement dit le *DEF-context assembly*.

Il possède les trois attributs déjà décrits ci-dessus. Il suffira juste de préciser que l'attribut *name* correspondra au nom du fichier ActiveX (extension .dll ou .ocx).

Les cinq éléments suivants sont des sous-éléments de l'élément *file*. Même si la documentation les décrit tous comme facultatifs, en pratique, le sous-élément *comClass* est au minimum nécessaire pour décrire les caractéristiques de(s) classe(s) exposée(s) par le composant.

2-2-7 - comClass

2-2-7-a - description

Une courte description de la classe. Pas indispensable mais utile surtout lorsque le composant expose de nombreuses classes.

2-2-7-b - clsid (requis)

Une chaine formatée représentant le GUID qui identifie de façon unique la classe.

2-2-7-c - threadingModel

Définit le modèle de gestion des *threads* supporté par le composant. Les valeurs possibles sont : "Apartment", "Free", "Both", et "Neutral". À ne pas négliger.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-7-d - tlbid

Une chaine formatée représentant le GUID de la bibliothèque de type du composant. Essentiellement pour info.

2-2-7-e - progid

Une chaine contenant l'identifiant de la version de la classe associée au composant. Il a généralement la forme "nomComposant.nomClasse.n" où n est le numéro de version de la classe. Il est théoriquement facultatif mais en pratique indispensable avec VBScript pour instancier effectivement la classe.

2-2-7-f - miscStatus

Comme les quatre derniers attributs qui suivent, cet attribut permet de définir les données susceptibles d'être requises pour la représentation graphique du composant si celui-ci est un contrôle ActiveX visuel. Il s'agit d'une liste de valeurs numériques, séparées par une virgule, qui définit les valeurs par défaut à retenir lorsque les autres attributs ne sont pas renseignés. Ces valeurs doivent être choisies parmi l'énumération OLEMISC.

2-2-7-g - miscStatusIcon

Fournit les valeurs attendues si le contrôle est affiché comme une icône.

2-2-7-h - miscStatusContent

Fournit les valeurs attendues si le contrôle est affiché dans un container.

2-2-7-i - miscStatusDocPrint

Fournit les valeurs attendues si le contrôle est affiché comme un aperçu de son impression.

2-2-7-j - miscStatusThumbnail

Fournit les valeurs attendues si le contrôle est affiché comme une vignette pour sa représentation dans un explorateur.

2-2-8 - typelib

Cet élément précise les caractéristiques de la bibliothèque de type éventuellement présente dans le composant. Il possède cinq attributs :

2-2-8-a - tlbid (requis)

La même chaine formatée que celle figurant éventuellement comme attribut de l'élément *comClass*.

2-2-8-b - version (requis)

Une chaine "N.n" représentant la version majeure et mineure de la bibliothèque de type.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-8-c - helpdir (requis)

Une chaine qui définit le répertoire dans lequel les fichiers d'aide de la bibliothèque de type sont susceptibles d'être placés. Si le composant est multilingue, plusieurs fichiers peuvent s'y trouver. Si l'aide est inexistante, une chaine nulle est requise.

2-2-8-d - resourceid

Une chaine représentant des valeurs hexadécimales (sans préfixe 0x) au format LCID désignant l'identifiant de localisation.

2-2-8-e - flags

Une chaine représentant les flags spécifiques de la bibliothèque. Les valeurs alternatives possibles sont : "", "RESTRICTED", "CONTROL", "HIDDEN" et "HASDISKIMAGE".

2-2-9 - comInterfaceExternalProxyStub

Le composant qui constitue le serveur COM possède un mode de gestion cloisonné des *threads* que l'on définit dans l'attribut threadingModel. Le client COM possède également son propre mode de gestion qui ne sera pas nécessairement compatible avec celui du premier. Aussi, afin de garantir une gestion correcte des appels au serveur, le modèle COM a-t-il prévu une norme particulière en vue de sérialiser l'échange des données entre les différents *threads*. Ce mécanisme s'appelle le *cross-apartment marshalling* puisqu'il permet aux données de passer en bon ordre "à travers la cloison".

Lorsqu'il s'agit d'interfaces dérivées d'*IDispatch*, ce qui est le cas des composants *Automation* reconnus par VBScript, le couple "entrée/sortie" ou *proxy/stub* assurant cette sérialisation, est fourni en externe par la bibliothèque **ole32.dll** d'où le nom de l'attribut. Mais sa création suppose un certain nombre de paramètres qui seront repris dans les attributs ci-dessous. A chaque interface dérivée correspondra un élément *comInterfaceExternalProxyStub*.

2-2-9-a - iid (requis)

Une chaine au format IID désignant l'interface concernée.

2-2-9-b - baseInterface

Une chaine au format IID désignant l'interface à partir de laquelle a été dérivée l'interface concernée. En pratique il s'agit toujours de l'interface *IDispatch* soit la valeur : "{00000000-0000-0000-C000-000000000046}".

2-2-9-c - numMethods

Le nombre de méthodes implémentées par l'interface.

2-2-9-d - name

Le nom de l'interface.

2-2-9-e - tlbid

Une chaine au format GUID désignant la bibliothèque de type qui décrit cette interface.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-9-f - proxyStubClsid32

Une chaine au format GUID qui désigne la classe chargée de créer dynamiquement le serveur *proxy/stub*. Le plus souvent, ce sera le sérialiseur (*marshaller*) universel *PSOAInterface* (*Proxy Stub OLE Automation*) soit la valeur : "{00020424-0000-0000-C000-000000000046}".

2-2-10 - comInterfaceProxyStub

Cet élément répond au même besoin que l'élément précédent. La seule différence est qu'il s'applique aux interfaces personnalisées d'un composant qui ne seraient pas dérivées d'*IDispatch*. Dans ce cas, le *proxy/stub* doit être créé par la bibliothèque référencée par l'*assembly* ou par un autre fichier affecté à cet usage et désigné par l'attribut *file*. Aux attributs similaires à ceux de l'élément précédent s'ajoute un attribut *threadingModel* qui décrit le modèle de gestion spécifique de l'interface.

2-2-10-a - iid (requis)

Une chaine au format IID désignant l'interface concernée.

2-2-10-b - name (requis)

Le nom de l'interface.

2-2-10-c - tlbid

Une chaine au format GUID désignant la bibliothèque de type qui décrit cette interface.

2-2-10-d - baseInterface

Une chaine au format IID désignant l'interface à partir de laquelle a été dérivée l'interface concernée. Normalement il s'agira de l'interface *IUnknown*.

2-2-10-e - numMethods

Le nombre de méthodes implémentées par l'interface.

2-2-10-f - proxyStubClsid32

Une chaine au format GUID qui désigne la classe chargée de créer dynamiquement le serveur *proxy/stub*.

2-2-10-g - threadingModel

Définit le modèle de gestion des *threads* supporté par l'interface. Les valeurs possibles sont : "Apartment", "Free", "Both", et "Neutral".

2-2-11 - windowClass

Cet élément ne possède qu'un attribut et permet de contrôler si le nom interne de la classe de la fenêtre peut contenir le numéro de version de l'*assembly* à l'origine de son enregistrement. Il inclut le nom de la classe de fenêtre.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

2-2-11-a - versioned

Deux valeurs possibles : "yes" et "no", étant précisé que seule la deuxième valeur présente un intérêt puisque la valeur par défaut est "yes".

3 - Écriture d'un exemple

La théorie c'est bien, la pratique c'est mieux. Nous allons nous pencher sur le cas d'un composant normalement présent dans le répertoire "C:\Program Files\Fichiers communs\System\ado" de toutes les configurations à savoir le composant *MS ActiveX Data Objects* (msado15.dll v2.81.1132.0). Il fournit les objets **Connection**, **Record**, **Stream**, **Command**, **Recordset** et **Parameter**. Nous allons écrire les deux *manifests* qui permettront d'exploiter ces objets, dans des scripts exécutés en contexte WSH, sans requérir au préalable l'inscription de ce composant dans la base de registre. L'exécutable **wscript.exe**, le fichier **msado15.dll** ainsi que les deux *manifests* doivent se trouver dans le même répertoire.

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"
  version="1.0.0.0" />

  <dependency>
          <dependentAssembly>
              <assemblyIdentity
                  type="win32"
                  name="msado15.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>
```

Ce fichier, très simple, définit pour l'exécutable **wscript.exe** une dépendance, en l'occurrence l'*assembly manifest***msado15.sxs.manifest**. Nous y ajouterons une deuxième, dont la seule fonction cosmétique, est d'assurer une apparence plus moderne aux contrôles visuels qui seront éventuellement affichés par le script (thème XP).

3-2 - Assembly manifest - msado15.sxs.manifest

La vraie difficulté consiste dans l'écriture de ce fichier. Mais si vous êtes arrivés à ce stade de la lecture, votre motivation est apparemment sans faille. Nous allons donc la tester en écrivant ce fichier "à mains nues". Pour y parvenir, nous aurons quand même besoin d'un petit outil capable d'explorer les données contenues dans la base de registre ou la bibliothèque de type du composant, et c'est *MS OleViewer* qui a été choisi. Ce n'est pas le plus convivial - et c'est peu de le dire - mais il est complet et mis gracieusement à notre **disposition** par Microsoft.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

Nous ouvrons, côté gauche de l'écran, notre éditeur de texte préféré - peu importe son nom pourvu qu'il sache sauvegarder le fichier texte au format *Unicode* - et entrons les éléments suivants :

1°- l'élément *assembly* qui est identique pour chaque fichier *manifest* ;

2°- l'élément *assemblyIdentity* qui décrira le *DEF-context assembly*. L'attribut *name* correspond au nom que nous allons donner au fichier *manifest* sans son extension ;

3°- l'élément *file* avec son attribut *name* qui désignera le nom du fichier dll du composant.

L'attribut name du DEF-context doit être identique à celui défini dans l'élément dependentAssembly de l'application manifest. L'attribut langage est négligé puisque notre composant est multi-langue ainsi que publicKeyToken puisqu'il s'agit d'un private assembly.

msado15.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"
    name="msado15.sxs"
    version="1.0.0.0"
    processorArchitecture="x86" />

<file name="msado15.dll">

</file>
</assembly>
```

Passé ce stade, ouvrons côté droit de l'écran la fenêtre d'*OleViewer*. Par défaut, celui-ci nous propose d'explorer les objets inscrits dans la base de registre. Dans un monde idéal, il aurait suffi de puiser directement dans la bibliothèque de type, puisque ce composant en possède une en ressource, pour y trouver toutes les informations nécessaires à l'écriture de notre *manifest*. Heureusement, Microsoft est là pour nous rappeler que ce monde n'existe pas et, accessoirement, pour nous obliger à parcourir la base de registre à la recherche des paramètres que la fonction *DllRegisterServer* y a inscrit. C'est essentiellement les attributs *progid, threadingModel, helpdir et proxyStubClsid32* qui sont concernés même si cette dernière valeur est a priori toujours la même. Nous réglerons ce point à la fin de cette section.

Il reste néanmoins plus simple de commencer l'écriture du *manifest* à partir de cette ressource plutôt que de partir à la pêche dans la base de registre.

Cliquons donc sur la petite icône aux trois triangles rouges (ou File|View Typelib...) et sélectionnons le fichier **msado15.dll** que nous aurons copié au préalable dans le répertoire de notre choix. Une fenêtre s'ouvre sur l'ensemble des données COM documentées par la bibliothèque de type. Sélectionnons l'option *Group by type kind*, histoire de mettre un peu d'ordre dans cette ménagerie.

Le panneau droit nous donne les informations relatives à la bibliothèque que nous venons d'ouvrir. L'occasion pour nous, de commencer par l'élément *typelib* qui sera placé dans l'élément *file*. Nous mémoriserons son uuid qui nous servira pour compléter les différentes occurrences de l'attribut *tlbid*. L'attribut *version* est directement tiré de la chaine version. Les attributs *helpdir, resourceid et flags* seront récupérés ultérieurement dans la base de registre.

élément file

```
<file name="msado15.dll">

<typelib tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}"
         version="2.8"
         helpdir=""
         flags="" />
</file
```

Il s'agit maintenant d'inclure dans l'élément *file* les différentes classes exposées par le composant, soit plus précisément les éléments *comClass* correspondants.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

Sélectionner le nœud *CoClasses* va nous donner la liste de toutes les classes disponibles. Six classes correspondant au modèle objet s'affichent devant nos yeux embués par l'émotion. Nous aurions pu rencontrer des classes supplémentaires arborant un attribut *noncreatable* qui aurait signalé l'impossibilité de les instancier directement. Mais dans notre cas le compte est bon. On remarquera que ces classes présentent toutes un attribut *licensed* qui signifie que leur instanciation s'effectuera en interne avec une clé de licence.

Nous ajoutons dans l'élément *file* les six éléments *comClass* correspondant à ces classes. L'attribut *description* est tiré de la chaine *helpstring* quand elle existe sinon, comme dans notre exemple, nous choisissons son contenu. L'attribut *clsid* correspond à la chaine uuid. Les deux autres attributs ne sont accessibles qu'à partir de la base de registre et nous les définirons plus tard. Les attributs *misc* sont sans objet s'agissant d'un composant non visuel.

élément file

```
<file name="msado15.dll">

<comClass
        description="Connection"
        clsid="{00000514-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
    progid="" />

<comClass
        description="Record"
        clsid="{00000560-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
        progid="" />

<comClass
        description="Stream"
        clsid="{00000566-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
        progid="" />

<comClass
        description="Command"
        clsid="{00000507-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
        progid="" />
<comClass
        description="Recordset"
        clsid="{00000535-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
        progid="" />

<comClass
        description="Parameter"
        clsid="{0000050B-0000-0010-8000-00AA006D2EA4}"
        threadingModel=""
        progid="" />

<typelib tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}"
         version="2.8"
         helpdir=""
         flags="" />
</file>
```

À ce stade, nous pourrions nous contenter de refermer la fenêtre du *ITypeLib Viewer* et aller chercher dans la base de registre les attributs manquants. Mais ce serait négliger le cas d'un client COM présentant un modèle de gestion des *threads* incompatible avec le composant. Aussi, la prudence commande de ne pas oublier les paramètres qui permettront une sérialisation correcte des données.

Elle passe par l'écriture des éléments *comInterfaceExternalProxyStub* de chaque interface en suite de l'élément *file*. Les attributs *proxyStubClsid32* et *baseInterface* ont une valeur fixe qui a été définie ci-dessus. L'attribut *tlbid* est identique à celui utilisé pour les *comClass*.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

Passons au nœud racine précédent dénommé "Dispinterfaces" qui détaille toutes les interfaces disponibles en liaison différée (*late binding*). Nous allons déployer son arborescence pour relever les interfaces dépendantes. L'attribut *name* correspond au nom du noeud et l'*iid* est tiré de la chaine uuid. On dénombre pas moins de 28 interfaces et donc autant d'éléments *comInterfaceExternalProxyStub*. Les plus curieux auront remarqué qu'il existe des redondances et que certaines interfaces héritent, en réalité, d'interfaces plus anciennes puisque le modèle COM impose la création d'une nouvelle interface dès lors qu'on ajoute une méthode ou une propriété au modèle objet.

éléments comInterfaceExternalProxyStub

```
<comInterfaceExternalProxyStub
    name="_Collection"
    iid="{00000512-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_DynaCollection"
    iid="{00000513-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_ADO"
    iid="{00000534-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Properties"
    iid="{00000504-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Property"
    iid="{00000503-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Error"
    iid="{00000500-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Errors"
    iid="{00000501-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Command15"
    iid="{00000508-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_Connection"
    iid="{00000550-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

éléments comInterfaceExternalProxyStub

```
<comInterfaceExternalProxyStub
    name="Connection15"
    iid="{00000515-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_Recordset"
    iid="{00000556-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Recordset21"
    iid="{00000555-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Recordset20"
    iid="{0000054F-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Recordset15"
    iid="{0000050E-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Fields"
    iid="{00000564-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Fields20"
    iid="{0000054D-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Fields15"
    iid="{00000506-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Field"
    iid="{00000569-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Field20"
    iid="{0000054C-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

éléments comInterfaceExternalProxyStub

```
<comInterfaceExternalProxyStub
    name="_Parameter"
    iid="{0000050C-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Parameters"
    iid="{0000050D-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Command25"
    iid="{0000054E-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_Command"
    iid="{B08400BD-F9D1-4D02-B856-71D5DBA123E9}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="ConnectionEvents"
    iid="{00000400-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020420-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="RecordsetEvents"
    iid="{00000266-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020420-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_Record"
    iid="{00000562-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="_Stream"
    iid="{00000565-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />

<comInterfaceExternalProxyStub
    name="Field15"
    iid="{00000505-0000-0010-8000-00AA006D2EA4}"
    proxyStubClsid32="{00020424-0000-0000-C000-000000000046}"
    baseInterface="{00000000-0000-0000-C000-000000000046}"
    tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}" />
```

Le plus laborieux a été fait, passons maintenant à la touche finale.

Fermons l'explorateur de bibliothèque de type et revenons à la base de registre pour compléter nos attributs manquants. Les classes de ce composant se retrouvent dans le nœud "All Objects". Le classement des items s'opère par une chaine description et une pression successive rapide sur les lettres "adodb" nous amène presque directement sur les classes qui nous intéressent. Nous récupérons les attributs *progid, threadingModel*. Les attributs *flags* et

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

helpdir en revanche seront récupérés dans l'interface du sous-nœud suivant. Au passage, nous en profitons pour vérifier la valeur *proxyStubClsid32*.

Lorsque la chaine FLAGS est égale à zéro, l'attribut flags doit être une chaine vide et non "0". Notez également que vous pouvez retenir le ProgID au lieu du VersionIndependentProgID pour définir l'attribut progid si vous souhaitez gérer simultanément des versions différentes d'un même composant.

Pour finir, voici à quoi doit ressembler l'élément *file*.

élément file complet

```
<file name="msado15.dll">

<comClass
        description="Connection"
        clsid="{00000514-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
    progid="ADODB.Connection" />

<comClass
        description="Record"
        clsid="{00000560-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
        progid="ADODB.Record" />

<comClass
        description="Stream"
        clsid="{00000566-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
        progid="ADODB.Stream" />

<comClass
        description="Command"
        clsid="{00000507-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
        progid="ADODB.Command" />

<comClass
        description="Recordset"
        clsid="{00000535-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
        progid="ADODB.Recordset" />

<comClass
        description="Parameter"
        clsid="{0000050B-0000-0010-8000-00AA006D2EA4}"
        threadingModel="Apartment"
        progid="ADODB.Parameter" />

<typelib tlbid="{2A75196C-D9EB-4129-B803-931327F72D5C}"
         version="2.8"
         helpdir="C:\WINDOWS\HELP"
         flags="" />
</file>
```

Le tout sera sauvegardé en format *Unicode* UTF-16 sous le nom **msado15.sxs.manifest**.

3-3 - Le script testRegFreeAdo.vbs

Maintenant, rédigeons un exemple de script concret pour exploiter nos *manifests*. Pour que le test soit significatif, il convient de vérifier que le composant *MS ActiveX Data Objects* n'a pas été enregistré dans la base de registre...

Plaçons dans un répertoire une copie de **wscript.exe**, **msado15.dll**, les deux manifests. Ajoutons le script ci-dessous et nous obtiendrons l'affichage du charset par défaut du stream ADO.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

testRegFreeAdo
```
On Error Resume Next
Set oSxs = CreateObject("ADODB.Stream")
If oSxs Is Nothing Then
  Set oShell = CreateObject("WScript.Shell")
 oShell.Run "wscript.exe " & Chr(34) & WScript.ScriptFullName & Chr(34)
 WScript.Quit
End If
MsgBox oSxs.Charset
```

4 - Inclusion sous la forme d'une ressource - utilitaire mt.exe

Lorsqu'on recherche des solutions portables légères, il est toujours intéressant de réduire le nombre de fichiers de la configuration. C'est dans cet esprit que Microsoft a autorisé la présence de ces *SxS manifests* sous la forme de ressource des fichiers PE.

De cette façon, l'*application manifest* pourra être inséré comme ressource dans l'exécutable **wscript.exe** et l'*assembly manifest* dans le fichier **msado15.dll**.

Pour rester dans une tonalité très *MS friendly*, nous choisirons l'utilitaire **mt.exe** disponible dans le *Microsoft Windows Software Development Kit* (SDK). Compte tenu de ses *bugs* et de sa documentation qui est restée longtemps incomplète (**ce n'est plus le cas**), des mauvaises langues ont suggéré que cet utilitaire, à usage interne, s'était retrouvé inclus dans le SDK par accident…

Faisant fi de ces calomnies ou plutôt de ces médisances, nous allons écrire deux lignes de commande qui fusionneront ces fichiers *manifests* dans les ressources des deux binaires.

*L'ennui étant fille de la simplicité, Microsoft a eu l'idée heureuse de compliquer les règles de dénomination des assembly manifests. Lorsque ce manifest se présente sous la forme d'un fichier externe, son nom doit être distinct de celui du fichier DLL qui l'accompagne. C'est la raison pour laquelle l'exemple que nous venons d'écrire porte le nom de "msado15.sxs" au lieu de "msado15". En revanche, lorsque ce manifest a vocation à être inclus sous la forme d'une ressource de ce fichier DLL, le manifest **DOIT** porter son nom. L'application manifest doit également être mis à jour puisqu'il contient le nom des assemblies qui en dépendent.*

Application manifest
```
mt.exe -manifest wscript.exe.manifest -outputresource:wscript.exe;#1
```

Assembly manifest
```
mt.exe -manifest msado15.manifest -outputresource:msado15.dll;#1
```

(noter msado15.manifest au lieu de msado15.sxs.manifest)

Le paramètre #1 sert à souligner que la ressource doit posséder l'ID n°1. Il peut être omis, c'est la valeur par défaut retenue par **mt**.

Les concepteurs du modèle SxS avaient initialement choisi un ordre de chargement privilégiant la souplesse. En conséquence les manifests externes étaient prioritaires au détriment des ressources manifests. Depuis Windows Server 2003 et pour des considérations de sécurité faciles à comprendre, la priorité a été inversée. Pensez à en tenir compte lors de la conception de vos manifests.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

5 - Les autres solutions

La solution décrite ci-dessus est simple puisqu'elle ne nécessite que deux fichiers texte mais elle présente la particularité d'être statique dans le sens où le contexte d'activation du composant est créé et actif dès la création du processus sans possibilité de le décharger. Aussi Microsoft propose des solutions dynamiques plus souples mais qui relèvent toujours de la technologie *SxS*. Elles sont examinées à la section 5.1.

La section suivante décrit un concept intéressant mais mort-né, celui du *Window Script Component* qui consiste à écrire des composants ActiveX dans un langage de script comme VBScript ou JScript. Sa particularité est de pouvoir être géré par un *moniker*, l'enregistrement devenant dans ce cas inutile. Ce concept est très différent des *SxS assemblies* mais il peut quelquefois apporter des solutions plus simples que les *manifests*.

Enfin, la dernière section aborde une technique un peu particulière et pas du tout conforme aux canons traditionnels du modèle COM, l'émulation de la fonction *CoCreateInstance*, au travers de deux produits conçus pour l'assurer de façon générique.

5-1 - SxS dynamique

5-1-1 - API SxS Assembly

Tout langage capable d'appeler l'API SxS proposée par Microsoft est en mesure de créer, activer, désactiver et détruire des contextes d'activation, et ce à tout moment. Ce sujet qui relève de la programmation win32 n'entre pas dans le champ du présent article et ne sera pas traité ici même s'il est techniquement possible d'écrire des scripts VBS/JS exploitant cette API.

5-1-2 - Objet Microsoft.Windows.ActCtx

Cet objet est issu d'un composant ActiveX fourni par Microsoft à partir de Windows Server 2003 sous la forme d'un fichier **sxsoa.dll**. Il permet, notamment pour les langages de script, de créer dynamiquement un contexte d'activation afin d'exploiter directement un composant sans inscription.

Mais alors pourquoi ne pas l'évoquer directement dans la première section au lieu de le décrire seulement après quatre sections d'un article qu'on qualifiera charitablement d'austère ?

Trois raisons à cela :
- ce composant n'est pas fourni avec Windows XP, même si cet environnement est officiellement supporté ;
- l'activation implique que la propriété manifest doit contenir le nom du fichier *assembly manifest*, ce qui signifie que l'on n'est pas dispensé de l'aspect le plus critique de la technique, à savoir l'écriture de ce fichier xml ;
- ce composant, qui est censé résoudre les problèmes de gestion de ses semblables, doit lui-même être enregistré dans la base de registre. Cette problématique de la poule et de l'œuf ne facilitera pas la vie de l'utilisateur sans droits d'administrateur dans un environnement hostile qui lui refuse l'accès à cette base.

En outre, sa mise en œuvre n'est pas facilitée par une documentation qui semble avoir été rédigée par un stagiaire pas trop motivé qui se serait enfui avant d'avoir terminé le boulot.

La documentation rectifiée devrait plutôt ressembler à ça :

Modèle objet Microsoft.Windows.ActCtx

Propriété :

Manifest (lecture/écriture)

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

Avant d'appeler la méthode CreateObject de l'objet, il est indispensable de lui fournir au préalable les données du contexte d'activation. Il suffit pour cela d'affecter à la propriété le nom du fichier *assembly manifest* qui aura été écrit à cet effet. Cette propriété ne peut évidemment pas être en lecture seule comme l'indique la documentation...

Propriétés non documentées :

ManifestText
ManifestURL

Ces deux propriétés ont la même fonction que la propriété officielle mais avec les différences suivantes :
- ManifestText contient non pas un nom de fichier *manifest* mais une chaine contenant l'intégralité de l'*assembly manifest* qui peut ainsi être assemblé en mémoire ;
- ManifestURL contient le nom du fichier sous la forme d'une adresse de ressource URL.

Méthodes :

CreateObject(objectID[, location])
Contrairement à ce qu'indique la documentation, cette méthode renvoie bien une valeur ou plus précisément une instance de la classe que l'on aura définie dans *objectID*, c'est-à-dire son *progID*. Le paramètre facultatif location désigne le nom du serveur distant sur lequel l'objet peut être instancié à l'instar de la méthode native *CreateObject* de VBScript (omis dans la doc…).

GetObject([moniker],[progID])
Là encore, cette méthode renvoie bien une instance qui, dans ce cas précis, est celle d'une classe existante. Sa syntaxe est identique à celle existant nativement sous VBScript, le *moniker* étant alors, non pas un nom complet de fichier, mais une valeur de clé de registre (qui est fausse dans la doc, il manque le sous-répertoire SOFTWARE).

Un exemple de script affichant le charset par défaut d'un objet *ADO Stream* :

exemple objet Windows.ActCtx
```
Set oSxs = CreateObject("Microsoft.Windows.ActCtx")
oSxs.Manifest = "msado15.sxs.manifest"
Set oS = oSxs.CreateObject("ADODB.Stream")
MsgBox oS.Charset
```

5-2 - WSC et moniker

Cette technique n'a pas de point commun avec les *SxS assemblies*. Elle repose en fait sur une astuce qui consiste à s'appuyer sur un serveur COM *proxy*, le *script component runtime*, qui assure la gestion des appels COM entre le composant WSC écrit en langage script et le client.

L'intérêt du dispositif est qu'il autorise l'appel du composant WSC sans inscription préalable de celui-ci dans la base de registre. Pour un exemple détaillé, voir **ici**.

Mais le bon fonctionnement de l'ensemble repose toujours sur l'enregistrement du serveur COM effectif et est limité aux seuls composants WSC. Son examen détaillé sort donc du cadre de cet article.

5-3 - Émulation de CoCreateInstance

Certains développeurs n'ont pas attendu le bon vouloir de Microsoft pour tenter de s'affranchir des contraintes liées à cet enregistrement. La piste empruntée est celle de l'émulation de la fonction *CoCreateInstance* disponible dans l'API Win32 pour instancier un objet COM.

Comme le suggère l'intitulé, il s'agit de reproduire le comportement de la fonction en substituant aux appels à la base de registre un code équivalent capable de s'en dispenser. La démarche peut être appliquée dans le code d'un

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

langage compilé et permettre ainsi l'appel en liaison précoce (*early binding*) d'un composant ActiveX. On aborde là un problème de programmation système Windows qui déborde du sujet de l'article. Néanmoins, cette émulation peut servir de base à un *wrapper* capable d'assurer les instanciations en mode émulé. Ces *wrappers* existent et vous les trouverez dans les sections ci-dessous.

5-3-1 - Composant ActiveX Pack1 - newObjects

Ce composant *freeware* de *newObjects*, encore disponible sur le **net**, est un monument à lui tout seul. Son auteur a cessé tout développement depuis septembre 2006 mais l'**AXPack1** existe toujours et met à la disposition du développeur un invraisemblable fourre-tout où se côtoient les fonctionnalités les plus improbables. Parmi celles-ci figure la possibilité d'instancier un objet "à la carte" selon une liste impressionnante de paramètres alternatifs. La documentation décrit notamment un mode de création "non-standard" sans s'étendre plus avant sur le sujet.

Cette technique "non-standard" est bien évidemment l'émulation de *CoCreateInstance*. Elle possède d'ailleurs les inconvénients reconnus de cette technique :
- nécessite la présence d'une ressource *TypeLib* dans le fichier ;
- incompatibilité avec certains composants qui cherchent à accéder spécifiquement à la base de registre quand ils n'ont pas l'idée saugrenue de tenter de s'auto-enregistrer.

Il en existe un troisième qui à notre connaissance n'est jamais évoqué nulle part, c'est la présence requise de l'interface *IProvideClassInfo*. Nous y reviendrons plus en détail dans la section suivante.

En tout état de cause, la méthode *CreateObject* de cet objet donne entière satisfaction dans les limites énoncées ci-dessus et il serait dommage de s'en priver d'autant que la **documentation** a été rédigée avec beaucoup de soin.

5-3-2 - Bibliothèque RegFreeX.dll - omen999

Le *wrapper* peut également se présenter sous la forme d'une bibliothèque DLL classique exposant l'API nécessaire pour instancier les objets d'un composant quelconque. Du point de vue de la portabilité, c'est une solution idéale à partir du moment où le langage du client COM sait appeler ces API. Cet aspect est réglé de façon à peu près satisfaisante pour les langages *Active Scripting* sous *Windows Script Host* grâce au composant **DynamicWrapperX** dont l'intégration sans inscription faisait justement l'objet de la première partie de cet article.

Pour être tout à fait complet, le sujet de cette section devait porter sur une bibliothèque *freeware***DirectCOM** disponible sur le *net*, écrite par son auteur pour faciliter l'installation de ses produits écrits en VB6.

Malheureusement, les conventions d'appel de cette bibliothèque sont incompatibles avec **DynamicWrapperX**. Il a donc fallu écrire un *wrapper* capable de s'accommoder des limites de ce dernier. Cette bibliothèque **RegFreeX** est disponible **ici** avec sa documentation.

L'étude en détail de cette émulation a d'ailleurs permis de relever, lorsqu'elle intervient en mode liaison différée, l'existence d'une condition supplémentaire jusque-là inconnue. En effet, si le composant n'expose pas l'interface *IProvideClassInfo* (c'est le cas de MS ADO), l'appel des méthodes et des propriétés de l'objet échoue. Le composant **AXPack1** comme **DirectCOM** ne proposent pas de solution pour résoudre ce problème.

En revanche, **RegFreeX** supporte un mode dégradé qu'on pourrait qualifier de "liaison très différée" qui autorise, par le biais d'une méthode *CallByNameEx*, l'appel de chaque méthode ou propriété de l'objet instancié quand le composant ne fournit pas d'interface *IProvideClassInfo*.

6 - Liens

Le **fichier archive** contenant les fichiers *manifests* donnés en exemple et le script *testRegFreeAdo.vbs*.
Documentation mt.exe
Bibliothèque RegFreeX

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.

7 - Remerciements

Je tiens à remercier **bbil**, **ThierryAIM** et **_Max_** pour leur relecture attentive et leurs observations pertinentes.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2011 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.