e i g
Ecole d'ingénieurs de Genève

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
University of Applied Sciences Western Switzerland

Gérer *Vista*

Unité d'enseignement et de recherche UER4

Filière :
Télécommunications

Laboratoire :
Transmission de données

Étudiant :
TAVARES José

Classe :
TE3

Session :
2006

Professeur responsable
LITZISTORF Gérald

En collaboration avec :
Lionel CAU

e i g

Ecole d'ingénieurs
de Genève

Automne 2006

Session de diplôme

SECURITE DES SYSTEMES D'INFORMATION
GERER VISTA

Descriptif :

La sécurisation du poste de travail (intégrité, disponibilité, confidentialité) constitue une tache particulièrement complexe pour les entreprises qui veulent limiter les risques des PCs des utilisateurs.

Ce travail a pour objectifs :

- de maîtriser le processus de création des images Vista en visant notamment une réduction du nombre de composants,
- d'acquérir une expertise dans l'utilisation du langage de script et du *shell* interactif de PowerShell, disponible pour les systèmes Windows XP et Vista, afin d'automatiser certaines tâches d'administration telles que la configurations des paramètres du système,
- de tester les nouvelles fonctions liées à la gestions des *logs*.

Travail demandé :

Ce travail se compose des parties suivantes :

1) Etude théorique du format WIM, de la commande Ximage, de WIMGAPI et de Windows PE
   Mise en oeuvre
   Identifier les principaux composants et *packages*
   Familiarisation de l'outil *System Image Manager*
   Définir un scénario (besoin) d'entreprise et le mettre en œuvre
   Etudier l'application des correctifs pour des composants absents

2) Etude théorique de PowerShell (architecture, syntaxe, possibilités, ...)
   Identifier des fonctions intéressantes pour le labo (*forensics*, ...)
   Mettre en œuvre

3) Etude théorique de la gestion des logs sous Vista
   Nouvel outil MMC *Event Viewer*, filtrage et actions gérées par *Task Scheduler*
   API

Sous réserve de modifications en cours de travail.

Unité d'enseignement
Et de recherche UER4

Classe : TE3

Timbre de l'Ecole

ÉCOLE D'INGÉNIEURS * RÉPUBLIQUE ET CANTON DE GENÈVE * DIRECTION

Candidat :
M. TAVARES JOSE
Filière d'études :
Télécommunication
Travail de diplôme soumis à une convention de stage en entreprise : non
Travail de diplôme soumis à un contrat de confidentialité : non

Professeur(s) responsable(s) :
Litzistorf Gérald

En collaboration avec : LANexpert SA

Table des matières

Préambule

Ce document a été élaboré lors de mon projet de diplôme de la session 2006.

Ce mémoire est divisé en 3 parties pouvant être résumées comme suit :

La première partie concerne tout ce qui est images *Vista* et déploiement, comment créer, gérer, installer des images, comment déployer les images, quel sont les outils disponibles et les moyens mis en œuvre.

La seconde partie concerne le nouveau *shell* nommé *PowerShell*, qu'est ce qu'il est possible de faire avec ce nouveau *shell*, a-t-il une utilisé en ce qui concerne la première partie de ce diplôme ? Quels sont les scénarios intéressants ?

La troisième et dernière partie concerne la gestion des *logs* sous *Windows Vista*, la nouvelle version de l'outil *MMC Event Viewer* disponible dans *Windows Vista* sera étudiée, je montrerai aussi comment centraliser les *logs* des divers postes *Vista* présents sur le réseau.

De plus pour faciliter la lecture ce de mémoire de semestre, certaines conventions typographiques ont été adoptées :

- Verdana 10 normal : texte normal
- **Verdana 10 gras** : phrases ou termes importants
- *Verdana 10 italique* : termes anglais
- Verdana 8 normal : légendes
- **`Courrier 10 gras`** : commandes, noms de fichiers/dossiers

Je tiens aussi à remercier les personnes suivantes qui m'ont aidé lors de ce projet :

- M. LITZISTORF Gérald pour ses remarques et conseils, ainsi que pour la qualité de son enseignement.
- M. CONTRERAS Sébastien et M. SADEG Nicolas pour leur aide.
- M. CAU Lionel pour m'avoir offert la possibilité d'étudier la problématique liée à ce sujet ainsi que pour son aide.

Les systèmes d'exploitation utilisés sont les suivants :

- *Windows XP SP2*
- *Windows Vista RC1 Build 5600 Version Ultimate*, sortie le 5 Septembre 2006
- *Windows Vista RC2 Build 5744 Version Ultimate*, sortie le 7 Octobre 2006

Les versions finales de *Windows Vista* seront normalement disponibles autour du 30 Novembre 2006 pour les entreprises, et autour du 30 Janvier 2007 pour les particuliers.

Les versions RC et Beta de *Windows Vista* cesseront de fonctionner le 1er juin 2007, elles ne sont d'ailleurs plus disponibles au téléchargement sur le site de *Microsoft*.

Voici un tableau résumant les différentes versions de *Windows Vista* proposées ainsi qu'une estimation de leur prix pour l'Europe (selon www.clubic.com, ayant comme sources Amazon Allemagne) :

	Prix européens	Prix américains
Vista Home Basic	259 euros	199 dollars (~155 euros)
Vista Home Premium	329 euros	239 dollars (~186 euros)
Vista Business	349 euros	299 dollars (~233 euros)
Vista Ultimate	549 euros	399 dollars (~311 euros)

Remarque : Les prix européens sont beaucoup plus chers que les prix américains, éventuellement commander des licences en Amérique ?? (Étant donné que *Vista* est indépendant du langage, il suffit de rajouter un *package* contenant la langue française pour avoir un *Windows Vista* en français !!

Configuration minimum pour *Vista* (d'après *Microsoft*) :

- Processeur >=800 MHz
- 512MB RAM
- Disque dur 20GB

Configuration conseillée pouvant utiliser toutes les options disponibles avec ***Vista*** (d'après *Microsoft*) :

- Processeur 1 GHz
- 1024MB RAM
- Carte graphique supportant *DirectX 9* avec un driver *WDDM* (*Windows Display Driver Model*), 128MB de mémoire graphique afin de pouvoir utiliser *Aero*, *Pixel Shader 2.0* et 32 bits par pixel
- Disque dur 40GB avec 15GB d'espace libre

1 Images *Vista*

7 semaines d'étude

1.1 Introduction

Dans une entreprise, l'installation de systèmes d'exploitation sur plusieurs ordinateurs est une tâche qui peut prendre énormément de temps aux ingénieurs système.

Par exemple, si cet ingénieur doit installer chaque poste de travail individuellement avec un DVD, effectuer les installations traditionnellement en suivant toutes les étapes par défaut puis personnaliser l'installation, cela demandera énormément de temps. De plus il lui faudra se déplacer dans l'entreprise pour installer chaque ordinateur.

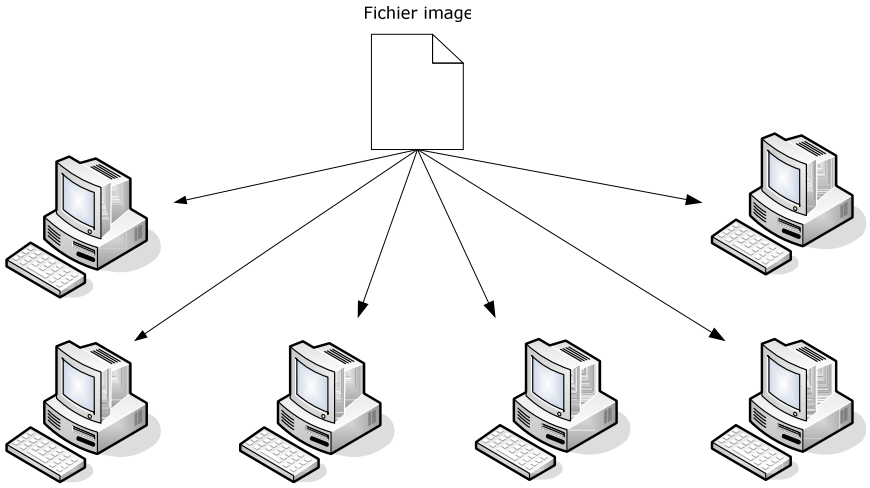
Le but de cette première partie de mon diplôme, est d'analyser les différentes possibilités qu'offre *Windows Vista* avec les nouveaux outils conçus spécialement par *Microsoft* afin de gérer les images *Vista*, automatiser les installations et faciliter leur déploiement (voir chapitre suivant pour la définition d'un déploiement).

Ceci permettra à un ingénieur système de gagner du temps et donc l'entreprise fera des économies, sans compter le fait que les installations se feront bien plus rapidement.

1.2 Déploiement

1.2.1 Qu'est ce que c'est ?

Le déploiement d'une image consiste à transmettre et installer une même image sur plusieurs ordinateurs.

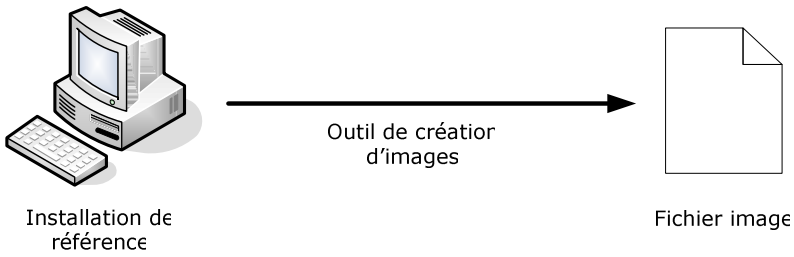


1.2.2 Qu'est ce qu'une image ?

Une image au sens "image système", est une copie bit à bit intégrale (éventuellement avec compression) de l'information numérique présente sur un support d'information.

Dans notre cas ce support d'information sera un disque dur (éventuellement une partition) où le système d'exploitation a été installé.

Ceci peut se faire à l'aide d'un outil spécialement créé à cet effet, qui créera une image sur un ou plusieurs fichiers (dépendant de l'outil et du format d'image qui lui est associé).



1.2.3 Format d'image *WIM* dans *Vista*

Le format d'image dans *Windows Vista* est le format ***WIM*** (*Windows Imaging*).

Lorsqu'une image *WIM* est créée, elle est stockée en 1 seul et unique fichier, un peu comme un fichier ZIP compressé.

Ce nouveau format *WIM* n'est plus basé sur les secteurs du disque dur mais sur les fichiers qui y sont contenus !

En copiant les **secteurs disque**, il n'y a aucun moyen de savoir à quel fichier appartient un bit copié.
En installant une image basée sur secteurs disque, les différents secteurs de disque contenus dans l'image seront simplement recopiés sur le disque dur, sans se soucier de quels fichiers les différents bits représentent.

En copiant les **fichiers** (copie bit à bit) contenus sur le disque, on sait exactement ou se trouvent les bits sur le disque dur d'un fichier à copier. Ces fichiers seront tout d'abord compressés puis copiés vers notre image afin de représenter notre fichier.
En installant une image basée sur les fichiers, ces fichiers seront simplement copiés vers le disque dur, peut importe leur emplacement sur les secteurs disque.

Remarque : Les fichiers sont copiés bit à bit et compressés, ce n'est pas le disque dur qui est copié bit à bit !

Le format *WIM* présente les caractéristiques suivantes :

- **Indépendant du *Hardware*** (configuration matérielle).
  Jusqu'à présent il fallait effectuer une image pour chaque configuration *Hardware*, maintenant avec le nouveau format *WIM,* une image peut être installée sur n'importe quelle configuration *hardware*.

  Pour tester ceci, différentes images ont été créées :

  1) Installation capturée d'ordinateur portable *Dell Inspiron 8600* ayant les caractéristiques suivantes :

  - Processeur Intel Pentium M 1.6GHz
  - 768MB RAM
  - Carte graphique ATI Radeon 9600PRO 128DDR
  - Disque dur IDE FUJITSU 60GB 5400tours/minute
  - Graveur DVD+RW NEC ND-6100A

  Puis installée sur un pc *Dell Optiplex GX260* ayant les caractéristiques suivantes :

  - Processeur Intel Pentium 4 2.8GHz
  - 1024MB RAM
  - Carte graphique incorporée sur carte mère
  - Disque dur IDE 7200tours/minute
  - Lecteur DVD HL-DT-ST GDR8082N

  2) Installation capturée d'ordinateur sur mesure ayant comme caractéristiques :

  - Processeur Intel Pentium 4 2.6 GHz
  - 1024MB RAM
  - carte graphique Nvidia 6600GT 128DDR
  - Disque dur S-ATA Seagate Barracuda 200GB 7200tours/minute
  - Disque dur S-ATA Western Digital 300GB 7200tours/minute
  - Graveur DVD Plextor PX-755A

  Puis installée sur un pc portable *Dell Inspiron 8600* ayant les caractéristiques définies lors du premier test.

  Aucun problème constaté lors de ces divers tests.

- **Plusieurs images** peuvent être **contenues dans 1 seul fichier**.
  Imaginons le cas d'une entreprise avec des ingénieurs système et des secrétaires. Les ingénieurs système ont besoin de programmes spécifiques pour effectuer leur travail, tandis que les secrétaires ont juste besoin d'un éditeur de texte.
  Il n'est pas nécessaire d'effectuer 2 images différentes d'environ 3 gigabytes chacune.
  Il suffit d'incorporer ces 2 images dans le même fichier *WIM*.
  A l'intérieur d'un fichier *WIM*, il y a une seule instance pour chaque fichier contenu dans les différentes images, ce qui ne doublera pas la taille de notre image *WIM* ! (voir point suivant).
  Les différentes étapes nécessaires pour incorporer diverses images à l'intérieur d'un même fichier *WIM* sont décrites dans le paragraphe 1.5.5.2.2 page 31.

- **Une seule instance pour chaque fichier**.
  Par exemple on effectue une image de notre système que l'on nomme *image.wim*.
  Nous ajoutons un programme de 100Mb à notre système puis effectuons une nouvelle image de notre système sur le même fichier (*image.wim*).
  La taille de ce fichier n'aura pas doublé, elle aura juste augmenté de 100Mb et sera composée de 2 images !
  L'intérêt étant d'effectuer plusieurs images à l'intérieur du même fichier *WIM* (par exemple une image pour les ingénieurs système, une autre pour les secrétaires, un autre pour les développeurs, ce scénario est expliqué en détail au paragraphe 1.5.2 page 18).
  Ceci permet d'occuper nettement moins d'espace de stockage pour les différentes images et de les gérer plus facilement.

- **Les fichiers sont compressés**.
  Une compression est utilisée pour diminuer l'espace de stockage nécessaire à un fichier *WIM*.
  Il existe 2 modes de compression : *XPress* et *LZX*.
  Le mode par défaut est *XPress*.
  Le mode *LZX* offre une meilleure compression, cependant le temps nécessaire à la création d'image ainsi que l'installation sera plus élevé.
  Les différents modes de compression sont testés dans le paragraphe 1.8.3 page 72.

- Possibilité de **gérer une image en mode *Offline***.
  Le mode *Offline* signifie qu'il n'est pas nécessaire de charger une image pour pouvoir la modifier.
  Il est possible de modifier une image (ajout de *drivers* (pilotes), *packages*) sans avoir besoin d'installer cette image.
  Selon *Microsoft*, tous les prochains *services packs* pour *Windows Vista* seront distribués en tant que *packages*, il sera alors très facile de les incorporer à une image (voir en annexes).

- Possibilité d'installer une image disque sur des partitions (ou disques) de taille différentes (pour autant qu'il y ait suffisamment d'espace disponible).

- API pour le format *WIM* nommée WIMGAPI.
  Cette API permet aux développeurs de créer un programme spécifique à leurs besoins.
  Pour le moment cette API reste une publicité de *Microsoft*, il n'y a pas beaucoup de personnes qui développent des programmes utilisant cette API.

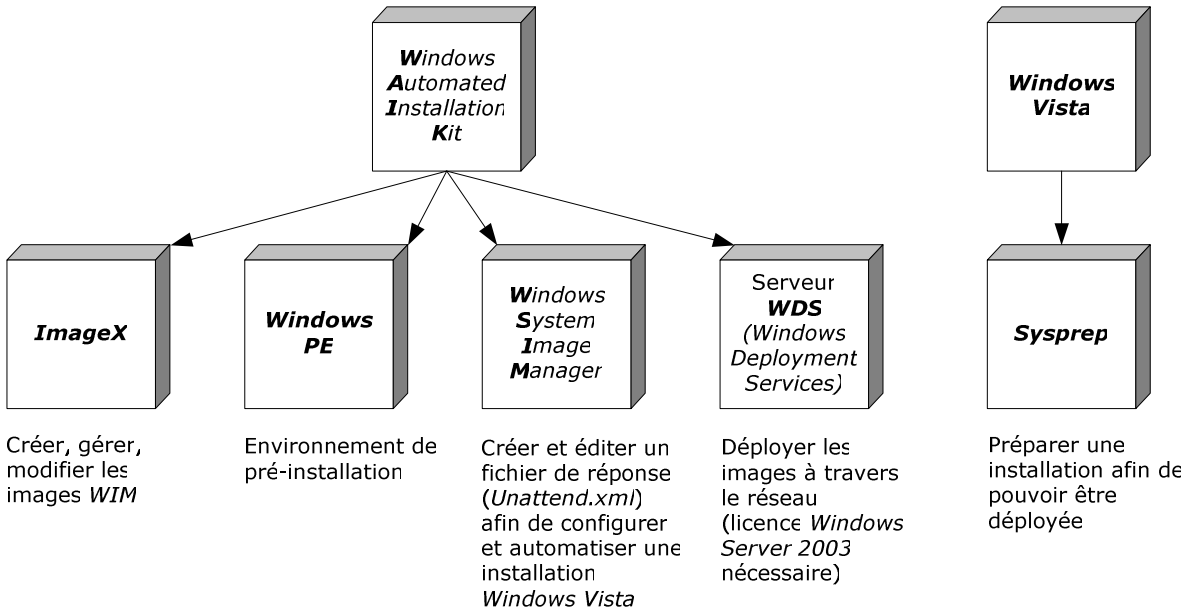
- **Installation d'images non destructive**.
  L'installation d'une image sur un disque n'efface pas les données présentes sur ce disque.

Remarque : Une image *WIM* peut être gérée aussi bien sur *Windows Vista* que sur *Windows PE*, *Windows XP* Professionnel ou *Windows Server 2003*.

Il est aussi possible de créer des images *WIM* d'autres systèmes d'exploitation que *Windows Vista*, cependant le format *WIM* a été spécialement conçu pour *Vista*, permettant tous les points énumérés ci-dessus.
Si une image *WIM* est créée par exemple pour *Windows XP*, cette image ne sera bien évidemment pas déployable sur d'autres configurations *hardware* (ceci étant propre à *Vista*).

1.3 Outils disponibles

Voici les différents outils qui seront utilisés dans ce diplôme :



Pourquoi utiliser ces outils plutôt que d'autres ?

Ces outils sont gratuits et peuvent être téléchargés sur le site de *Microsoft*, mis à part *Sysprep* qui est installé par défaut lors d'une installation *Windows*.
Ils ont été élaborés pour être utilisés avec le nouveau format d'image *WIM*.

Tout au long de cette première partie de mon diplôme, ces divers outils vont être utilisés et testés dans divers scénarios.

Une étude détaillée est disponible pour :

- *Windows Automated Installation Kit* dans le paragraphe suivant
- *ImageX* en page 71, paragraphe 1.8
- *Windows PE* en page 59, paragraphe 1.7
- *Windows System Image Manager* en annexes
- Serveur *WDS* en page 87, paragraphe 1.11
- *Sysprep* en page 74 paragraphe 1.9

1.4 Windows Automated Installation Kit

1.4.1 Qu'est ce que c'est ?

Windows Automated Installation Kit (abrégé "*Windows AIK*" ou "*WAIK*") regroupe plusieurs outils ainsi que l'aide qui leur est associée.

Ces outils (comme indiqué sur la figure ci dessus) sont utilisés pour automatiser une installation *Vista*, gérer les différentes images, pouvoir déployer les images, etc.

Le but étant de pouvoir gérer le plus simplement possible les images *WIM*, automatiser le plus possible une installation et effectuer facilement des déploiements.

1.4.2 Comment installer *Windows AIK* ?

Windows AIK n'est a ce jour disponible que dans *Business Desktop Deployment 2007* (abrégé "*BDD 2007*") qui peut être téléchargé à l'adresse :

http://www.microsoft.com/technet/desktopdeployment/bdd/2007/default.mspx

Installer *BDD 2007*, puis lancer l'installation de *WAIK* qui se trouve par défaut dans **`C:\Program Files\BDD 2007\WAIK\waikx86.msi`**

Cette version est une version pour les processeurs 32 bits. Il existe aussi 2 autres versions, une pour les processeurs 64 bits (waika64.msi) et une version spéciale pour AMD 64 bits (waikamd64.msi).

Remarques :

- *Windows AIK* peut seulement être installé sur *Windows XP SP2*, *Windows Server 2003 ou Windows Vista RC1/RC2.*

- Pour installer *BDD 2007* sur *Windows XP*, il faut avoir *.NET Framework 2.0* installé, il peut être téléchargé à l'adresse : http://www.microsoft.com/downloads/details.aspx?FamilyId=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=fr

- Pour installer *Windows AIK* sur *Windows XP*, il faut avoir *MSXML 6.0* installé, il peut être téléchargé à l'adresse : http://www.microsoft.com/downloads/details.aspx?FamilyID=993C0BCF-3BCF-4009-BE21-27E85E1857B1&displaylang=fr

1.5 Scénarios

Les différents scénarios vont être étudiés.

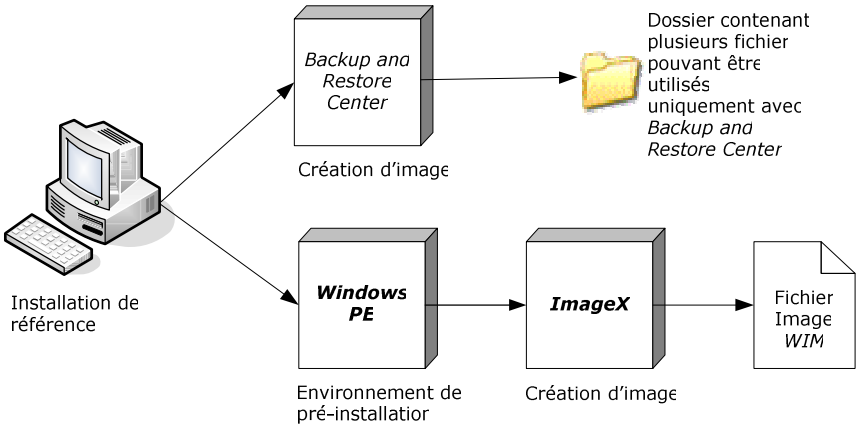
L'estimation de temps nécessaire aux différentes étapes à été faite en fonction de la configuration *hardware* (matérielle) suivante :

- Processeur Intel Pentium 4 2.8GHz FSB800 1MB cache
- 1024RAM DDR 400MHz
- Disque dur IDE 7200 tours/minute

Conseil : Il est conseillé de "s*yspreper*" toutes les installations avant d'en créer une image car *Windows Vista* se bloque s'il n'est pas activé dans les 14 jours et sera difficile à activer.

1.5.1 Scénario 1

Un **utilisateur à domicile** souhaite faire une image de son installation PC.
Il aimerait installer *Windows Vista*, ajouter des programmes, configurer ses options réseau, ses périphériques puis en créer une image afin de pouvoir réinstaller rapidement son pc en cas de problèmes.



Différentes étapes à effectuer et temps nécessaire (approximatif) :

Solution 1 ***Backup and Restore Center***

- Utilisation du programme afin de créer une image (15minutes)

Solution 2 ***Windows PE* et *ImageX***

- Créer un CD *bootable* de *Windows PE* personnalisé avec *ImageX* (15 à 20minutes)
- Démarrer sur *Windows PE* (2 minutes)
- Créer une image de notre *Vista* personnalisé à l'aide d'*ImageX* (20 minutes avec une compression rapide)
- Installer l'image de notre *Vista* personnalisé (10 minutes)

  Remarque : Si le pc sur lequel on souhaite installer notre image *Vista* personnalisé comporte déjà un système d'exploitation, et que nous désirons formater la partition de ce système, il faudra ajouter le temps nécessaire à ce formatage, qui dépend directement de la taille (en *gigabytes*) de cette partition.

1.5.2 Scénario 2

Afin de pouvoir installer un serveur *WDS*, il est nécessaire de disposer d'une licence *Windows Server 2003*. Or, toutes les entreprises ne disposent pas d'une telle licence, notamment les petites entreprises.

Nous pouvons alors distinguer 2 autres scénarios intéressants :

- Une entreprise ne disposant pas de licence *Windows Server 2003* et ne pouvant donc pas installer de serveur *WDS*.

- Une entreprise disposant d'une licence *Windows Server 2003* et pouvant donc installer un serveur *WDS*.

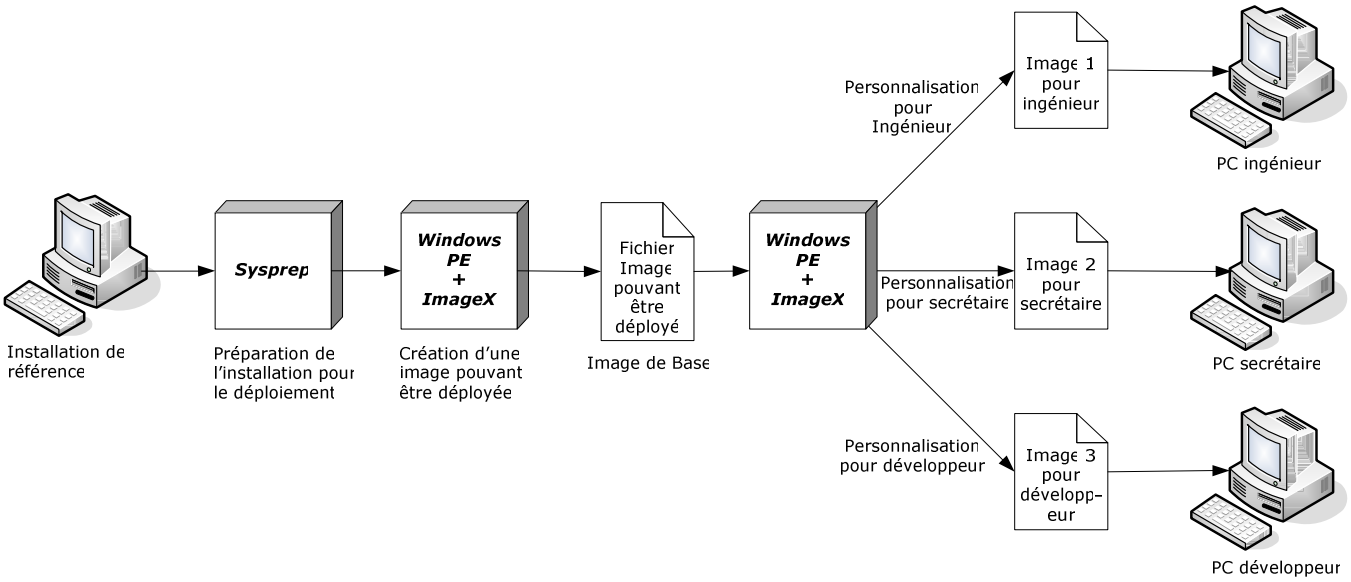
Nous allons commencer par étudier le scénario d'une **petite entreprise ne pouvant pas installer de serveur *WDS*** et souhaitant déployer une image.

Cette entreprise possède plusieurs ordinateurs de configurations *hardware* (matérielles) différentes et aimerait simplifier la gestion de ses images.

Dans cette entreprise, il y a un ingénieur réseau, une secrétaire et un développeur. L'ingénieur réseau a besoin de programmes spécifiques pour son travail, la secrétaire a besoin d'un éditeur de texte uniquement, et le développeur a quant à lui besoin de programmes de développement.

L'idéal serait donc d'avoir au moins 3 images.

Pour simplifier le diagramme suivant, les blocs *Windows PE* et *ImageX* ont étés regroupés (il faut d'abord *booter* sur *Windows PE* avant d'utiliser *ImageX*).



Différentes étapes à effectuer et temps nécessaire (approximatif) :

- Capturer une installation de base de *Windows Vista* (20minutes avec une compression rapide)
- Créer et capturer une installation personnalisée pour un ingénieur système (dépendant du nombre de *softwares* à installer)
- Créer et capturer une installation personnalisée pour une secrétaire (dépendant du nombre de *softwares* à installer)
- Créer et capturer une installation personnalisée pour un développeur (dépendant du nombre de *softwares* à installer)
- Installer une des images créées sans fichier *Unattend.xml* (10minutes)
- Installer une des images créées avec fichier *Unattend.xml* (10minutes)

Remarque : Pour simplifier la création des différentes images (ingénieur, secrétaire, développeur) et pour éviter de devoir réinstaller à chaque fois le système d'exploitation, j'ai créé au laboratoire uniquement un fichier texte sur le bureau (nommé ingénieur.txt, secretaire.txt ou développeur.txt) en fonction de l'image à créer.
Le principe est exactement identique à celui d'installation de programmes.
Le temps nécessaire à la création d'une installation personnalisée sera donc quasi nul (le temps de créer un fichier texte), le temps de capture d'installation sera d'environ 15minutes.

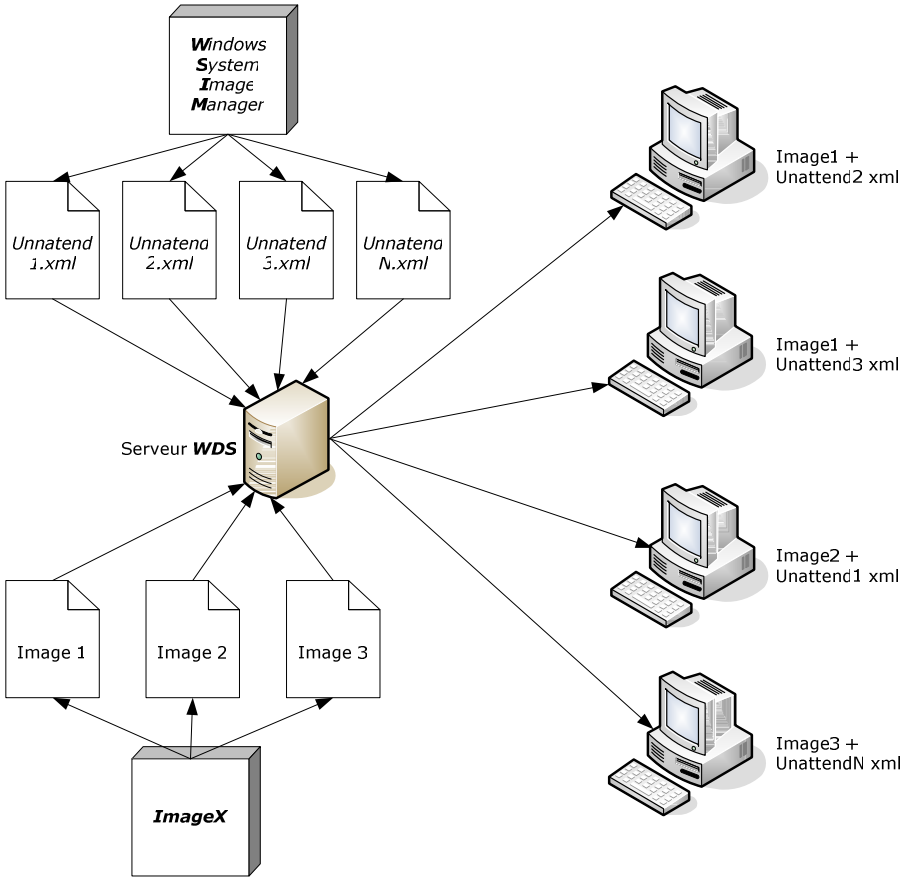
1.5.3 Scénario 3

Afin de pouvoir installer un serveur *WDS*, il est nécessaire de disposer d'une licence *Windows Server 2003*.

Nous allons donc terminer par étudier le scénario d'une **grande entreprise disposant d'une licence *Windows Server 2003* afin d'installer un serveur *WDS***.

Cette grande entreprise possède plusieurs ordinateurs de configurations *hardware* différentes et aimerait simplifier la gestion de ses images, déployer ces images à travers son réseau.

De plus il faudrait que les installations soient le plus automatisées possibles afin de demander un minimum d'interactions humaines.



Différentes étapes à effectuer et temps nécessaire (approximatif) :

- Installer un serveur *WDS* (*Windows Deployment Services*) (2minutes, sans compter les pré-requis tels qu'installer un serveur *DHCP*, *Active Directory* et autres)
- Configurer le serveur *WDS* (8minutes)
- Installer des images de *boot* sur le serveur *WDS* (10minutes)
- Installer des images d'installation (*Vista*) sur le serveur *WDS* (9minutes)
- Automatiser une installation avec un fichier *Unattend.xml* (20minutes)
- Personnaliser une installation avec un fichier *Unattend.xml* (10minutes)
- Installer une image sur un pc distant en utilisant notre serveur *WDS* (15minutes)

1.5.4 Manipulations pour scénario 1

« Un **utilisateur à domicile** souhaite faire une image de son installation PC.
Il aimerait installer *Windows Vista*, ajouter des programmes, configurer ses options réseau, ses périphériques puis en créer une image afin de pouvoir réinstaller rapidement son pc en cas de problèmes. »

Pour ce scénario, il y a 2 solutions possibles :

- Utiliser le programme ***Backup and Restore Center***, par défaut dans *Windows Vista*.

- Créer une **image *WIM*** et pouvoir installer cette image

1.5.4.1 Comparaison des 2 solutions possibles

Le tableau suivant compare les 2 solutions qui seront détaillées ci-après :

	ImageX	*Backup and Restore Center*
Choix de la partition à partir de laquelle créer une image	Oui	Non
Installation possible sur d'autres ordinateurs (déploiement)	Oui	Non
Possibilité de mettre à jour une image	Oui	Oui
Choix de la compression	Oui	Non
Taille de l'image créée	3.28 Gigabytes (avec compression rapide)	6.73 Gigabytes
Temps nécessaire à la création de l'image	20minutes (avec compression rapide)	15minutes
Possibilité de réinstallation sans avoir besoin de *booter* sur *Windows Vista*	Oui, avec *Windows PE*	Oui, avec le DVD d'installation de *Windows Vista*

1.5.4.2 Outils nécessaires pour la première solution

Solution1 : « Utiliser le programme ***Backup and Restore Center***, par défaut dans *Windows Vista*. »

- Une installation de *Windows Vista* traditionnelle, avec *Backup and Restore Center.*

1.5.4.3 Opérations à effectuer pour la première solution

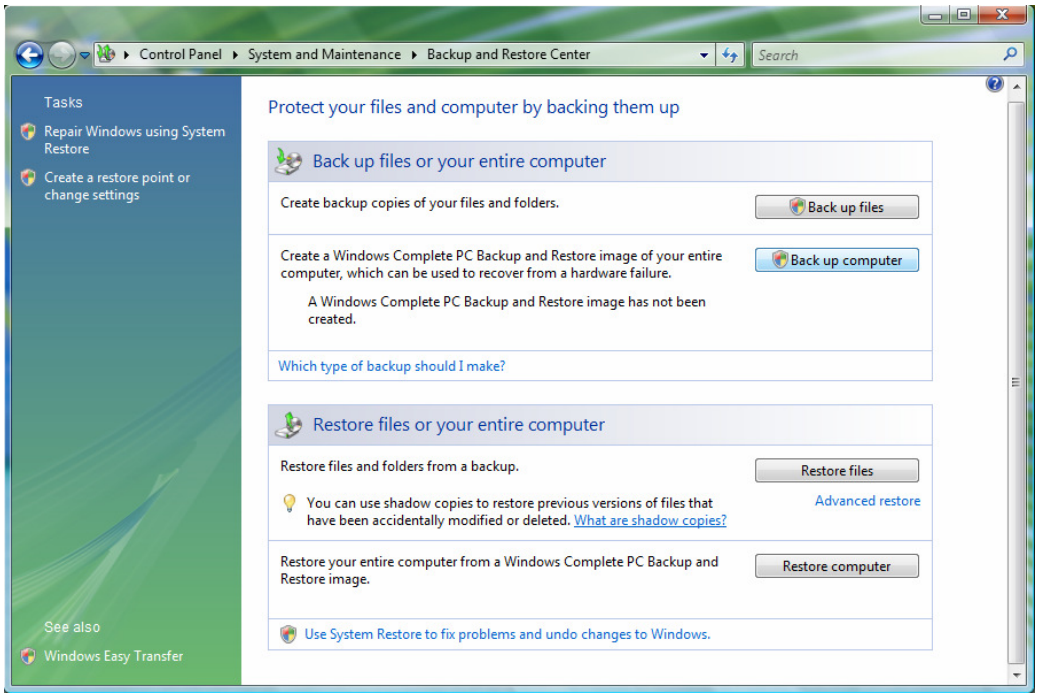
Solution1 : « Utiliser le programme ***Backup and Restore Center***, par défaut dans *Windows Vista*. »

Ce programme nous permet de faire 2 choses :

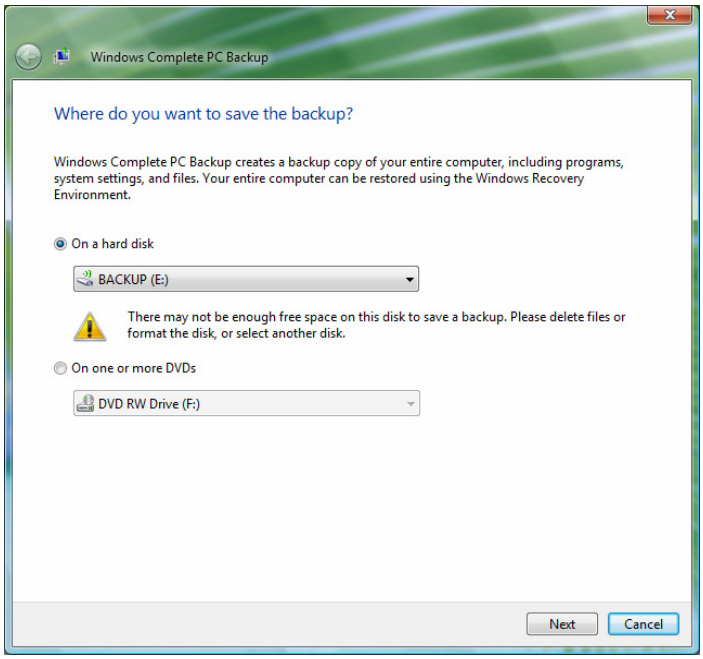
- Un *backup* complet de notre pc
- Un *backup* uniquement de fichiers personnels (fichiers audio, vidéo, images), cette option ne sera pas traitée dans ce document car elle ne nous intéresse pas vraiment.

Le terme "*backup*" ici désigne une sauvegarde.

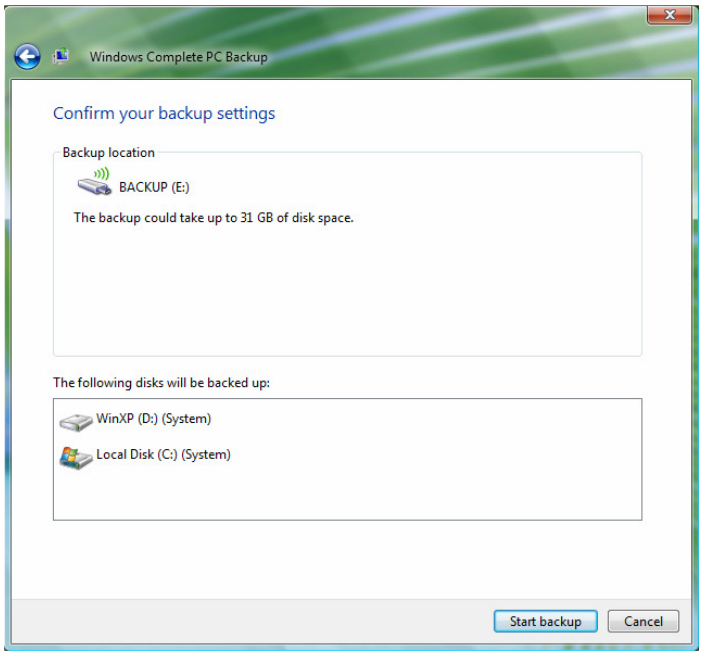
Pour effectuer un *backup* entier du pc, ouvrir le ***Backup and Restore Center***.



Sélectionner le bouton *Back Up Computer.*



Sélectionner l'endroit vers lequel le *backup* sera stocké.
On peut choisir d'y stocker sur le disque dur (sur une partition) ou d'y graver directement sur un ou plusieurs DVD.



Confirmer l'endroit de destination choisi pour sauvegarder notre *backup*.

Le pc utilisé comportait aussi une installation *Windows XP*, nous remarquons que ce programme effectue aussi un *backup* de cette partition, il n'est pas possible de choisir les partitions pour le *backup*, ce programme prend toutes les partitions système.

Sélectionner le bouton *Start Backup*, le programme effectuera alors les opérations demandées. Cet outil crée un dossier y contenant plusieurs fichiers pouvant être utilisés (à des fins de restauration) uniquement avec *Backup and Restore Center*. Ces différents fichiers n'ont pas été étudiés du fait que ce programme a été conçu spécialement pour les utilisateurs à domicile, il n'est pas intéressant dans le cadre de ce projet.

Pour réinstaller une image créée avec cet outil, il faut *booter* sur le DVD d'installation de *Windows Vista*, puis sélectionner *Repair your Computer*.

1.5.4.4 Outils nécessaires pour la deuxième solution

Solution2 : « Créer une **image *WIM*** et pouvoir installer cette image »

- Avoir une installation de *Windows AIK*
- *ImageX* afin de créer et installer une image *WIM*.
- *Windows PE*, en créer un CD *bootable* et y inclure *ImageX*.
- *OSCDIMG*, afin de créer une ISO *bootable* de notre *Windows PE*, cet outil est disponible par défaut dans *Windows PE*.
- Un programme afin de graver l'image *bootable* de *Windows PE* (image ISO).
- DVD de base *Windows Vista*

1.5.4.5 Opérations à effectuer pour la deuxième solution

Solution2 : « Créer une **image *WIM*** et pouvoir installer cette image »

Une étude de *Windows PE* est disponible en page 59, paragraphe 1.7
Une étude d'*ImageX* est disponible en page 71, paragraphe 1.8

Dans ce chapitre nous allons :

- Créer un CD *bootable* de *Windows PE* personnalisé avec *ImageX* (10 à 15minutes)
- Démarrer sur *Windows PE* (2 minutes)
- Créer une image de notre *Vista* personnalisé à l'aide d'*ImageX* (20 minutes avec une compression rapide)
- Installer l'image de notre *Vista* personnalisé (10 minutes)

Remarque : Si le pc sur lequel on veut installer notre image *Vista* personnalisé comporte déjà un système d'exploitation, et que nous désirons formater la partition de ce système, il faudra ajouter le temps nécessaire à ce formatage, qui dépend directement de la taille (en *gigabytes*) de cette partition.

1.5.4.5.1 Créer un CD *bootable* de *Windows PE* personnalisé avec *ImageX*

Pour créer une image de notre *Vista* personnalisé, il faut pouvoir démarrer l'ordinateur avec *Windows PE* ou sur un autre système d'exploitation, puis capturer l'image de notre installation.

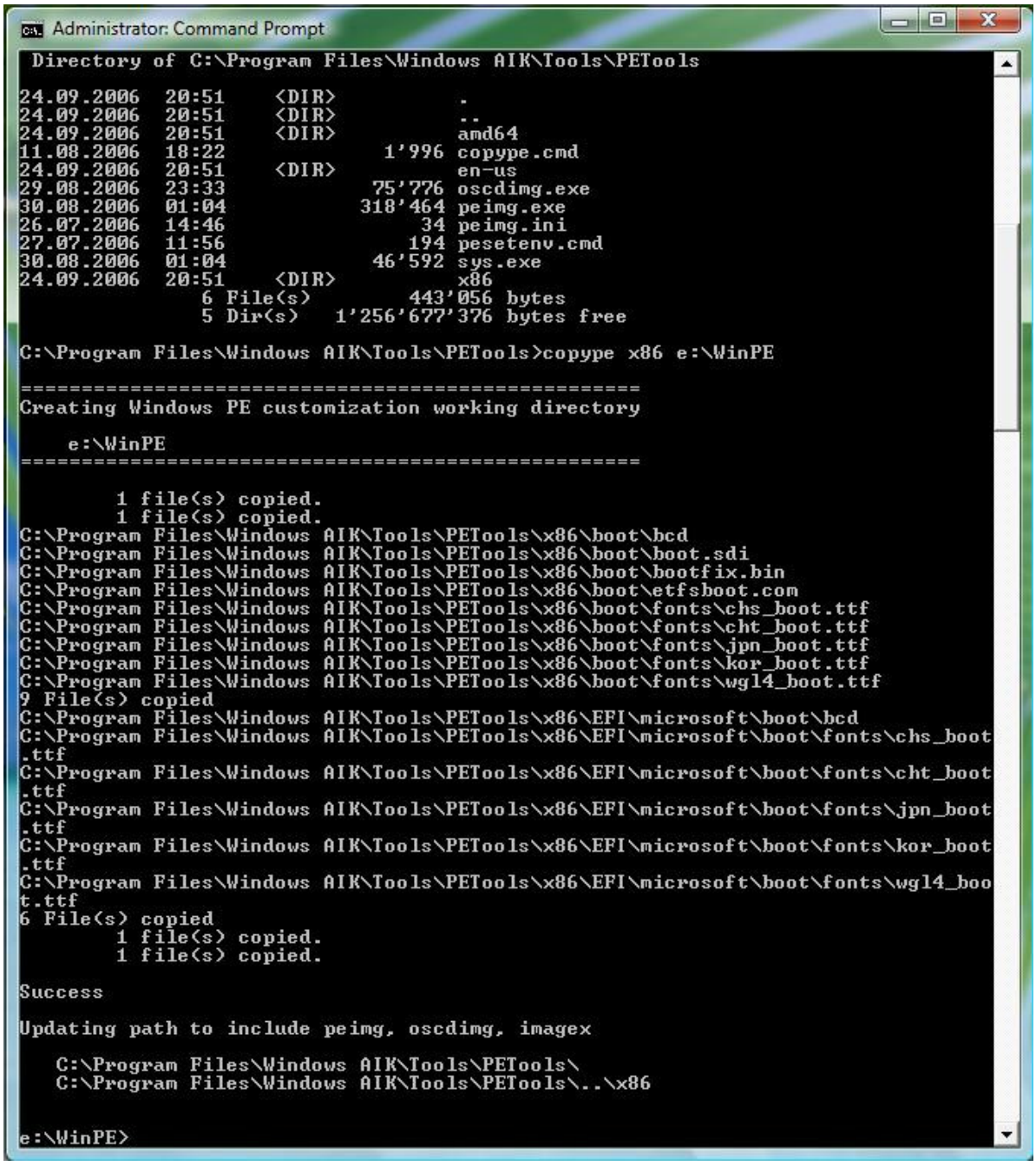
Ceci est dû au fait qu'il n'est pas possible de créer une image de notre installation pendant son fonctionnement (fichiers en cours d'utilisation qui ne peuvent être copiés).

Nous partons du principe que seul *Windows Vista* est installé sur le pc, et allons créer un CD *bootable* de *Windows PE* personnalisé en y incluant *ImageX*.

1.5.4.5.1.1 Créer la structure de fichiers *Windows PE*

Effectuer les commandes suivantes dans l'ordre

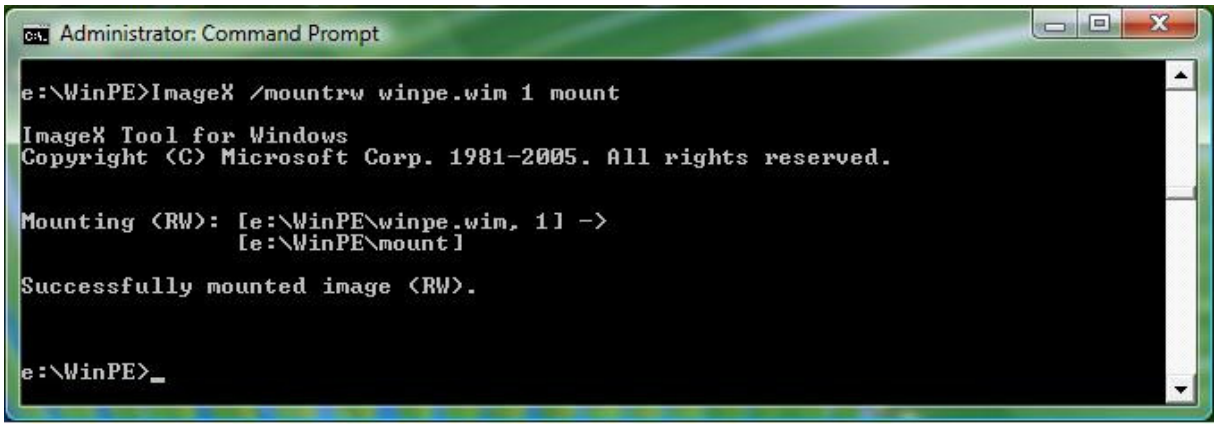
`cd C:\Program Files\Windows AIK\Tools\PETools`	Dans le répertoire contenant les programmes (*tools*) que nous devons utiliser
`COPYPE X86 e:\WinPE`	Prépare les fichiers de *Windows PE* pour traitement vers **e:\WinPE** Ici **e**:\ est une autre partition, il est possible de le faire sur la même partition que le système d'exploitation



1.5.4.5.1.2 Monter l'image pour la personnalisation

Monter une image signifie rendre visible le contenu réel d'une image, vers un dossier par exemple. (Voir ce qu'il y a à l'intérieur de cette image, les différents fichiers, dossiers).

`ImageX /mountrw winpe.wim 1 mount`	"monte" l'image de *Windows PE* (**`winpe.wim`**) vers le répertoire **`mount`**. C'est dans ce répertoire que vont se faire les mofifications. L'option **`/mountrw`** monte l'image en *read-write* (lecture-ecriture) Au lieu de **`/mountrw`**, il est possible d'utiliser l'option **`/mount`** dans le cas ou l'on ne souhaite pas modifier l'image.



1.5.4.5.1.3 Ajouter *ImageX* à notre image de *Windows PE*

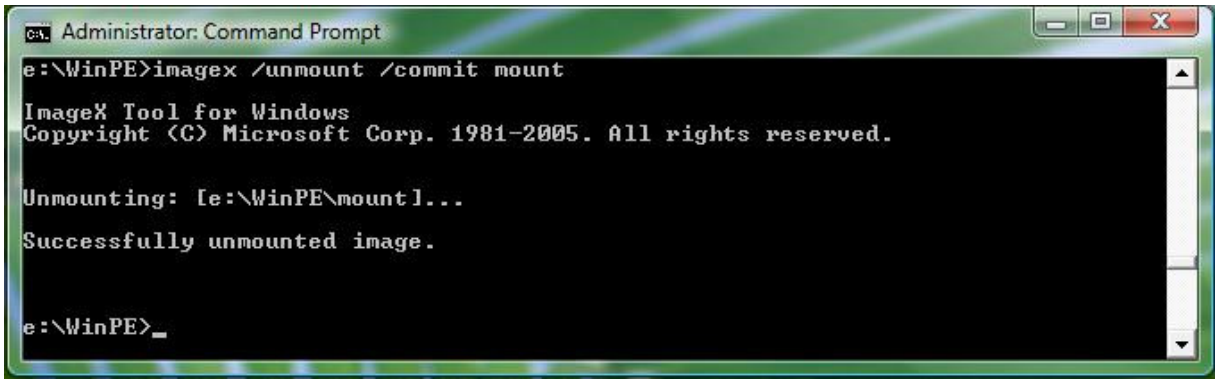
Ceci peut se faire via l'explorer ou à travers la ligne de commande :

Copy "C:\Program files\Windows AIK\Tools\x86\imagex.exe" e:\WinPE\mount\Program Files	Copie imagex.exe vers le répertoire de notre *Windows PE* personnalisé.

1.5.4.5.1.4 Créer l'image personalisée de *Windows PE*

Maintenant que nous avons effectué toutes les modifications voulues à *Windows PE*, il ne nous reste plus qu'a créer son image au format *WIM* à l'aide d'*ImageX*.

ImageX /Unmount /commit mount	Le paramètre **/commit** applique les changements effectués à l'image de base *Windows PE*. Le paramètre **/Unmount** "Démonte" l'image du répertoire **mount**, qui est à présent vide.



Cette méthode ne nécessite pas de devoir recréer une image complète, elle copie uniquement les changements à l'intérieur de l'image de base de *Windows PE*.

1.5.4.5.1.5 Créer un CD *bootable* avec notre image de *Windows PE*

Nous pouvons maintenant créer un CD *bootable* avec notre image *WIM*, ceci se fera à travers l'outil *OSCDIMG*.

Il faut commencer par copier notre image *WIM* personnalisée de *Windows PE* vers le répertoire ISO. Ce répertoire sera utilisé pour créer une image ISO de *Windows PE*, afin de pouvoir la graver sur CD.

copy "C:\WinPE\winpe.wim" e:\WinPE\ISO\sources	Copie notre image de *Windows PE* personnalisée vers le répertoire **e:\WinPE\ISO\sources**
cd e:\WinPE\ISO\sources	On se place dans le répertoire où sont à présent les images de *Windows PE*
del boot.wim	Efface l'image de base de *Windows PE* (**boot.wim**)
ren winpe.WIM boot.wim	Renomme notre image personnalisée de *Windows PE*
OSCDIMG –n –be:\WinPE\etfsboot.com e:\WinPE\ISO e:\WinPE\WinPECustom.iso	Crée l'image de *Windows PE* au format ISO, pour pouvoir être gravé sur CD. **Attention** : pas d'espaces après l'option **–b** et le répertoire contenant **etfsboot.com** !!! Cette opération durera quelques minutes. L'image ISO peut maintenant être gravée sur un CD.

1.5.4.5.2 Démarrer sur *Windows PE*

Avec le CD que l'on vient de créer, il est maintenant possible de l'insérer dans l'ordinateur et de *booter* sur notre *Windows PE* personnalisé.

Il faut bien évidemment au préalable avoir configuré le BIOS du pc de manière à pouvoir *booter* au démarrage sur CD.

1.5.4.5.3 Créer une image de notre *Vista* personnalisé à l'aide d'*ImageX*

Effectuer les commandes suivantes dans l'ordre :

cd X:\Program Files	Dans le répertoire contenant l'exécutable **imagex.exe** Ici, le disque **X** désigne l'image de *Windows PE* (copiée en mémoire RAM) qui est émulée sur un disque virtuel, portant la lettre X.
imagex /compress fast /check /capture c: d:\image.wim "Image personalisée Windows Vista" /verify	Capture une image de notre installation personnalisée **c:** vers **d:\image.wim** L'option **/check** contrôle l'intégrité de l'image *WIM.* L'option **/verify** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Il est conseillé de stocker l'image sur une autre partition que le système d'exploitation (ici sous **d:**)

Remarque : Comme je l'ai déjà cité précédemment, il est fortement conseillé d'activer notre installation *Vista* avant d'en créer une image, en effet *Vista* se bloque si l'activation n'est pas effectuée dans les 14 jours.

Lorsque *Vista* se bloque, un menu apparaît au démarrage, nous permettant d'activer notre installation en ligne. Cependant, ce menu étant très mal conçu, il ne nous permet pas de configurer les options réseau !

1.5.4.5.4 Installer l'image de notre *Vista* personnalisé

Cette opération est semblable à l'opération précédente qui crée une image.

Il faut pouvoir démarrer le pc avec notre CD de *Windows PE* puis effectuer les commandes :

cd X:\Program Files	Dans le répertoire contenant l'exécutable **imagex.exe**
imagex /apply d:\image.wim 1 c: /verify	Installe l'image **d:\image.wim** sur le disque (ou partition) **c:** Le **1** signifie que l'on installe l'image ayant l'identificateur 1 (possibilité d'avoir plusieurs images à l'intérieur d'un même fichier). L'option **/verify** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

1.5.5 Manipulations pour scénario 2

« Afin de pouvoir installer un serveur *WDS*, il est nécessaire de disposer d'une licence *Windows Server 2003*. Or, toutes les entreprises ne disposent pas d'une telle licence, notamment les petites entreprises.

Nous pouvons alors distinguer 2 autres scénarios intéressants :

- Une entreprise ne disposant pas de licence *Windows Server 2003* et ne pouvant donc pas installer de serveur *WDS*.

- Une entreprise disposant d'une licence *Windows Server 2003* et pouvant donc installer un serveur *WDS*.

Nous allons commencer par étudier le scénario d'une **petite entreprise ne pouvant pas installer de serveur *WDS*** et souhaitant déployer une image.

Cette entreprise possède plusieurs ordinateurs de configurations *hardware* (matérielles) différentes et aimerait simplifier la gestion de ses images.

Dans cette entreprise, il y a un ingénieur réseau, une secrétaire et un développeur. L'ingénieur réseau a besoin de programmes spécifiques pour son travail, la secrétaire a besoin d'un éditeur de texte uniquement, et le développeur a quant à lui besoin de programmes de développement.

L'idéal serait donc d'avoir au moins 3 images. »

Etant donné que l'entreprise ne possède pas de *Windows Server 2003*, il ne lui est pas possible d'installer un serveur *WDS* pour effectuer un déploiement d'images.

Il lui reste donc 2 solutions :

- Utiliser, comme lors du scénario 1, un CD *bootable* de *Windows PE* avec *ImageX* pour capturer et installer une image. L'image doit être préalablement gravée sur un DVD ou plusieurs CD. (pour créer un CD *bootable Windows PE*, voir scénario précédent ou en annexes)

- Si les postes où l'image doit être déployée disposent déjà d'un système d'exploitation, il est possible de copier une image disponible sur le réseau (dossier partagé), puis de l'installer sur le pc en question avec *ImageX*. Cette solution étant simple, elle ne sera pas décrite en détail dans ce document.

Jusqu'ici cela reste relativement semblable au scénario 1.

Cependant, comme l'entreprise dispose de plusieurs ordinateurs de configurations *hardware* différentes, je vais décrire comment créer une image personnalisée qui puisse être déployée sur d'autres ordinateurs.

De plus, comme plusieurs images différentes peuvent être stockées à l'intérieur du même fichier *WIM*, je vais montrer comment gérer ces différentes images.

1.5.5.1 Outils nécessaires

- *Sysprep* afin de préparer une installation pour son déploiement
- *ImageX* afin de créer et installer une image *WIM*.
- *Windows PE*, en créer un CD *bootable* et y inclure *ImageX*.
- *Format*, afin de formater une partition ou un disque dur, cet outil est disponible par défaut dans *Windows PE*.
- DVD de base *Windows Vista*

Une étude de *Sysprep* est disponible en page 74, paragraphe 1.9
Une étude d'*ImageX* est disponible en page 71, paragraphe 1.8
Une étude de *Windows PE* est disponible en page 59, paragraphe 1.7

1.5.5.2 Opérations à effectuer

- Capturer une installation de base de *Windows Vista* (20minutes avec une compression rapide)
- Créer et capturer une installation personnalisée pour un ingénieur système (dépendant du nombre de *softwares* à installer)
- Créer et capturer une installation personnalisée pour une secrétaire (dépendant du nombre de *softwares* à installer)
- Créer et capturer une installation personnalisée pour un développeur (dépendant du nombre de *softwares* à installer)
- Installer une des images créées sans fichier *Unattend.xml* (10minutes)
- Installer une des images créées avec fichier *Unattend.xml* (10minutes)

Une étude des fichiers *Unattend.xml* est disponible en page 82, paragraphe 1.10

Remarque : Pour simplifier la création des différentes images (ingénieur, secrétaire, développeur) et pour éviter de devoir réinstaller à chaque fois le système d'exploitation, j'ai créé au laboratoire uniquement un fichier texte sur le bureau (nommé ingénieur.txt, secretaire.txt ou développeur.txt) en fonction de l'image à créer.
Le principe est exactement identique à celui d'installation de programmes.
Le temps nécessaire à la création d'une installation personnalisée sera donc quasi nul (juste le temps de créer un fichier texte), le temps de capture sera d'environ 15 minutes.

1.5.5.2.1 Capturer une installation de base de *Windows Vista*

Commençons par **effectuer une installation de base** de ***Windows Vista*** sur un pc quelconque.

Cette installation par défaut de *Windows Vista* nous servira comme base pour créer les différentes images nécessaires au sein de l'entreprise.

Lorsque l'installation est terminée, redémarrer le pc sous *Windows PE* (voir scenario1 paragraphe 1.5.4.5.2 page 27) puis **capturer cette image** de base **à l'aide d'*ImageX***.

La question qui nous vient à l'esprit est la suivante: pourquoi capturer une telle installation de base ?
Tout simplement parce qu'il sera plus rapide de réinstaller cette image de base créé par *ImageX* (environ 10minutes) plutôt que de réinstaller à chaque fois *Windows Vista* avec le DVD d'origine (environ 30minutes).

Les différentes commandes à exécuter sous *Windows PE* afin de capturer notre installation de base sont les suivantes :

cd X:\Program Files	Dans le répertoire contenant l'exécutable **imagex.exe**
imagex /compress fast /check /capture c: d:\image.wim "Image de base Windows Vista" /verify	Capture une image de notre installation de base **c:** vers **d:\image.wim** L'option **/check** contrôle l'intégrité de l'image *WIM.* L'option **/verify** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Notre installation de base est maintenant capturée, elle portera l'**identificateur numéro 1** à l'intérieur du fichier **image.wim**

Remarque : Nous n'avons pas effectué de *sysprep* sur cette installation de base. Pourquoi ?
Nous allons dans ce scénario, effectuer toutes les installations personnalisées sur un seul poste.
Etant donné que nous n'avons pas effectué de *sysprep* sur l'installation de base de *Windows Vista*, ceci nous permettra de réinstaller sur le poste cette image de base sans devoir passer par la phase *Windows Welcome* ceci car elle a déjà été effectuée (en effectuant un *sysprep*, cette phase nécessite entre 5 et 10 minutes dépendant de la configuration *hardware* de la machine).

1.5.5.2.2 Créer et capturer une installation personnalisée pour un ingénieur système

Démarrer le pc **sur Windows Vista**, ajouter tous les programmes nécessaires pour l'ingénieur système.

Lorsque la personnalisation est terminée, ouvrir une fenêtre d'invite de commandes sous *Vista* et **utiliser *sysprep*** comme suit :

cd C:\Windows\System32\sysprep	Dans le répertoire contenant l'exécutable **sysprep.exe**
sysprep /oobe /generalize	L'option **/oobe** nous permet de lancer *Windows Welcome* au prochain démarrage. L'option **/generalize** enlève toutes les données spécifiques système de l'installation *Windows*. Ceci inclut les *event logs*, les SIDs ainsi que les autres informations uniques (adresses IP etc.)

Ceci nous permet de préparer une installation afin la déployer sur n'importe quel ordinateur.

Remarque : ***Sysprep* m'a posé quelques problèmes lors de son utilisation sur *Windows Vista RC1* et *RC2***. Parfois des erreurs sont survenues et par la suite, impossible d'utiliser correctement *Sysprep*, la seule solution a été de réinstaller *Windows Vista* !

Redémarrer le pc **sous *Windows PE***, puis effectuer les commandes suivantes :

`cd X:\Program Files`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`imagex /append /check c: d:\image.wim "Image pour ingénieur système" /verify`	Crée et ajoute l'image de notre installation personnalisée pour ingénieur système **`c:`** dans **`d:\image.wim`** L'option **`/check`** contrôle l'intégrité de l'image *WIM.* L'option **`/verify`** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Le fichier **`image.wim`** existe déjà, avec notre installation de base.
En utilisant *ImageX* et la commande **`/append`**, on ajoute à ce fichier l'image destinée aux ingénieurs système. Cette image portera l'**identificateur numéro 2** (chaque fois que l'on ajoute une image à un fichier *WIM* existant, le plus grand identificateur parmi toutes les images sera incrémenté de 1).

Remarque : La taille du fichier **`image.wim`** ne sera pas doublée, un fichier *WIM* comporte une seule instance pour chaque fichier (comme vu dans le paragraphe 1.2.3 page 12).

1.5.5.2.3 Créer et capturer une installation personnalisée pour une secrétaire

Maintenant que nous avons effectué une installation personnalisée pour un ingénieur, nous avons donc déjà un poste configuré pour un ingénieur.

Afin de créer une installation personnalisée pour une secrétaire, nous avons deux solutions :

- Effectuer cette installation sur un nouveau poste, ce qui implique réinstaller *Windows Vista* avec le DVD d'installation (environ 30minutes nécessaires avec un Pentium 4 2.8GHz 1GB RAM) ou en utilisant l'image créée dans le paragraphe 1.5.5.2.1 (ce qui implique graver cette image sur un DVD).
  De plus avec cette solution l'image de base doit être "*sysprepée*" pour pouvoir être déployée sur un autre ordinateur, ce qui implique le lancement de *Windows Welcome* et une perte de temps entre 5 et 10 minutes !

- Formater le poste où l'installation personnalisée pour un ingénieur a été créée, puis installer l'image de base de *Windows Vista* créée dans le paragraphe 1.5.5.2.1

Je vais ici montrer les manipulations pour la deuxième solution afin de tout effectuer sur le même poste, ceci afin d'éviter les lancements de *Windows Welcome* et de plus éviter de graver le fichier *WIM* (pour pouvoir installer l'image sur un autre ordinateur) comportant, pour l'instant, uniquement l'image de base de *Vista* ainsi que l'image personnalisée pour un ingénieur.

Les diverses manipulations sont donc les suivantes :

Réinstaller l'image de base sur le pc, pour cela **démarrer sous *Windows PE*** et effectuer les commandes suivantes :

`format c:`	Formate la partition **`c:`** Etant donné que l'ordinateur comporte une installation pour un ingénieur, il est nécessaire de formater cette partition avant de réinstaller l'image de base de *Windows Vista*.
`cd X:\Program Files`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`imagex /apply d:\image.wim 1 c: /verify`	Installe l'image **`d:\image.wim`** sur le disque (ou partition) **`c:`** Le **`1`** signifie que l'on installe l'image ayant l'identificateur 1 qui est notre image *Vista* de base. L'option **`/verify`** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Lorsque l'installation est terminée, **redémarrer le pc sur *Windows Vista***, puis installer tous les programmes nécessaires pour les secrétaires.

Lorsque la personnalisation est terminée, ouvrir une fenêtre d'invite de commandes sous *Vista* et **utiliser *sysprep*** comme suit :

`cd C:\Windows\System32\sysprep`	Dans le répertoire contenant l'exécutable **`sysprep.exe`**
`sysprep /oobe /generalize`	L'option **`/oobe`** nous permet de lancer *Windows Welcome* au prochain démarrage. L'option **`/generalize`** enlève toutes les données spécifiques système de l'installation *Windows*. Ceci inclut les *event logs*, les SIDs ainsi que les autres informations uniques (adresses IP etc.)

Sysprep nous permet de préparer une installation afin la déployer sur n'importe quel ordinateur.

Redémarrer le pc **sous *Windows PE***, puis effectuer les commandes suivantes :

`cd X:\Program Files`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`imagex /append /check c: d:\image.wim "Image pour secretaires" /verify`	Crée et ajoute l'image de notre installation personnalisée pour secrétaires **`c:`** dans **`d:\image.wim`** L'option **`/check`** contrôle l'intégrité de l'image *WIM.* L'option **`/verify`** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Le fichier **`image.wim`** existe déjà, avec les deux images créées précédemment (image de base *Vista* et image pour ingénieur système).
En utilisant *ImageX* et la commande **`/append`**, on ajoute à ce fichier l'image destinée aux secrétaires. Cette image portera l'**identificateur numéro 3**.

1.5.5.2.4 Créer et capturer une installation personnalisée pour un développeur

Le principe est le même que précédemment, voir Créer une installation personnalisée pour une secrétaire.

1.5.5.2.5 Tests effectués

Comme indiqué précédemment, pour simplifier la création des différentes images (ingénieur, secrétaire, développeur) et pour éviter de devoir réinstaller à chaque fois le système d'exploitation, j'ai créé au laboratoire uniquement un fichier texte sur le bureau (nommé ingénieur.txt, secretaire.txt ou développeur.txt) en fonction de l'image à créer.

Chaque installation a été capturée et ajoutée au même fichier *WIM*.

Le but étant de montrer comment incorporer plusieurs images à l'intérieur du même fichier *WIM* ainsi que de vérifier la taille du fichier *WIM* lors de chaque ajout d'image.

J'ai utilisé pour cela la version *Windows Vista RC2 Build 5744*, avec une installation de *BDD 2007* et *Windows AIK*.

Les résultats obtenus sont les suivants :

	Taille du fichier *WIM*
Création d'une image de l'installation de référence	3'652'558 *KBytes*
Ajout d'une image pour ingénieurs (fichier ingénieur.txt)	3'674'605 *KBytes*
Ajout d'une image pour secrétaires (fichier secretaire.txt)	3'696'226 *KBytes*
Ajout d'une image pour développeurs (fichier developpeur.txt)	3'717'533 *KBytes*

Nous pouvons constater que la taille du fichier *WIM* augmente d'environ 22 *KBytes* à chaque ajout d'image.

Pourtant, chaque fichier texte ajouté ne fait pas 22 *KBytes*, sa taille réelle est de 9 *Bytes* mais son occupation sur disque est de 4 *KBytes*.

La question qui nous vient alors à l'esprit est : Pourquoi le fichier augmente-t-il d'environ 22*Kbytes* alors qu'il y a seulement un fichier de 9*bytes* ajouté ?

La réponse est relativement simple est évidente : Comme décrit dans le chapitre 1.2.3 page 12, les fichiers *WIM* comportent une seule instance pour chaque fichier.
Une seule instance par fichier pour les différentes images au sein du fichier *WIM* implique que l'on doit avoir des liens référençant quels sont les fichiers présents dans chaque image.
Ce sont ces divers liens qui font que la taille du fichier *WIM* augmente d'environ 22 *KBytes* pour un fichier teste de seulement 9 *Bytes*

Cependant, en pratique, nous n'ajoutons pas un fichier texte pour personnaliser les installations, mais des programmes volumineux !

Par exemple si un programme de 100 *MBytes* est ajouté, la taille de l'image augmentera de 100 *MBytes* plus environ 22 *KBytes* qui deviennent alors vraiment négligeables !
Nous pouvons consulter les métadonnées du fichier *WIM* comportant les diverses images avec la commande suivante :

imagex /info d:\image.wim	Affiche les informations de toutes les images présentes dans **image.wim**, y compris le numéro d'identificateur associé à chaque image.

Remarques :

Partie **WIM Information** :

- Nous pouvons voir le **GUID** qui est un numéro hexadécimal unique pour chaque fichier *WIM*.

- Nous pouvons voir le nombre d'images contenues dans notre fichier *WIM* (**Image Count : 4**)

- La méthode de compression est aussi affichée (**Compression : XPRESS**)

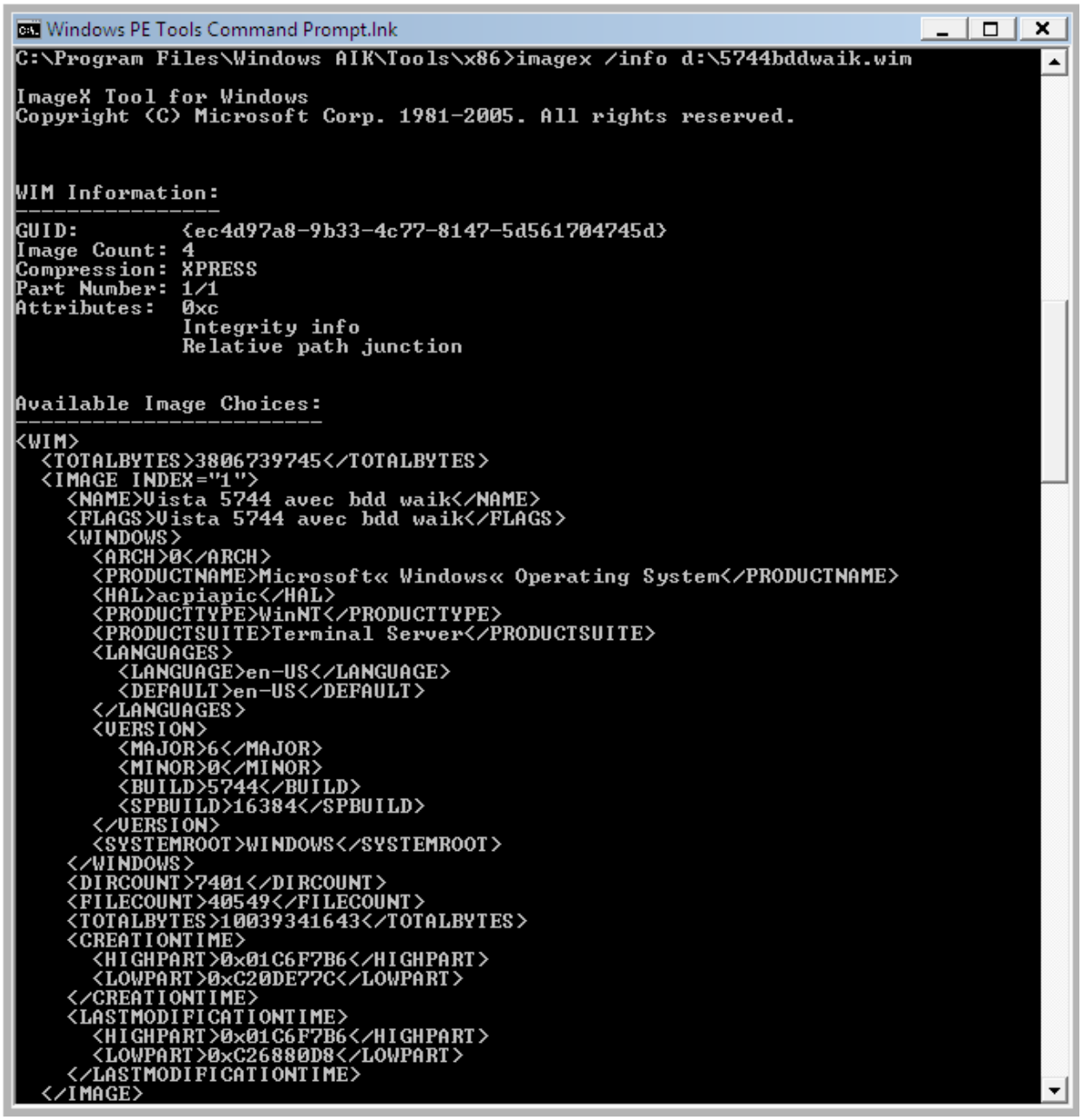
- Si le fichier *WIM* a été découpé en plusieurs parties de plus petite taille, le nombre de parties est affiché (ici une seule partie ce qui affiche **Part Number : 1/1**)

Partie **Available Image Choices** :

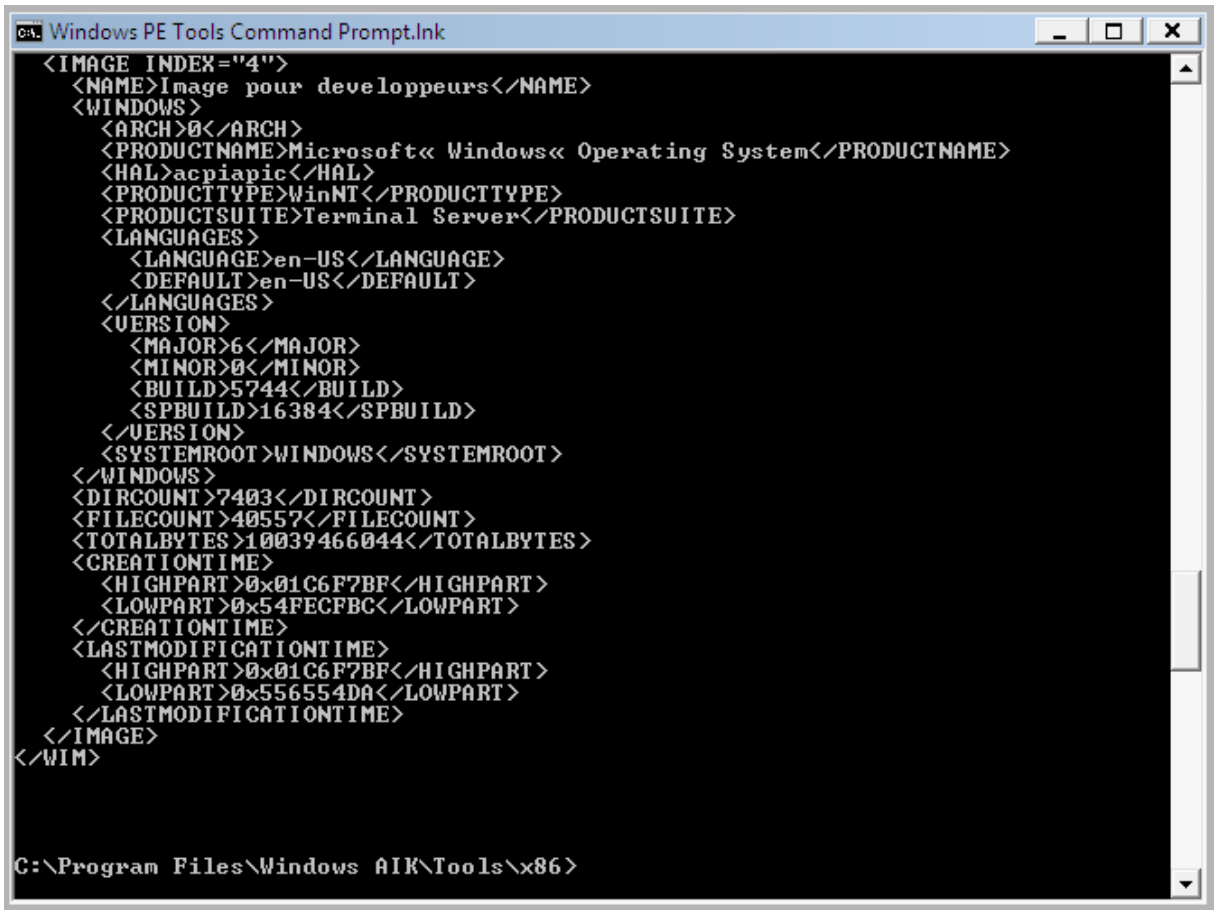
- Il est possible de voir l'identificateur associé à chaque image dans les balises **<IMAGE INDEX="numéro d'index">**

- Il est possible de voir après la balise **<NAME>**, le nom que l'on a donné précédemment à l'image

- Diverses autres informations sont disponibles.


Windows PE Tools Command Prompt.lnk
C:\Program Files\Windows AIK\Tools\x86>imagex /info d:\5744bddwaik.wim
ImageX Tool for Windows
Copyright (C) Microsoft Corp. 1981-2005. All rights reserved.
WIM Information:
GUID: {ec4d97a8-9b33-4c77-8147-5d561704745d}
Image Count: 4
Compression: XPRESS
Part Number: 1/1
Attributes: 0xc
Integrity info
Relative path junction
Available Image Choices:
<WIM>
<TOTALBYTES>3806739745</TOTALBYTES>
<IMAGE INDEX="1">
<NAME>Vista 5744 avec bdd waik</NAME>
<FLAGS>Vista 5744 avec bdd waik</FLAGS>
<WINDOWS>
<ARCH>0</ARCH>
<PRODUCTNAME>Microsoft« Windows« Operating System</PRODUCTNAME>
<HAL>acpiapic</HAL>
<PRODUCTTYPE>WinNT</PRODUCTTYPE>
<PRODUCTSUITE>Terminal Server</PRODUCTSUITE>
<LANGUAGES>
<LANGUAGE>en-US</LANGUAGE>
<DEFAULT>en-US</DEFAULT>
</LANGUAGES>
<VERSION>
<MAJOR>6</MAJOR>
<MINOR>0</MINOR>
<BUILD>5744</BUILD>
<SPBUILD>16384</SPBUILD>
</VERSION>
<SYSTEMROOT>WINDOWS</SYSTEMROOT>
</WINDOWS>
<DIRCOUNT>7401</DIRCOUNT>
<FILECOUNT>40549</FILECOUNT>
<TOTALBYTES>10039341643</TOTALBYTES>
<CREATIONTIME>
<HIGHPART>0x01C6F7B6</HIGHPART>
<LOWPART>0xC20DE77C</LOWPART>
</CREATIONTIME>
<LASTMODIFICATIONTIME>
<HIGHPART>0x01C6F7B6</HIGHPART>
<LOWPART>0xC26880D8</LOWPART>
</LASTMODIFICATIONTIME>
</IMAGE>


Windows PE Tools Command Prompt.lnk
<IMAGE INDEX="2">
<NAME>Image pour ingenieurs</NAME>
<WINDOWS>
<ARCH>0</ARCH>
<PRODUCTNAME>Microsoft« Windows« Operating System</PRODUCTNAME>
<HAL>acpiapic</HAL>
<PRODUCTTYPE>WinNT</PRODUCTTYPE>
<PRODUCTSUITE>Terminal Server</PRODUCTSUITE>
<LANGUAGES>
<LANGUAGE>en-US</LANGUAGE>
<DEFAULT>en-US</DEFAULT>
</LANGUAGES>
<VERSION>
<MAJOR>6</MAJOR>
<MINOR>0</MINOR>
<BUILD>5744</BUILD>
<SPBUILD>16384</SPBUILD>
</VERSION>
<SYSTEMROOT>WINDOWS</SYSTEMROOT>
</WINDOWS>
<DIRCOUNT>7402</DIRCOUNT>
<FILECOUNT>40571</FILECOUNT>
<TOTALBYTES>10040528117</TOTALBYTES>
<CREATIONTIME>
<HIGHPART>0x01C6F7B9</HIGHPART>
<LOWPART>0x5F29BB7E</LOWPART>
</CREATIONTIME>
<LASTMODIFICATIONTIME>
<HIGHPART>0x01C6F7B9</HIGHPART>
<LOWPART>0x5F8DDE42</LOWPART>
</LASTMODIFICATIONTIME>
</IMAGE>
<IMAGE INDEX="3">
<NAME>Image pour secretaires</NAME>
<WINDOWS>
<ARCH>0</ARCH>
<PRODUCTNAME>Microsoft« Windows« Operating System</PRODUCTNAME>
<HAL>acpiapic</HAL>
<PRODUCTTYPE>WinNT</PRODUCTTYPE>
<PRODUCTSUITE>Terminal Server</PRODUCTSUITE>
<LANGUAGES>
<LANGUAGE>en-US</LANGUAGE>
<DEFAULT>en-US</DEFAULT>
</LANGUAGES>
<VERSION>
<MAJOR>6</MAJOR>
<MINOR>0</MINOR>
<BUILD>5744</BUILD>
<SPBUILD>16384</SPBUILD>
</VERSION>
<SYSTEMROOT>WINDOWS</SYSTEMROOT>
</WINDOWS>
<DIRCOUNT>7402</DIRCOUNT>
<FILECOUNT>40535</FILECOUNT>
<TOTALBYTES>10039486846</TOTALBYTES>
<CREATIONTIME>
<HIGHPART>0x01C6F7BD</HIGHPART>
<LOWPART>0x0D8CB966</LOWPART>
</CREATIONTIME>
<LASTMODIFICATIONTIME>
<HIGHPART>0x01C6F7BD</HIGHPART>
<LOWPART>0x0DF0DC2A</LOWPART>
</LASTMODIFICATIONTIME>
</IMAGE>



1.5.5.2.6 Installer une des images créées sans fichier *Unattend.xml*

Démarrer sur *Windows PE* et effectuer les commandes suivantes :

`cd X:\Program Files`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`imagex /info d:\image.wim`	Affiche les informations de toutes les images présentes dans **`image.wim`**, y compris le numéro d'identificateur associé à chaque image.
`imagex /apply d:\image.wim X c: /verify`	Installe l'image **`d:\image.wim`** sur le disque (ou partition) **`c:`** A la place du **`X`**, mettre le numéro d'identificateur de l'image à installer. L'option **`/verify`** contrôle qu'il n'y ait pas d'erreurs ni de fichiers dupliqués.

Remarque : Ne pas oublier de formater au préalable si besoin.

1.5.5.2.7 Installer une des images créées avec fichier *Unattend.xml*

L'utilisation d'un fichier *Unattend.xml* afin d'automatiser et configurer une installation *Windows Vista* n'est pas possible avec *ImageX*.

Pour pouvoir utiliser un fichier *Unattend.xml* lors d'une installation, il faut utiliser soit un serveur *WDS* soit *Windows Setup*.

Etant donné que dans notre scénario, l'entreprise ne dispose pas de licence *Windows Server 2003*, on ne peut pas utiliser un serveur *WDS*.

Nous allons donc nous concentrer sur l'utilisation de *Windows Setup*, cet outil est fourni dans le DVD de *Windows Vista*.

Copier l'intégralité du **DVD d'installation de *Windows Vista* sur le disque dur**, sauf le fichier image *install.wim* ainsi que les *catalog files* (*.clg) qui lui sont associés. Ces fichiers se trouvent dans le dossier **`\sources`**.

Copier notre **image personnalisée *Windows Vista*** vers le répertoire **`\sources`**, la renommer en tant que **`install.wim`**

Ouvrir cette image avec *Windows System Image Manager*, autoriser cet outil à créer le *catalog file* qui sera associé à notre image, puis créer le fichier *Unattend.xml* souhaité (voir document *Windows System Image Manager* en annexes)

Redémarrer le PC puis ***booter*** sur ***Windows PE***.

Dans l'invite de commandes, se mettre dans le répertoire où le DVD de base *Windows Vista* a été copié, puis exécuter la commande suivante :

`setup.exe /unattend:d:\unattend.xml`	Remarque : Il n'y a pas d'espaces après **`/unattend:`** , le champ suivant indique l'emplacement de notre fichier à utiliser.

Remarque : Ne pas oublier de formater au préalable si besoin.

1.5.6 Manipulations pour scénario 3

« Afin de pouvoir installer un serveur *WDS*, il est nécessaire de disposer d'une licence *Windows Server 2003*.

Nous allons donc terminer par étudier le scénario d'une **grande entreprise disposant d'une licence *Windows Server 2003* afin d'installer un serveur *WDS***.

Cette grande entreprise possède plusieurs ordinateurs de configurations *hardware* différentes et aimerait simplifier la gestion de ses images, déployer ces images à travers son réseau.

De plus il faudrait que les installations soient le plus automatisées possibles afin de demander un minimum d'interactions humaines. »

Le principe de la gestion des différentes images à l'intérieur du même fichier *WIM* est décrit dans le scénario précédent, page 29

Nous allons ici voir comment configurer et utiliser un serveur *WDS* (*Windows Deployment Services*) sur *Windows Server 2003* pour le déploiement d'images à travers le réseau.

Nous allons aussi voir comment automatiser une installation, ceci à l'aide de l'outil *Windows System Image Manager* qui nous permet de créer un fichier *Unattend.xml*.

Nous verrons aussi comment personnaliser une installation à distance avec un fichier *Unattend.xml*

Une étude plus sur les fichiers *Unattend.xml* est disponible en page 82, paragraphe 1.10
Une étude plus détaillée de *Windows System Image Manager* est disponible en annexe

1.5.6.1 Outils nécessaires

- Une installation de *Windows AIK*
- Un *Windows Server 2003* avec une installation de serveur *RIS* (*Remote Installation Services*) puis mise à jour de *RIS* vers *WDS* (*Windows Deployment Services*)
- Des cartes réseau compatibles *PXE*, pour pouvoir *booter* directement sur le réseau les pc ou l'on souhaite déployer une image, sans installations supplémentaires.
- *Sysprep* afin de préparer une installation pour son déploiement
- *ImageX* afin de créer des images *WIM*.
- *Windows PE*
- DVD de base *Windows Vista*.

Une étude de *WDS* est disponible en page 87, paragraphe 1.11
Une étude de *PXE* est disponible en page 88, paragraphe 1.11.5
Une étude de *Sysprep* est disponible en page 74, paragraphe 1.9
Une étude d'*ImageX* est disponible en page 71, paragraphe 1.8
Une étude de *Windows PE* est disponible en page 59, paragraphe 1.7

1.5.6.2 Opérations à effectuer

- Installer un serveur *WDS* (2minutes sans compter les pré-requis tels qu'installer un serveur *DHCP*, *Active Directory* et autres)
- Configurer le serveur *WDS* (8minutes)
- Installer des images de *boot* sur le serveur *WDS* (10minutes)
- Installer des images d'installation (*Vista*) sur le serveur *WDS* (9minutes)
- Automatiser une installation avec un fichier *Unattend.xml* (20minutes)
- Personnaliser une installation avec un fichier *Unattend.xml* (10minutes)
- Installer une image sur un pc distant en utilisant notre serveur *WDS* (15minutes)

1.5.6.2.1 Installer un serveur *WDS*

Avant de pouvoir installer un serveur *WDS*, notre réseau doit contenir :

- ***Windows Server 2003 SP1*** avec ***RIS*** installé. Ce serveur *RIS* n'a pas besoin d'être configuré, il est uniquement nécessaire pour que l'*update* (mise à jour) vers un serveur *WDS* puisse s'effectuer.

- ***DHCP***, il faut un serveur *DHCP* car *WDS* utilise *PXE* (qui lui même utilise *DHCP*).

- ***DNS***, un serveur *DNS* doit être présent pour exécuter *WDS*.

- ***Active Directory***, un serveur *WDS* doit être membre d'un domaine *Active Directory*.

- Une **partition *NTFS* sur** le **serveur *WDS*** car *WDS* à besoin d'une partition *NTFS* pour stocker la ou les images.

L'installation de ces différents outils ne sera pas décrite dans ce document du fait qu'il s'agit d'installations basiques.

Nous partons du principe qu'un *Windows Server 2003* est installé, avec y compris un serveur *RIS*, *BDD 2007* et *Windows AIK* (voir page 16 pour installation de *BDD 2007* et *Windows AIK*)

Dans le cas ou il n'y a pas de serveur *RIS* installé sous *Windows Server 2003*, il est toutefois possible de l'ajouter avec le CD d'installation de *Windows Server 2003*.

Pour installer la mise à jour du serveur *RIS* vers un serveur *WDS*, il n'est pas nécessaire de configurer les options de *RIS* au préalable, il suffit de lancer l'exécutable correspondant à notre architecture :

- **`windows-deployment-services-update-x86.exe`** pour les pc avec une architecture 32 bits
- **`windows-deployment-services-update-amd64.exe`** pour les pc avec une architecture 64 bits AMD

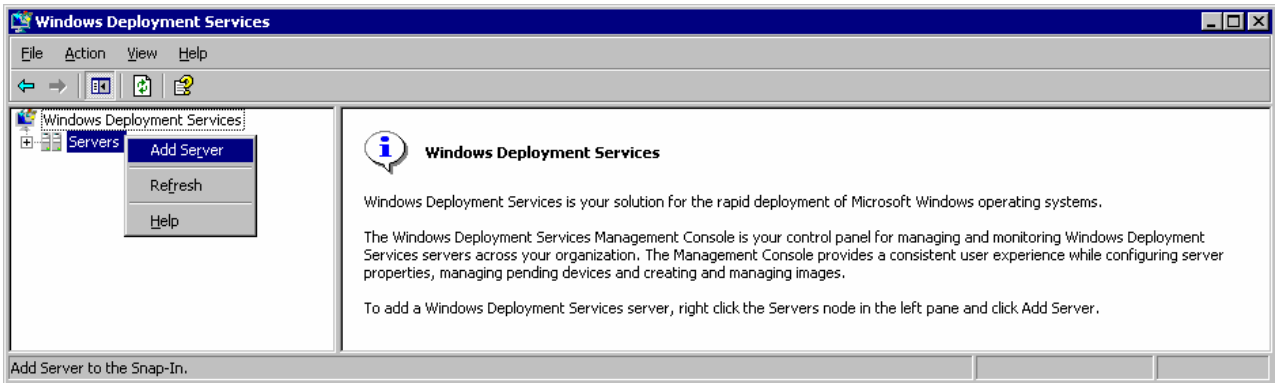
Ces exécutables sont disponibles dans le répertoire
`C:\Program Files\BDD 2007\WAIK\WDS`

1.5.6.2.2 Configurer le serveur *WDS*

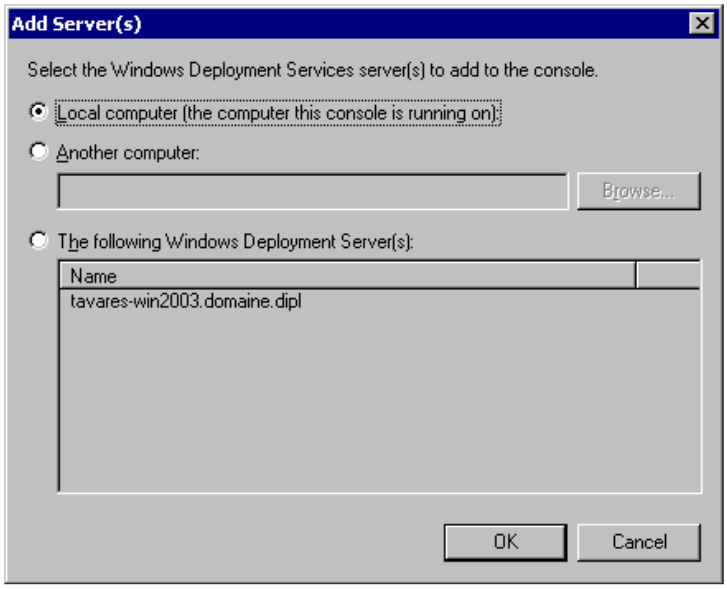
Nous allons ici configurer le serveur via son interface graphique. Il est aussi possible d'effectuer ces configurations en lignes de commandes, pour plus d'informations consulter le fichier **`wdsoobstepbystep.doc`** qui est présent dans **`C:\Program Files\BDD 2007\WAIK\WDS`** (avec *BDD2007* installé bien évidemment).

Aller dans *Administrative Tools* et ouvrir *Windows Deployment Services.*

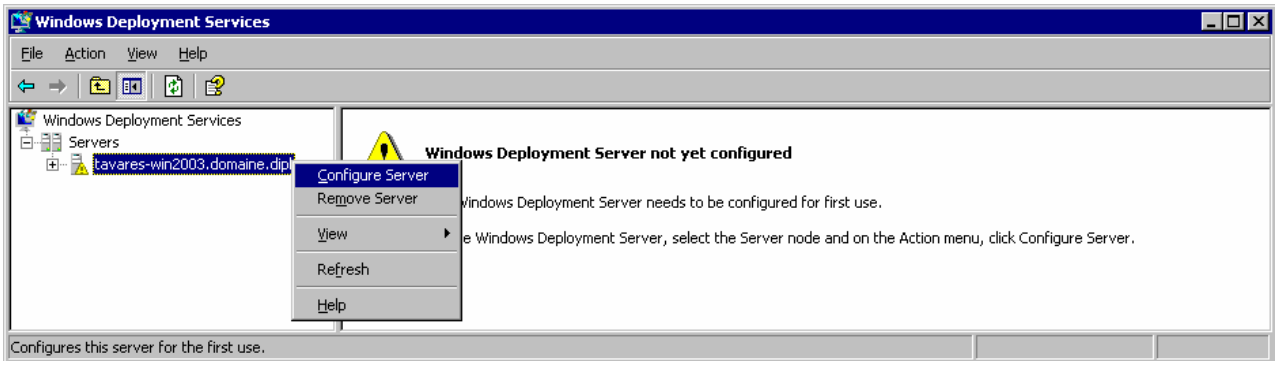
Pour ajouter un nouveau serveur, clic droit sur *Servers* puis *Add Server.*



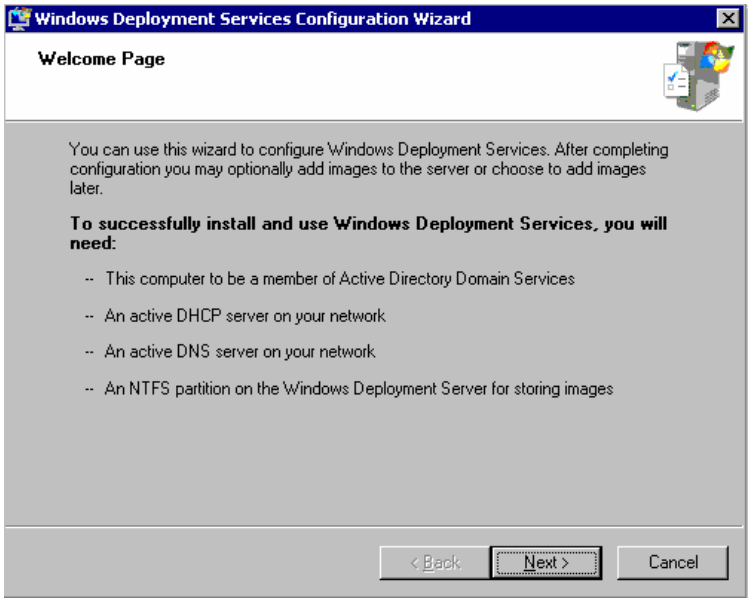
Sélectionner *Local computer (the computer this console is running on)*
Ceci signifie que notre serveur *WDS* sera installé sur la même machine que notre *Windows Server 2003*.



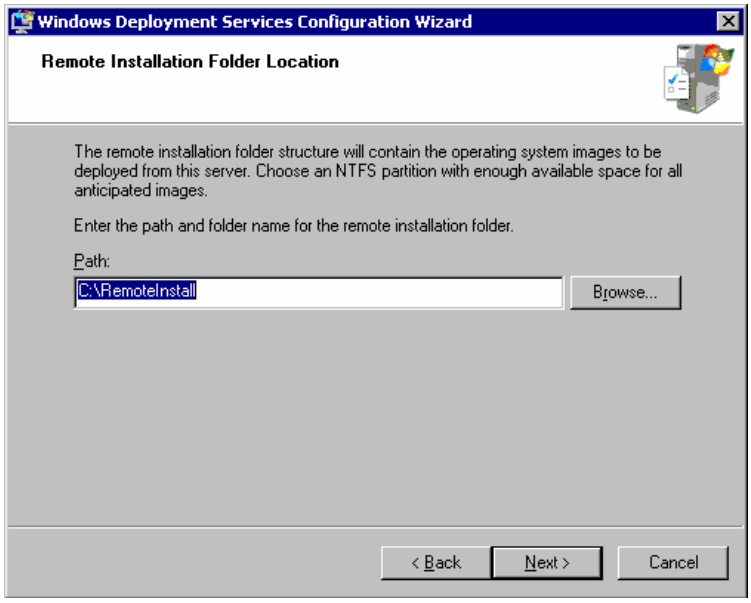
Nous avons créé notre serveur *WDS*, nous devons maintenant le configurer, pour cela effectuer un clic droit sur notre serveur, puis *Configure Server*.



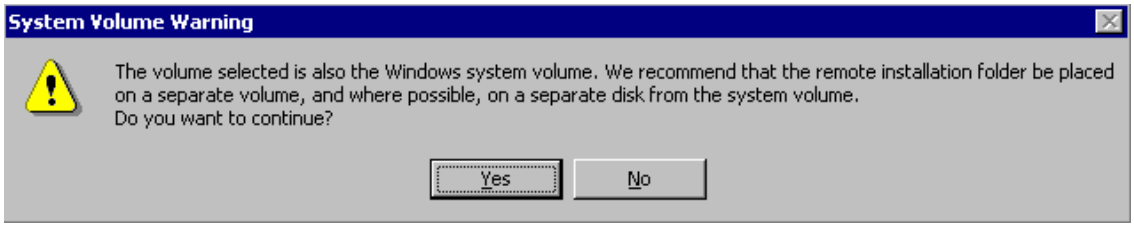
Cette action lance *Windows Deployment Services Configuration Wizard*.
La première fenêtre de ce *Wizard* (assistance de configuration) nous rappelle les différents éléments réseau qu'il faut avoir pour avoir un serveur *WDS* fonctionnel.



Choisir l'emplacement de stockage des différentes images pour leur déploiement à partir du serveur *WDS*.

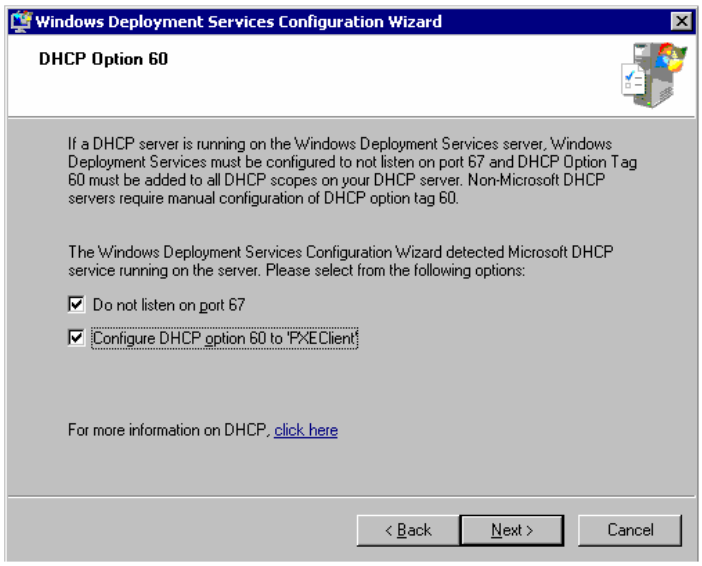


Si l'emplacement choisi se trouve sur le même disque (ou partition) que notre système d'exploitation, le message suivant sera affiché :



Il est recommandé de choisir un emplacement situé sur un autre disque (ou partition) que notre système d'exploitation, ceci afin de ne pas perdre nos images si le système d'exploitation doit être réinstallé.

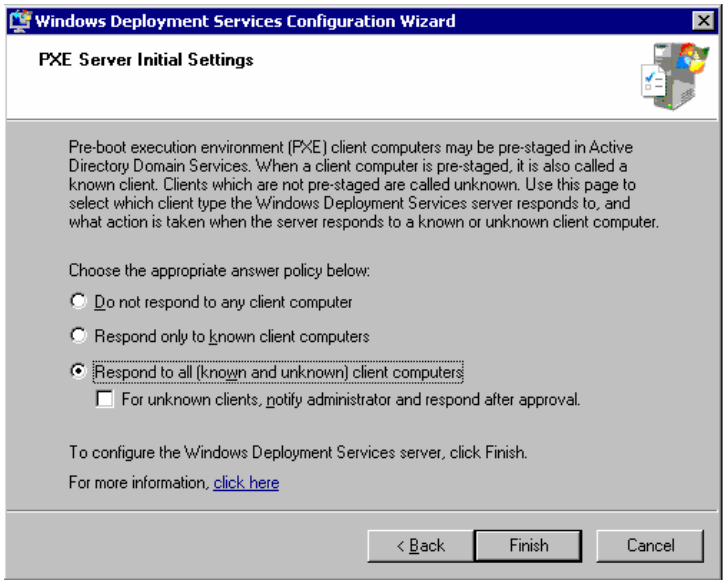
Si notre serveur *DHCP* est situé sur la même machine que notre serveur *WDS*, il nous faut activer les 2 options ci dessous :



Pour configurer notre serveur *PXE*, choisir une des options ci dessous.

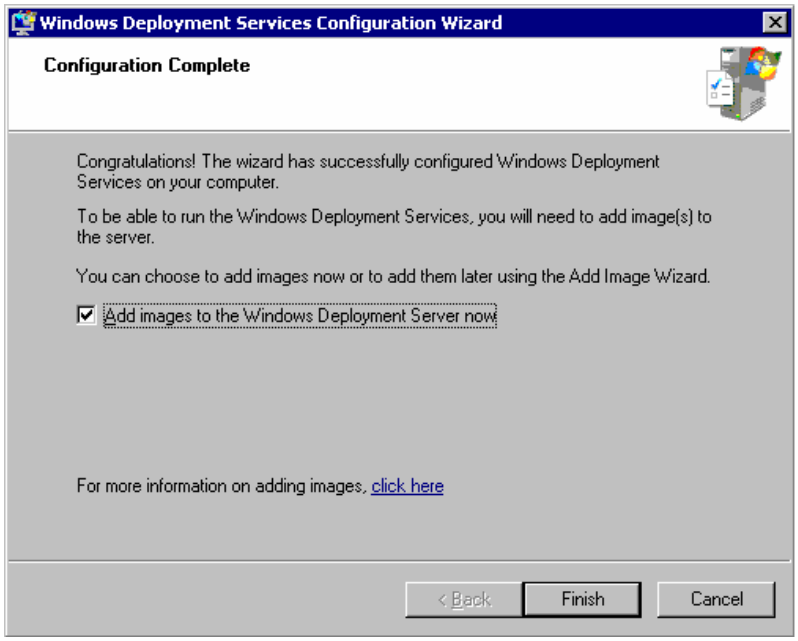
Si l'option *Respond only to known client computers*est choisie, les pc clients devront au préalable être ajoutés dans notre *Active Directory* (Consulter la documentation fournie avec *WDS* pour plus de renseignements).

Dans notre cas, nous allons répondre à toutes les demandes *PXE* :



La configuration de notre serveur est maintenant terminée.

Il est maintenant possible d'ajouter des images, cette opération sera décrite dans le sous-chapitre suivant.

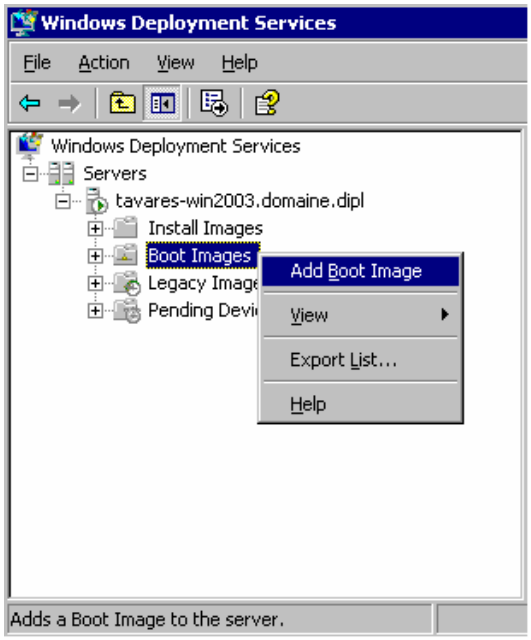


1.5.6.2.3 Ajouter des images de *boot* sur le serveur *WDS*

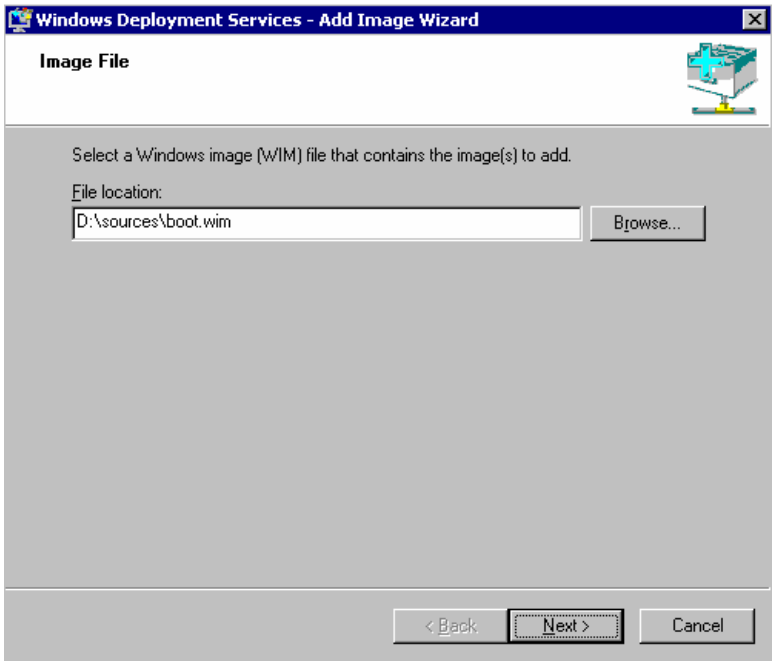
Une image de *boot* est nécessaire afin de faire démarrer les différents pc clients à l'aide de *Windows PE*.

Nous allons ici ajouter l'image *boot* de *Windows PE*, disponible sur le DVD de *Windows Vista*.
Cette image de *boot* est nécessaire aux pc clients qui désirent se connecter à notre serveur *WDS* (ceci à l'aide de *PXE*).

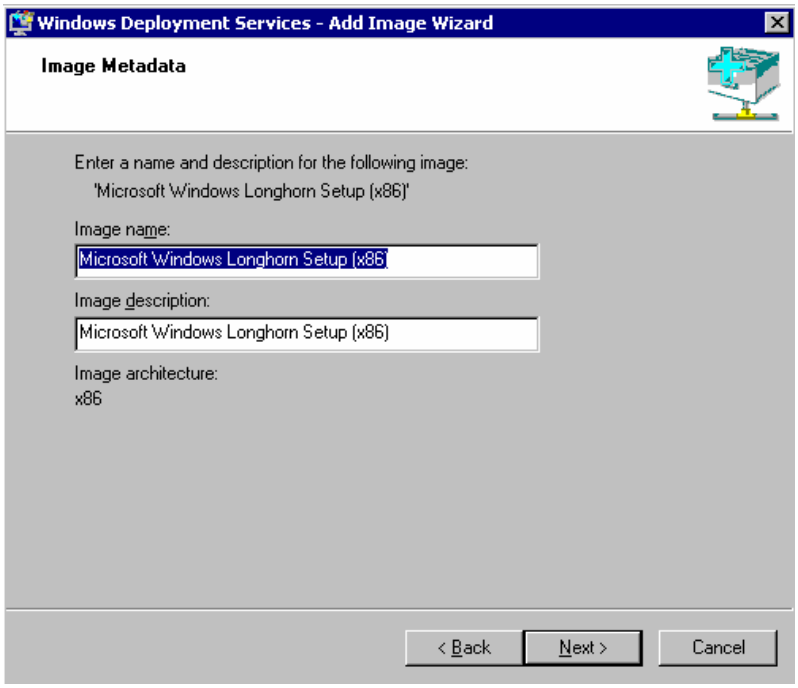
Effectuer un clic droit sur *Boot Images*, puis sélectionner *Add Boot Image*.



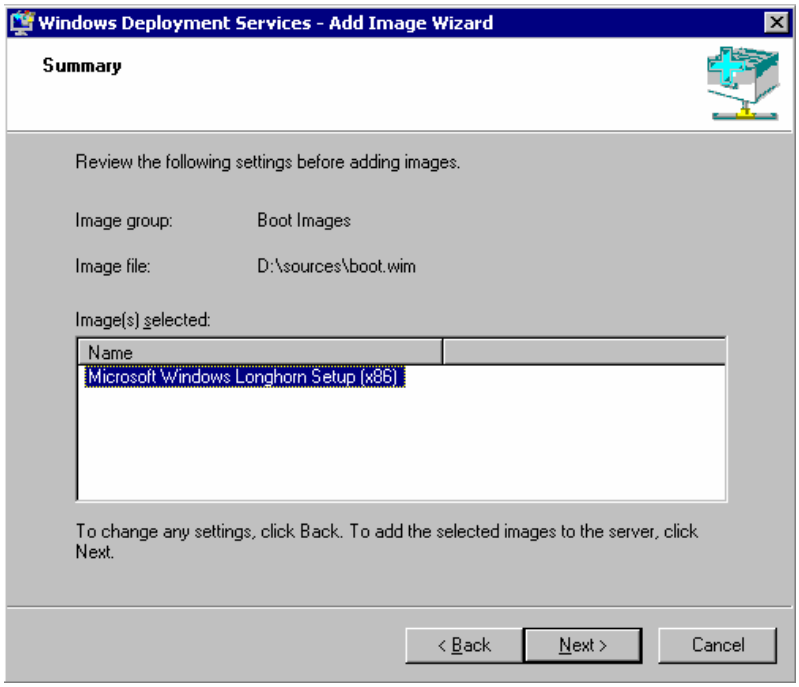
Sélectionner le fichier image de Windows PE (nommé **`boot.wim`**) se trouvant sur le DVD de *Windows Vista*.



Entrer un nom et une description pour l'image sélectionnée.



Une fenêtre de résumé est affichée, en cliquant sur *Next* les différentes opérations demandées seront effectuées.

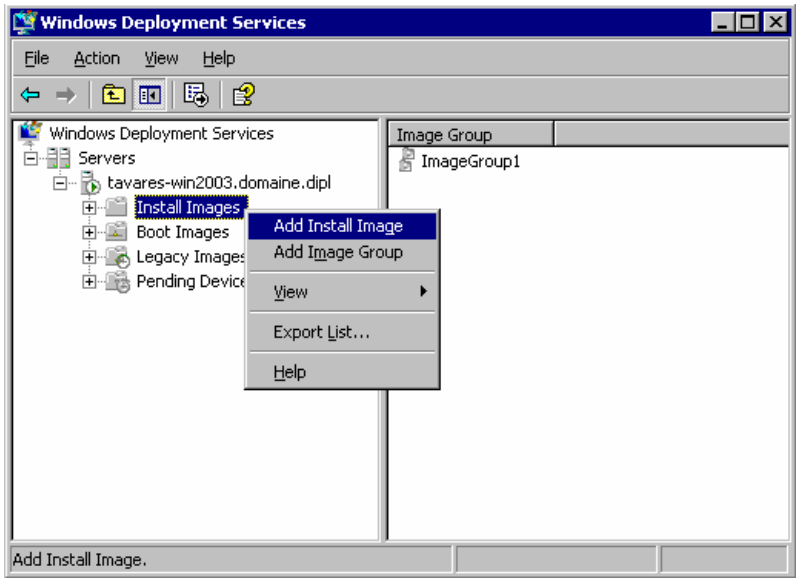


1.5.6.2.4 Ajouter des images d'installation (*Vista*) sur le serveur *WDS*

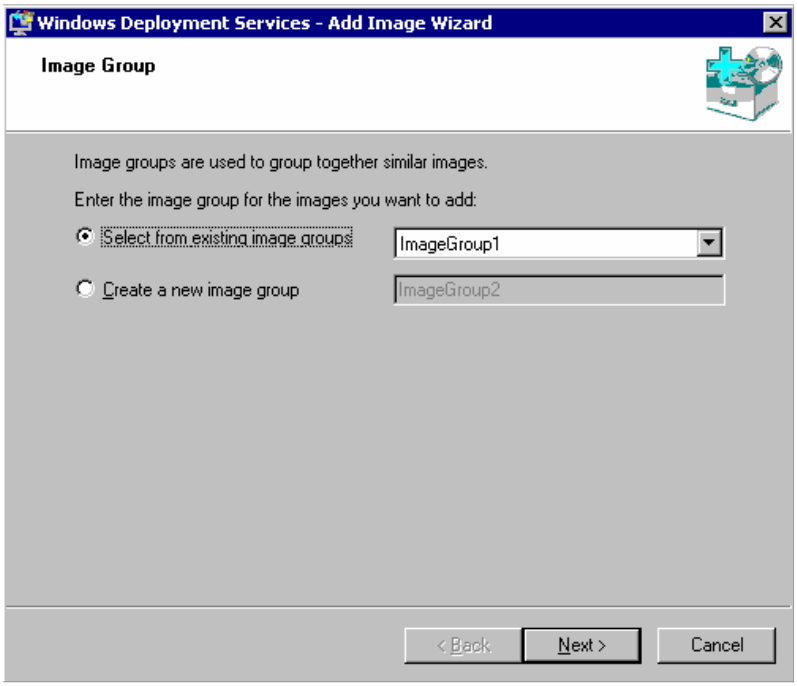
Ayant au moins une image de *boot* disponible sur notre serveur *WDS*, il est maintenant possible d'ajouter des images *WIM* de *Windows Vista* pour leur déploiement.

Nous allons ici ajouter l'image d'installation de base de *Windows Vista*, disponible sur le DVD d'installation de *Windows Vista*.

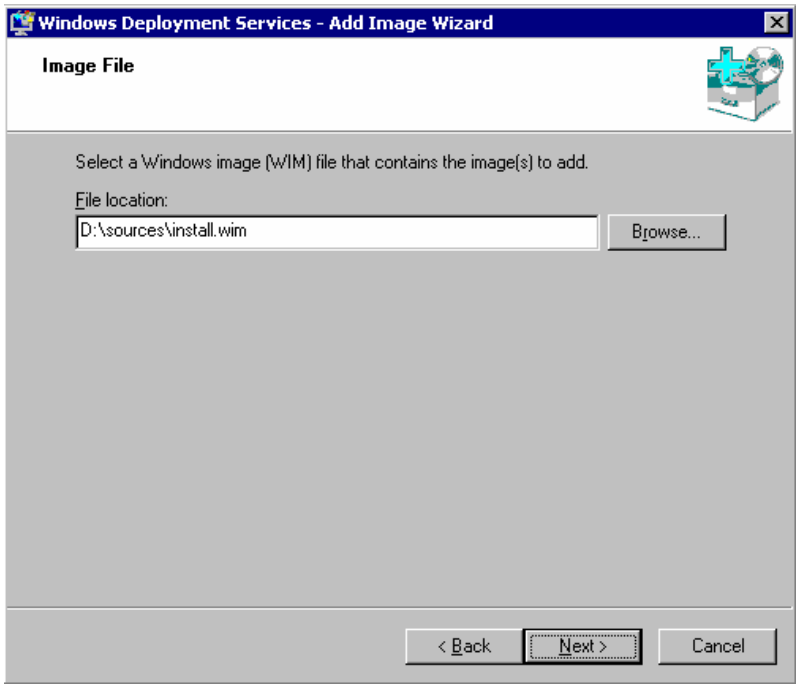
Effectuer un clic droit sur *Install Images*, puis *Add Install Image*



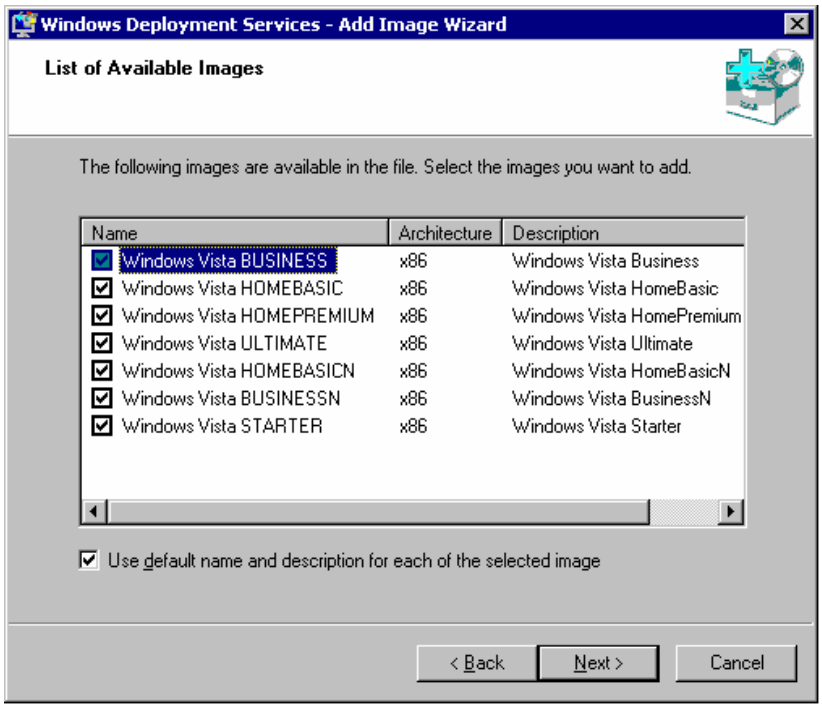
Sélectionner un groupe d'images (si déjà existant) autrement en créer un nouveau.



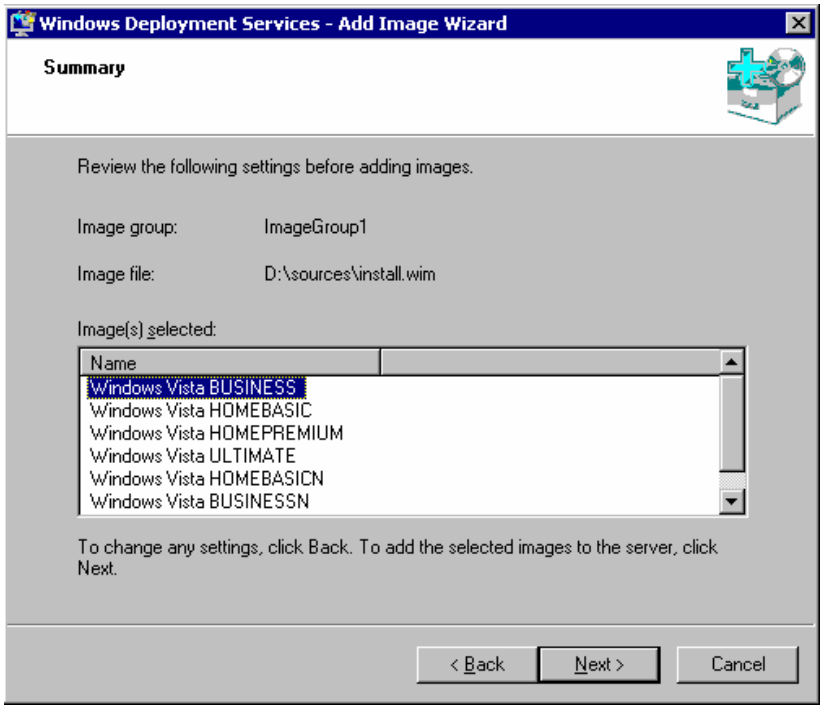
Sélectionner le fichier image de *Windows Vista* (nommé **`install.wim`**) se trouvant sur le DVD de *Windows Vista*.



Notre serveur *WDS* nous affiche les différentes images contenues à l'intérieur de ce fichier.
Sélectionner la ou les images souhaitées.



Une fenêtre de résumé est affichée, en cliquant sur *Next* les différentes opérations demandées seront effectuées.



1.5.6.2.5 Automatiser une installation avec un fichier *Unattend.xml*

Pour automatiser une installation, un fichier *Unattend.xml* peut être créé à l'aide de *Windows System Image Manager*.

Nous allons ici automatiser cette installation en incluant dans notre fichier *Unattend.xml* la clé de *Windows Vista*, créer une partition et la formater puis pour finir ordonner l'installation sur la partition c:\

Les diverses manipulations sont disponibles en annexes.

1.5.6.2.6 Personnaliser une installation avec un fichier *Unattend.xml*

Un serveur *WDS* utilise 2 fichiers *Unattend.xml*, un pour la phase de configuration 1 qui est *Windows PE*, et un autre pour les phases de configuration 2 à 7.

Nous nous demandons alors pourquoi avoir effectué cette séparation ?

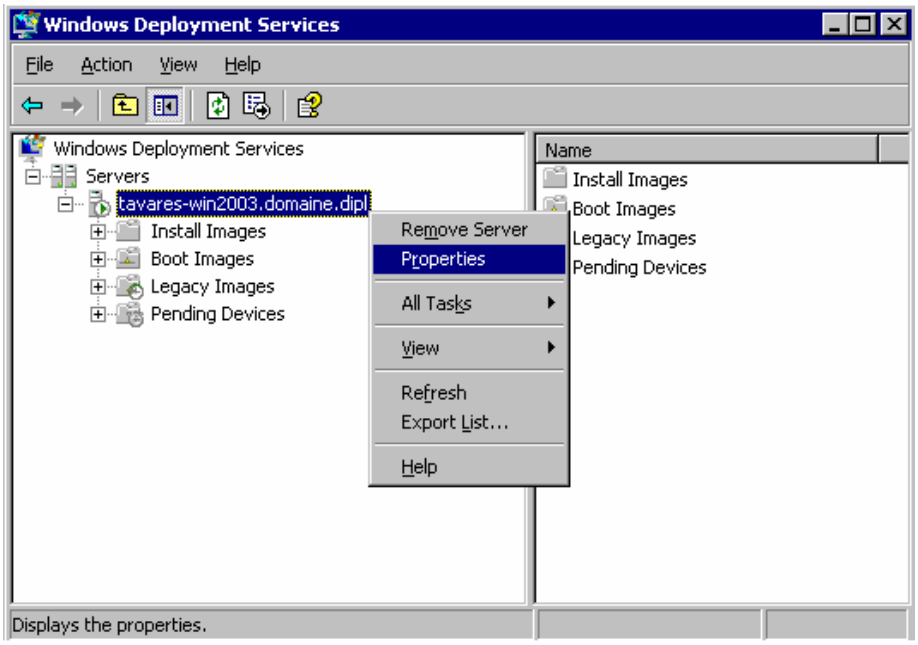
Sur le serveur *WDS*, nous pouvons spécifier un fichier *Unattend* pour la première phase de configuration (*Windows PE*), ce fichier sera utilisé pour toutes les architectures de même type (x86, x64).
Ceci nous permet par exemple de formater toujours la même partition, se connecter automatiquement à notre serveur *WDS* en spécifiant notre nom d'utilisateur dans notre domaine ainsi que son mot de passe, choisir automatiquement l'image à installer.

L'autre fichier *Unattend.xml* contenant les phases de configuration 2 à 7 doit être associé à une image spécifique se trouvant sur notre serveur *WDS*.

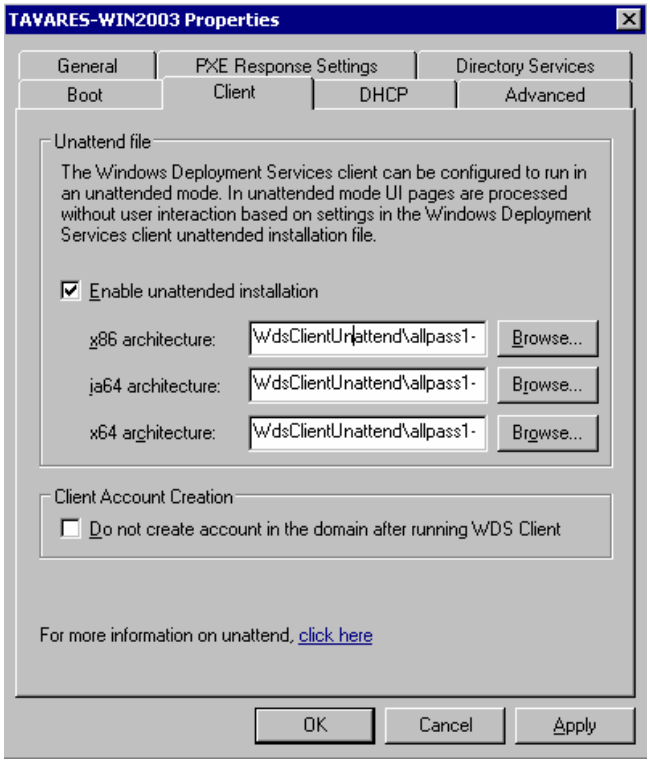
En séparant donc un fichier *Unattend.xml* en 2 parties (phases 1 et phases 2 à 7), nous pouvons effectuer toujours les mêmes actions dans la phase 1 et différentes actions dans les phases 2 à 7 en fonction de l'image à installer.

Pour associer ces différents fichiers dans notre serveur *WDS* :

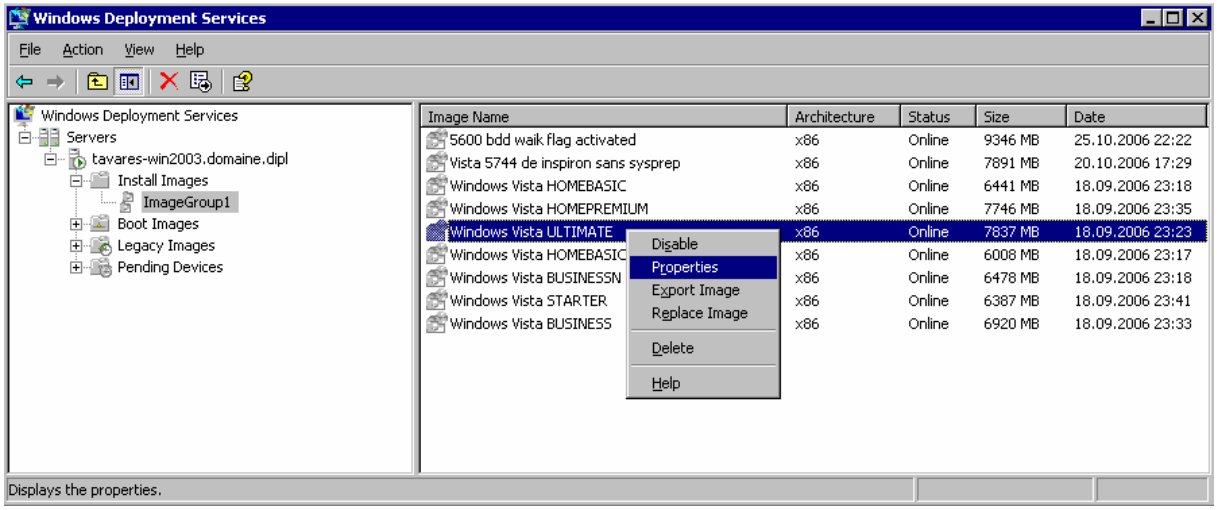
Ouvrir les propriétés de notre serveur *WDS*



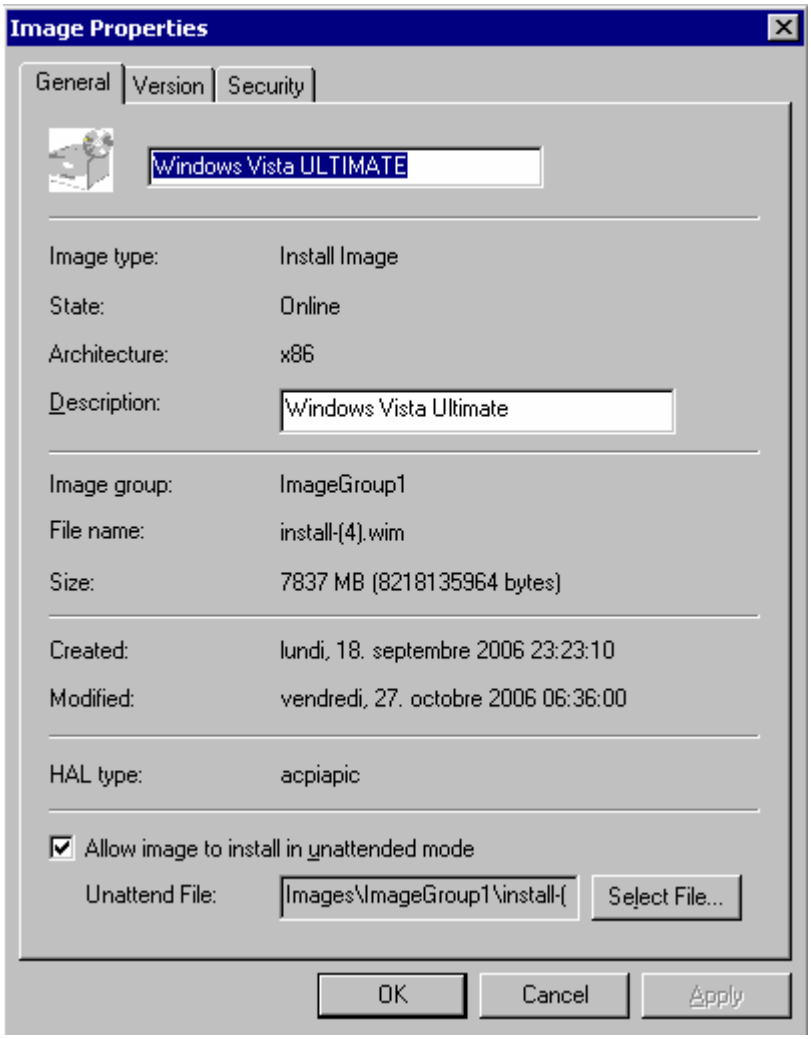
Sélectionner l'onglet *Client*,
Cocher la case ***Enable unattended installation***,
Ajouter le fichier *Unattend.xml* contenant les paramètres à configurer pour la **phase 1** (*Windows PE*) à l'architecture souhaitée :



Pour ajouter le fichier *Unattend.xml* contenant les paramètres à configurer durant les phases 2 à 7, **ouvrir les propriétés de l'image** sur laquelle on veut associer le fichier *Unattend.xml* :



Cocher la case *Allow image to install in unattended mode*, **associer** à notre image le **fichier *Unattend.xml* contenant les phases 2 à 7** à l'aide du bouton *Select File…*



Remarque :

Il existe une ligne de commande pour associer un fichier *Unattend.xml* à un pc spécifique. Cette commande est la suivante et doit être entrée sur le serveur *WDS* :

wdsutil /set-device /device:*computername* /ID:*MAC_ADDRESS* /WdsClientUnattend:path

Cependant, après plusieurs tests, je n'ai pas réussi à faire fonctionner cette commande.
Il y avait une erreur en utilisant le paramètre **/device:** suivi du nom d'ordinateur (sur lequel on aimerait utiliser le fichier *Unattend.xml*).

1.5.6.2.7 Installer une image sur un pc distant en utilisant notre serveur *WDS*

Pour qu'un pc client (sur lequel on veut installer une image) puisse utiliser notre serveur *WDS*, il doit être capable de *booter* en mode *PXE*.

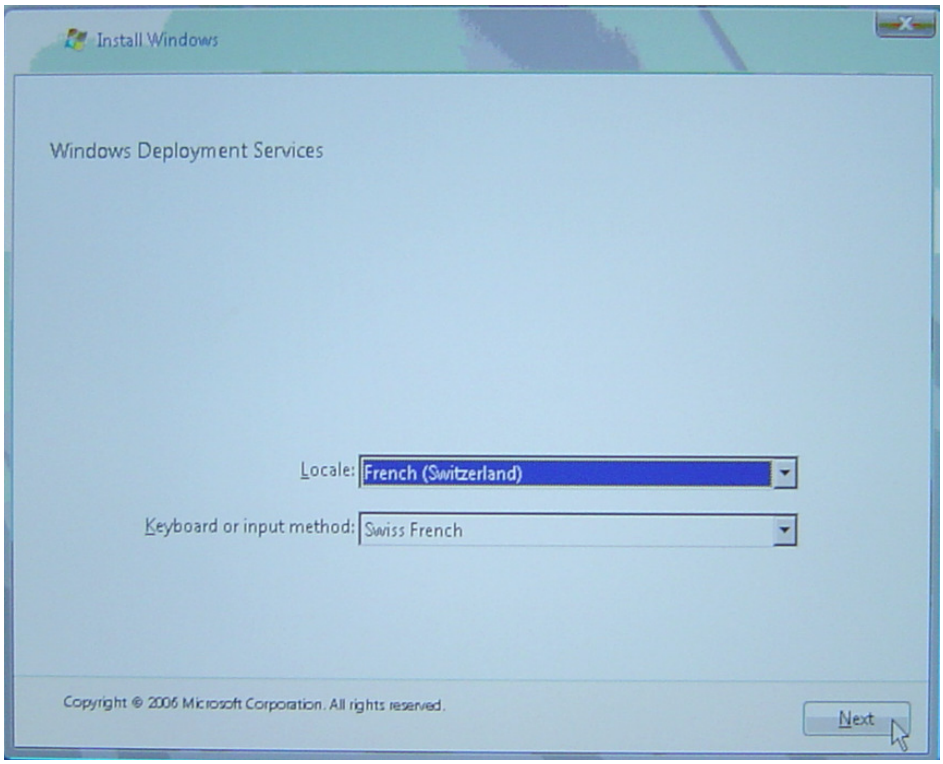
Il faut tout d'abord configurer le BIOS du pc client afin de pouvoir utiliser *PXE*.

Sur le pc DELL Optiplex gx260 utilisé au laboratoire, la sélection de *boot* se fait en appuyant F12 au démarrage.

La configuration *PXE* complète de ce pc est traitée en annexe.

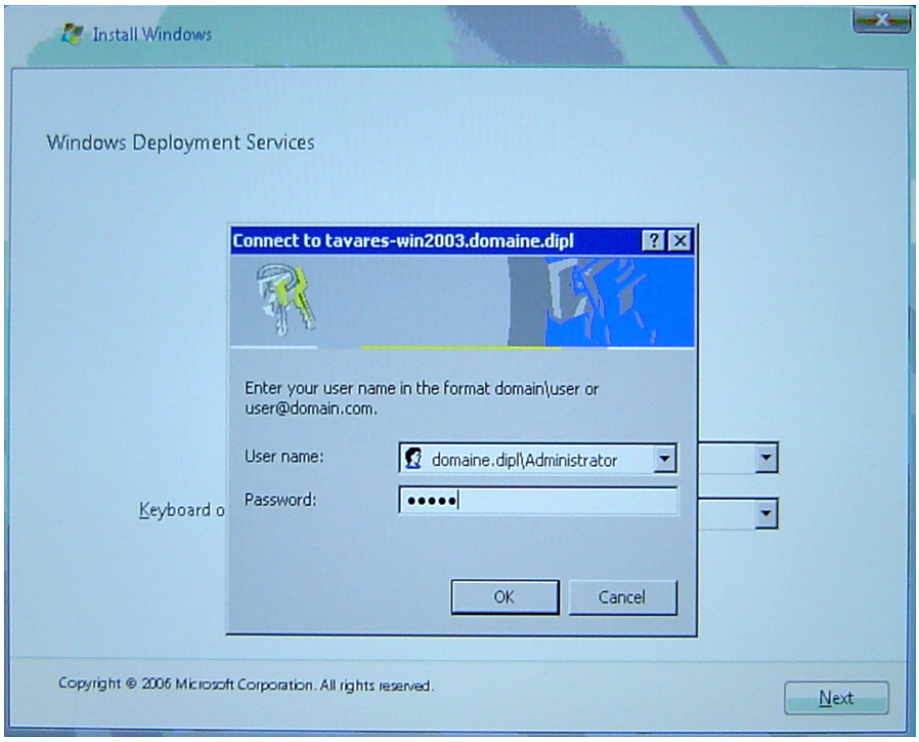
Comme je l'ai déjà précisé précédemment, l'image de *boot* de *Windows PE* se trouvant sur le DVD d'installation de *Windows Vista* doit être présente sur le serveur *WDS*.

Sélectionner cette image, l'écran suivant s'affichera après environ 2 minutes :

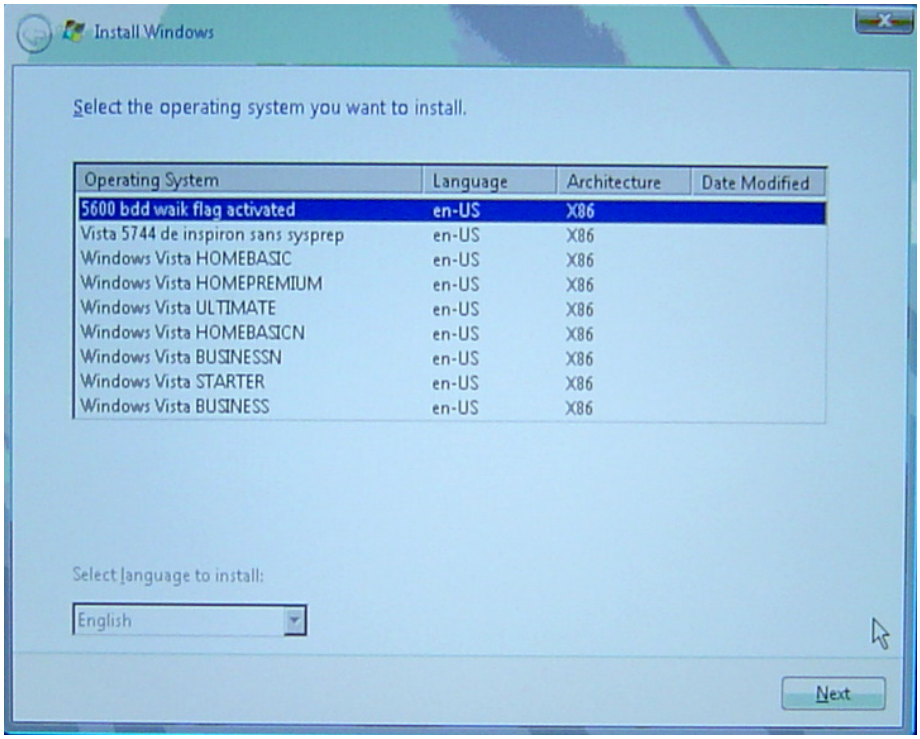


Sélectionner la langue (anglais, français, etc.) puis la langue du clavier.
Il nous faudra ensuite entrer notre nom d'utilisateur (du domaine) et le mot de passe.

Remarque : Le nom de domaine "domaine.dipl" a été utilisé au laboratoire.



Les différentes images présentes sur le serveur sont affichées.
Il est possible d'afficher seulement certaines images pour un compte bien précis.
Par exemple si une secrétaire désire réinstaller son poste, il faudrait que son compte ait uniquement la permission d'installer les images créées pour les secrétaires !



Il ne reste plus qu'à sélectionner l'image à installer.

1.6 Donner un cours sur les images *Vista* et leur déploiement en entreprise ?

Afin de donner un cours d'environ une journée sur les images *Vista*, ainsi que leur déploiement en entreprise, je propose les laboratoires suivants ainsi que les équipements et installations nécessaires afin d'effectuer correctement ces laboratoires.

Pour chaque personne, il faudra avoir un poste installé au préalable avec *Windows Vista*, *BDD 2007* et *Windows AIK* (donner les liens de téléchargement de ces divers outils).
Ce poste devra être équipé d'une carte réseau supportant le *boot PXE*. Son BIOS doit être correctement configuré pour ceci.

De plus il leur faudra à chacun un poste installé au préalable avec *Windows Server 2003*, y inclure un serveur *DHCP*, *DNS*, *Active Directory, RIS* (nécessaire afin d'effectuer la mise à jour vers *WDS*) ceci afin de gagner du temps.

Chaque personne devra donc avoir un réseau totalement séparé (il n'est pas nécessaire de leur donner accès à Internet).

En tant que bref descriptif pouvant être mis sur le site de l'école, je propose les points suivants :

Création d'images *Vista* et déploiement
- Les nouveaux concepts d'images et le nouveau format dans *Windows Vista*
- Les outils proposés par *Microsoft* et leur utilisation
- Inclure plusieurs images au sein du même fichier, images indépendantes de la configuration *hardware*
- Gérer les images sans devoir les installer (mode *Offline*)
- Création d'un CD de pré-installation
- Automatisation d'une installation
- Personnalisation d'une installation
- Déploiement avec un serveur *WDS*
- Comparaison avec les autres alternatives disponibles sur le marché

Plus en détail, voici les divers points que je propose afin de donner ce cours :

- Commencer par **définir la problématique**, qu'est ce qu'une image, qu'est ce qu'un déploiement.

- Définir le nouveau **format d'image *WIM***, ses propriétés, définies dans le paragraphe 1.2.3 page 12. Mentionner que *Windows Vista* est installé à partir d'une image *WIM* dans le DVD d'installation *Vista*.

- **Présenter l'outil *ImageX***, permettant de capturer, gérer, installer des images *WIM*.
  Expliquer qu'il n'est pas possible de capturer une installation pendant son fonctionnement. La capture est seulement possible quand le système n'est pas utilisé.
  Indiquer le temps nécessaires pour effectuer une capture avec les diverses compressions possibles, puis la comparaison entre la taille du disque et la taille de l'image créée. Un tableau est disponible en page 72, paragraphe 1.8.3.

- Expliquer brièvement ***Windows PE***, que cet environnement de pré-installation permet de capturer (et installer) les images à l'aide de l'outil *ImageX*, puis proposer de créer un CD *bootable* de *Windows PE* personnalisé en incluant *ImageX* (labo nécessitant environ 15 minutes)

- Faire graver cette ISO si les postes disposent d'un graveur CD. Si tel n'est pas le cas, distribuer ces CD de *boot Windows PE* gravés préalablement aux différentes personnes, et leur assurer que le contenu est rigoureusement identique à l'ISO de *Windows PE* qu'ils viennent de créer.

- Expliquer la **problématique d'une installation de référence** (adresses IP, SIDs attribuées au pc, informations uniques à la machine, pour cela expliquer les points suivants, éventuellement consulter *Sysprep*, paragraphe 1.9 pour de plus amples informations :
  Lors de la création d'une image pour le déploiement, nous ne souhaitons pas garder les informations uniques au poste de référence dans l'image.
  Il nous est possible de les retirer à l'aide de ***Sysprep***.
  Expliquer qu'il existe un mode spécial qui est le mode *Audit*, que ce mode permet d'installer rapidement une application sans avoir à effectuer la phase de *Windows Welcome* (qui nécessite environ 10 à 15minutes, permet donc d'obtenir le contrôle de *Windows Vista* plus rapidement !). Indiquer que ce mode n'est pas souvent utilisé, mais tout de même le mentionner pour permettre une meilleure compréhension par la suite de *Windows System Image Manager* et des différentes phases de configuration.
  Utilisation de *Sysprep* aux personnes, exécuter la commande de base (**`sysprep /oobe /generalize`**), le pc redémarrera automatiquement (labo nécessitant environ 10 minutes y compris le point suivant).

- (Suite du labo), *Booter* immédiatement sous *Windows PE* à l'aide du CD qui vient d'être créé.
  C'est à ce moment précis que la capture d'installation doit être effectuée, mais ne pas créer une capture de l'installation complète ! (trop long)
  Montrer les **lignes de commande à effectuer avec *ImageX*** afin de capturer une installation, cependant étant donné que la capture d'une installation complète nécessite environ 20 minutes, il est peut-être préférable de capturer un dossier (contenant quelques fichiers) plutôt qu'une installation complète *Vista* !
  Ceci montrera le principe de l'utilisation d'*ImageX*.

- Etant donné que nous avons utilisé *Sysprep* sur notre système, lors du prochain *boot* sur *Windows Vista*, nous aurons en premier lieu ***Windows Welcome*** qui se lancera et qui fera une inspection la configuration matérielle de l'ordinateur afin d'optimiser *Windows Vista* pour cette machine.
  Imaginons que nous avons installé notre image *Vista* sur un pc quelconque de l'entreprise et que nous démarrons ce pc. Les personnes verront alors le lancement de *Windows Welcome*, il est peut-être utile de voir au moins une fois ce lancement, ceci facilitera la compréhension (ce labo nécessite 10 à 15 minutes)

- Sur *Vista*, utiliser *ImageX* avec la commande **`imagex /info`**, pour consulter les informations de l'image précédemment créée.
  Utiliser la commande **`imagex /mountrw`**, qui permet de monter une image, consulter et modifier son contenu.
  Le modifier par exemple en ajoutant un fichier texte.
  Enregistrer les modifications de l'image à l'aide de la commande **`imagex /unmount /commit`**.
  Noter que cette dernière commande enregistre les modifications directement dans l'image *WIM*, il n'est pas nécessaire d'effectuer une nouvelle capture !
  (labo nécessitant environ 15 minutes)

- **Inclure plusieurs images à l'intérieur du même fichier *WIM***.
  Ayant un fichier *WIM* composée de l'image d'un dossier avec des fichiers texte, ouvrir l'explorateur *Windows* et créer un nouveau fichier texte à l'intérieur du dossier en question.
  Effectuer une capture de ce dossier à l'aide de la commande **`imagex /capture`**, cette capture sera effectuée sur le même fichier *WIM*.
  Comparer les différences de taille, noter que la taille n'a pas doublé ! (une seule instance par fichier à l'intérieur du fichier *WIM*).
  Consulter le scénario 2 pour les diverses commandes exactes.
  (labo nécessitant environ 10 minutes)

- Présenter ***Windows System Image Manager***, outil permettant de créer un fichier de configuration et d'automatisation d'une installation de *Windows Vista*.
  Ce fichier est appelé fichier de réponse (ou *answer file* en anglais), c'est un fichier XML.
  Faire utiliser cet outil graphique aux différentes personnes.
  Suivre le scénario disponible en annexes, en tout cas suivre les parties d'automatisation (entrer la clé *Windows Vista*, créer une partition, formater la partition, spécifier la partition sur laquelle *Windows Vista* sera installé), puis paramétrer les divers *components* pour accroître la sécurité (bloquer les fenêtres de *popups* dans *internet explorer*, configurer l'assistance à distance, configurer (éventuellement) le component du *firewall* MPSSVC, définir un mot de passe administrateur).
  Ensuite sauver le fichier de réponse ainsi créé, puis l'ouvrir et vérifier que le mot de passe administrateur est bien chiffré !
  (labo nécessitant environ 30 minutes)

- Maintenant que nous savons créer des images, les modifier, inclure plusieurs images dans le même fichier *WIM* et créer des fichiers de réponse, nous allons maintenant **installer et configurer un serveur *WDS* pour le déploiement d'images**.
  Faire installer aux personnes la mise à jour *RIS* vers *WDS*.
  Configurer le serveur *WDS* de A à Z comme indiqué dans le scénario 3.

  Faire installer sur leur serveur *WDS*, par exemple, les images de base *Vista* contenues dans le DVD d'installation de *Windows Vista*.
  Montrer comment associer sur le serveur *WDS* un fichier *Unattend.xml* à une image (voir scénario 3).
  Noter que le fichier *Unattend.xml* est divisé en 2 parties sur le serveur *WDS*, expliquer pourquoi avoir fait cette séparation.
  (labo nécessitant environ 45 minutes)

- Faire **installer 1 fois *Windows Vista* avec le DVD de base**, pour que les personnes remarquent toutes les informations demandées lors d'une installation traditionnelle.
  Leur faire faire ensuite quelques **déploiements avec le serveur *WDS***, remarquer l'efficacité du **fichier de réponse** associé à une image.
  Expliquer brièvement PXE, qui permet de *booter* sur le réseau. Pour cela il est nécessaire de disposer d'un serveur DHCP et DNS (déjà installés préalablement sur leur serveur respectifs).
  (labo nécessitant environ 1 heure)

En option :

- Eventuellement donner un cours sur les possibilités qu'offre *Windows PE* en matière de récupération système, notamment la résolution des problèmes du *boot* de *Vista* (avec l'outil *Startup Repair*).

Nous arrivons donc à un temps total d'environ 3h20 de labo (sans l'option ci-dessus), plus le temps nécessaire aux présentations orales.
Ceci devrait pouvoir se faire en une journée.

1.7 Windows PE

1.7.1 Qu'est ce que *Windows PE* ?

Windows PE (*Windows Pre-installation Environment*) est un outil de démarrage système.

Cet outil a été conçu afin de remplacer MS-DOS comme environnement de pré-installation et pour faciliter les déploiements personnalisés à grande échelle de *Windows Vista*.

Windows PE est une version très *light* (légère, simplifiée) de *Windows Vista* (elle a été créée sur les mêmes bases que *Windows Vista*).

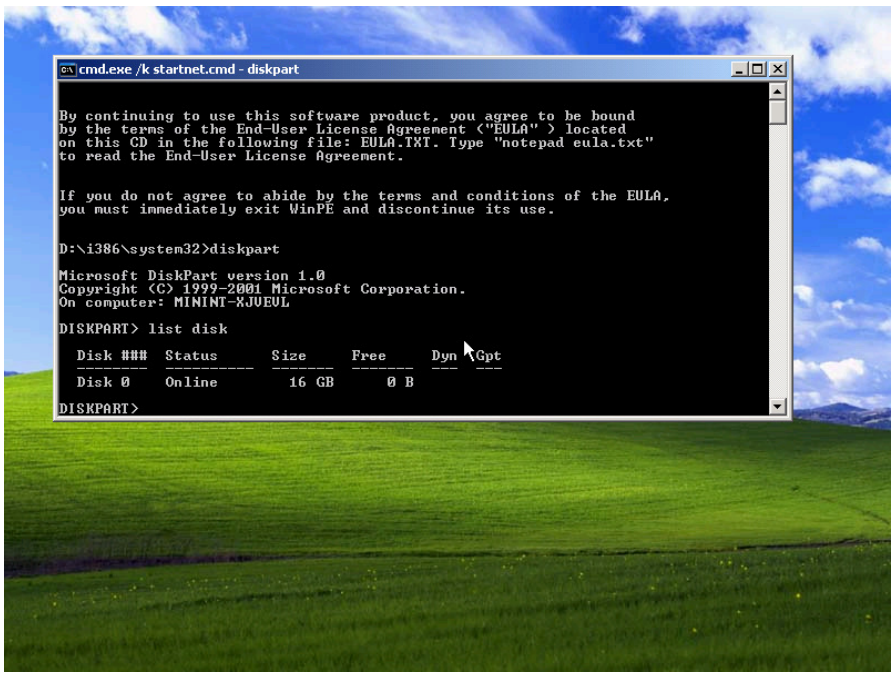
La version de base de *Windows PE* fait une taille de 154MB.

Windows PE permet utilisateurs d'ajouter des pilotes, configurer *Windows Vista* hors connexion (mode *Offline*).
En parlant de mode *Offline*, il convient aussi de parler du mode *Online.* Ce dernier signifie que le système d'exploitation à été démarré et que l'on effectue les modifications directement au sein de ce système.

Remarque : *Windows PE* qui est un sous-ensemble de *Windows Vista*, à été conçu spécialement pour *Windows Vista* et ne peut donc pas être acquis séparément.

1.7.2 A quoi ressemble-t-il ?

L'environnement *Windows PE* ressemble à la figure suivante :



En créant un *Windows PE* personnalisé, nous obtiendrons un environnement dans ce style, avec uniquement une fenêtre d'invite de commande qui se lance au démarrage.

1.7.3 Versions disponibles

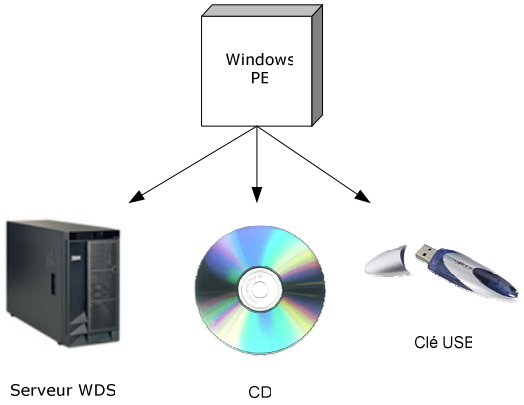
Les différentes versions disponibles de *Windows PE* sont les suivantes :

- *Windows PE* x86 (32 bits)
- *Windows PE* x64 (64 bits)

1.7.4 Comment s'exécute-t-il?

Windows PE s'exécute entièrement en mémoire RAM.

Selon *Microsoft*, il peut être lancé à partir d'un CD, d'une clé USB ou d'un réseau.



J'ai testé les scénarios suivants :

- **Lancement de *Windows PE* à partir d'un CD**.
  Un CD personnalisé de *Windows PE* a été créé (voir scénario1 en page 21 ou annexes).
  Il ne faut cependant pas oublier au préalable de configurer le BIOS du pc afin d'accepter de *booter* sur CD.

- **Lancement de *Windows PE* à partir du réseau**.
  Un serveur *WDS* (*Windows Deployment Services*) a été installé sur un *Windows Server 2003*.
  Une image de *Windows PE* a été ajoutée au serveur *WDS* afin de pouvoir être déployée sur le réseau.
  Le pc client (pc sur lequel on veut lancer *Windows PE* à travers le réseau) à été démarré en *PXE* afin de pouvoir se connecter sur le serveur *WDS*.

- **Lancement de *Windows PE* à partir d'une clé USB**.
  L'installation de *Windows PE* sur une clé USB est décrite dans l'aide de *Windows AIK*, cependant, il faut que le poste soit capable de *booter* sur une clé USB et de plus, faire attention à la vitesse de transfert de la clé USB! (en USB1.0 le transfert sera lent).

Toutes ces méthodes fonctionnent parfaitement.

Remarque : Son exécution entièrement en mémoire RAM permet d'utiliser *Windows PE* sur des ordinateurs qui ne sont pas encore équipés d'un disque dur formaté ou d'un système d'exploitation installé.
De plus il est possible d'utiliser un 2eme cd avec des drivers ou logiciels.

1.7.5 Que permet-t-il de faire?

Windows PE permet de :

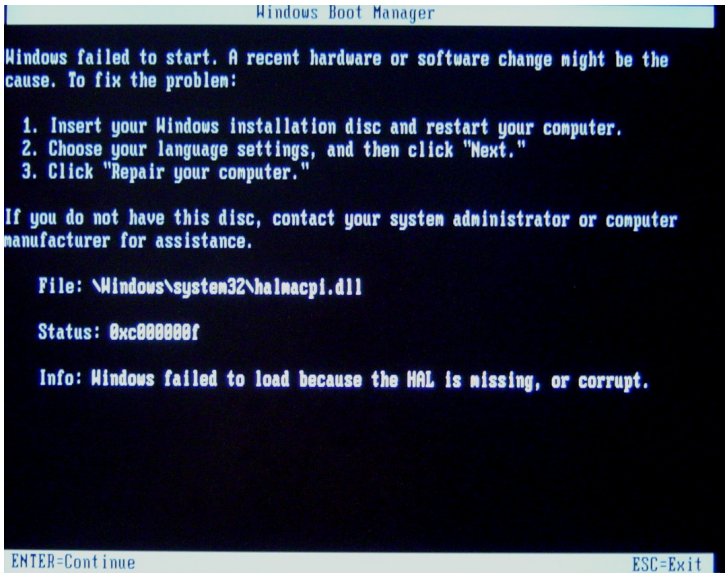
- **Installer Windows Vista**, *Windows PE* s'exécute lors de chaque installation de *Windows Vista*, les outils graphiques tels que *Windows Setup* collectent les informations de configuration lors d'une installation (*Windows Setup* s'exécute sous *Windows PE*).

  Exemple : L'installation *Windows Vista* s'exécutant sous *Windows PE*

- **Résoudre les problèmes**, lorsque *Windows Vista* ne parvient pas à démarrer correctement, par exemple en raison d'un fichier système corrompu, *Windows PE* peut être démarré à l'aide du DVD d'installation de *Windows Vista* ce qui nous permettra d'utiliser des outils de dépannage et de diagnostic.

  Test effectué : *Windows Vista Ultimate*, *BDD 2007* et *Windows AIK* ont étés installés sur un pc.
  J'ai démarré ce pc sous *Windows PE* puis supprimé toutes les dll contenues dans le répertoire **`C:\Windows\System32`** (total de 1601 dll).

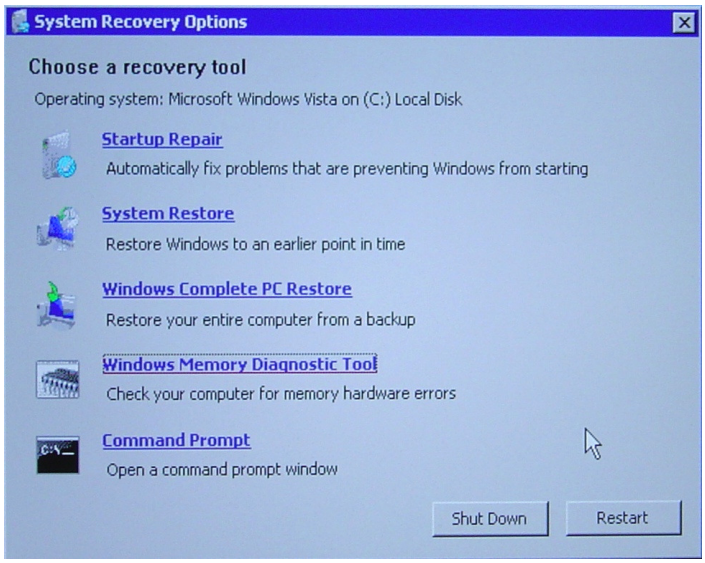
Le pc a ensuite été redémarré, l'écran suivant c'est affiché :



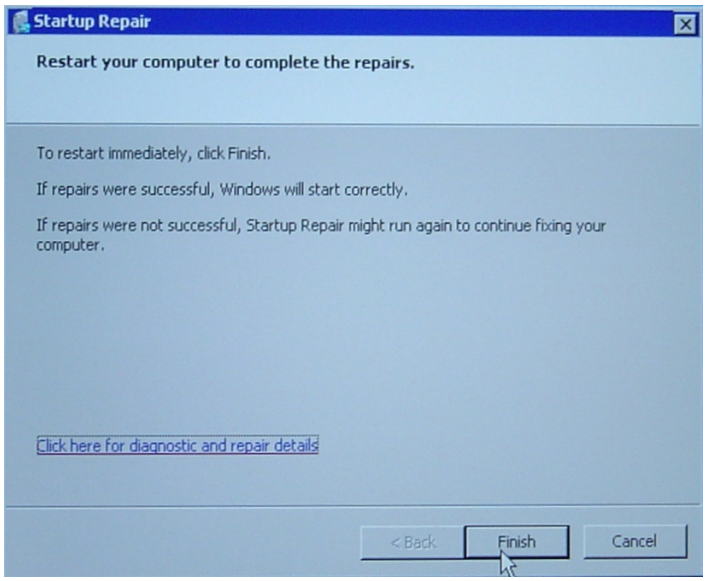
Windows Boot Manager nous informe qu'il y a des fichiers manquants ou corrompus, et que pour résoudre ce problème il faut utiliser le DVD d'installation de *Windows Vista*.

J'ai donc redémarré le pc et *booté* sur le DVD d'installation de *Vista*, qui a affiché l'interface suivante :

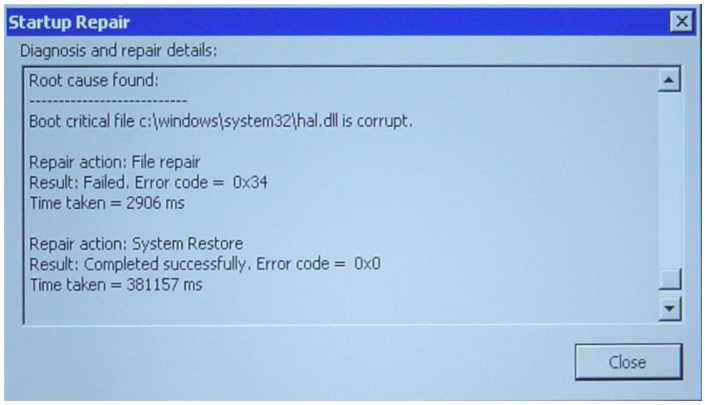
Sélectionner *Repair your computer*, le menu *System Recovery Options* s'affiche:



En sélectionnant *Startup Repair*, cet outil répare les dll défectueuses ou manquantes.



Voici un extrait des *logs* qui ont été créés, ces *logs* peuvent être consultés en cliquant sur *Click here for diagnostic and repair details*.



Remarque : Il faut lancer plusieurs fois *Startup Repair*, à la fin de chaque exécution, le pc doit essayer de redémarrer sous *Windows Vista*. Si le pc n'arrive pas à démarrer normalement, un rapport sera généré pour *Startup Repair*, il nous faudra alors *booter* sur le DVD d'installation de *Vista* et lancer *Startup Repair* qui essayera de réparer l'installation.

Tests :

Ce test a été effectué 3 fois sur la même installation, à la fin de chaque test, l'image de notre *Vista* (personnalisé avec *BDD 2007* et *Windows AIK*) a été réinstallée. De cette manière nous sommes exactement dans les mêmes conditions que lors du test précédent.

- Lors du premier test, *Windows Vista* a bel et bien démarré mais il lui manquait les pilotes de la carte graphique.

- Lors du second test, *Startup Repair* n'a pas réussi à réparer notre installation.

- Lors du dernier test, *Windows Vista* a démarré et tous les pilotes étaient présents, les 1601 dll de départ on bien été restaurées.

Conclusion :

Cet outil de restauration est une très bonne idée, cependant j'ai eu 3 résultats différents pour le même test effectué (sur la même machine, avec la même installation), ce qui prouve que cet outil n'est pas encore tout à fait au point.

De plus, le plus souvent nous n'avons pas une perte de toutes les dll en même temps (ce test a été fait dans le pire des cas).

Je recommande donc son utilisation qui peut s'avérer très efficace lors de quelques dll manquantes. Si le problème ne peut être résolu de cette manière, il est toujours possible d'effectuer une réinstallation à l'aide d'une image (créée au préalable).

Il est conseillé de garder les différents drivers à proximité afin de pouvoir les réinstaller si besoin.

- **Récupérer une installation *Vista***, *Windows PE* peut être utilisé afin de reformater un disque dur et de réinstaller *Windows Vista* avec les pilotes, paramètres et applications d'origine.

  Cette possibilité de récupération est surtout destinée aux utilisateurs normaux, qui ont créé une image de leur installation avec l'outil *Backup and Restore Center*, ou en effectuant un point de restauration système.

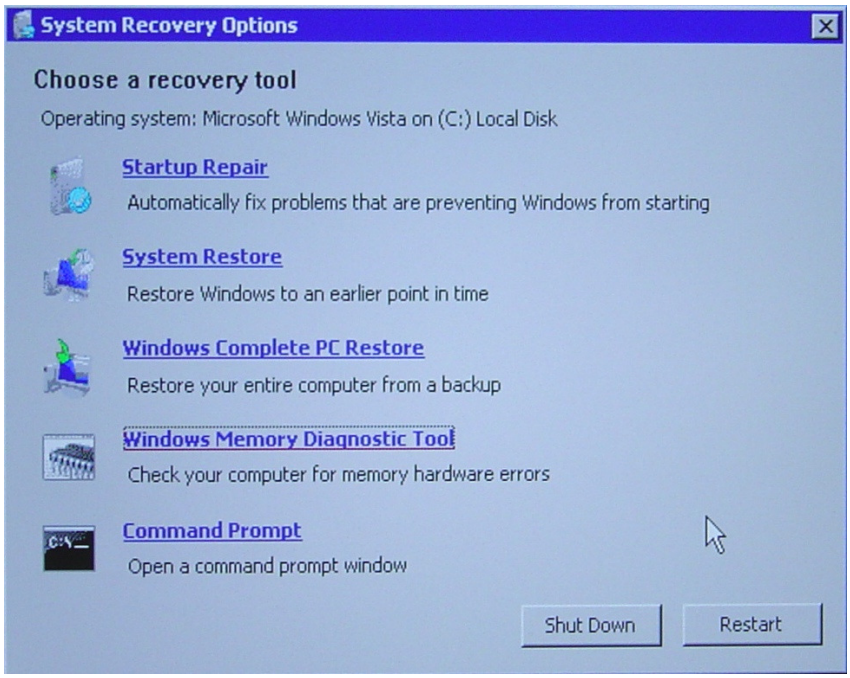
  Deux outils sont disponibles pour récupérer une installation Vista :

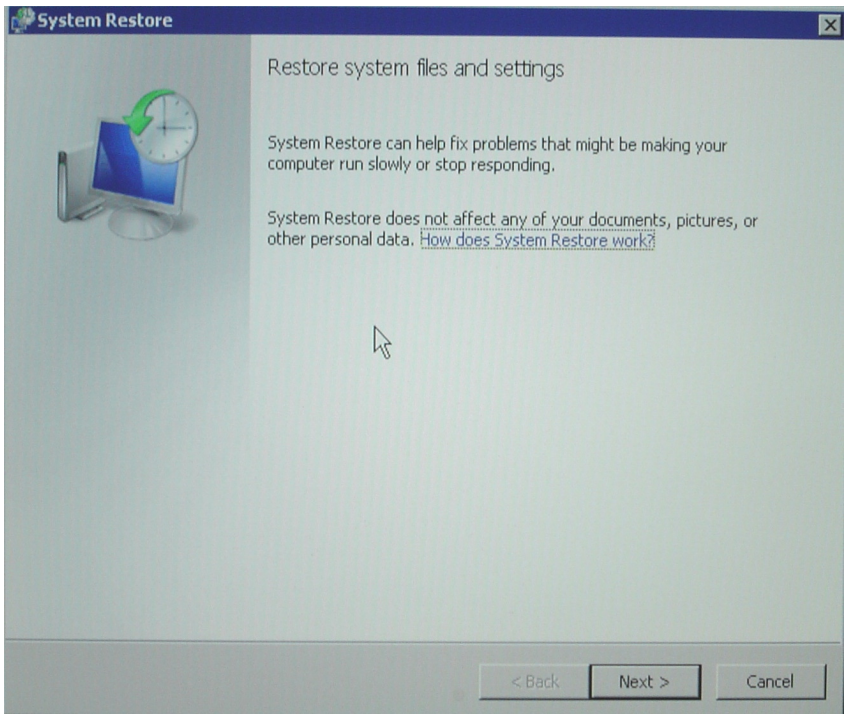
  - *System Restore*, qui permet de restaurer une installation à l'aide d'un point de restauration système.

  - *Windows Complete PC Restore*, qui permet de réinstaller une image créée avec *Backup and Restore Center*.

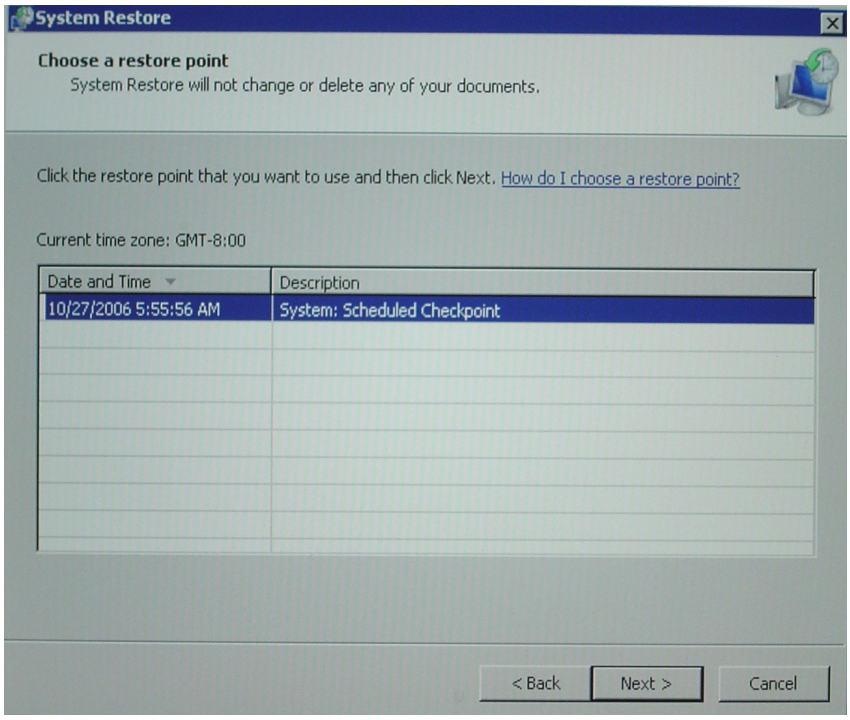
Outil *System Restore*

Sélectionner *System Restore* dans le menu *System Recovery Options*

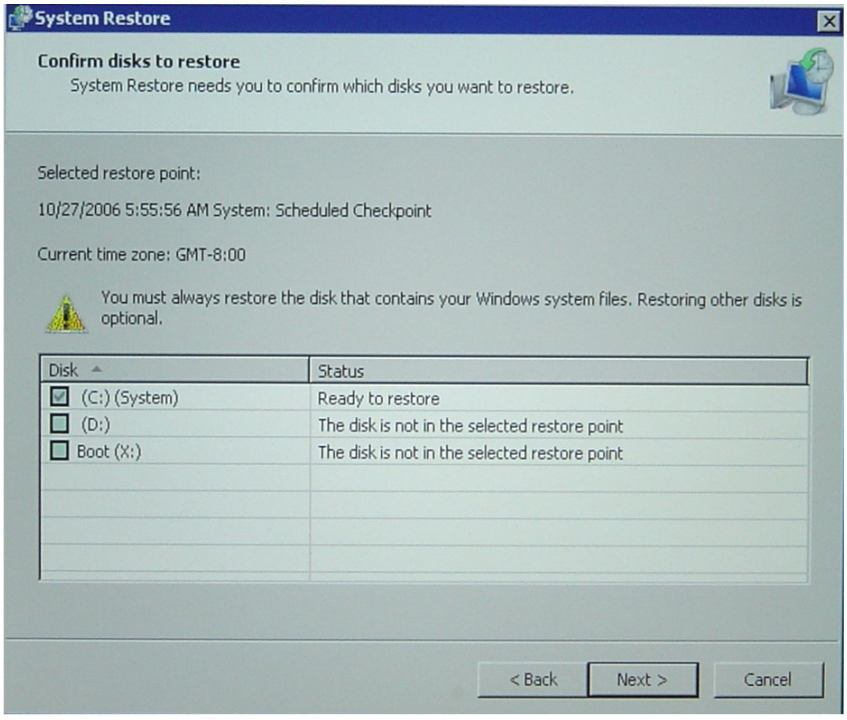




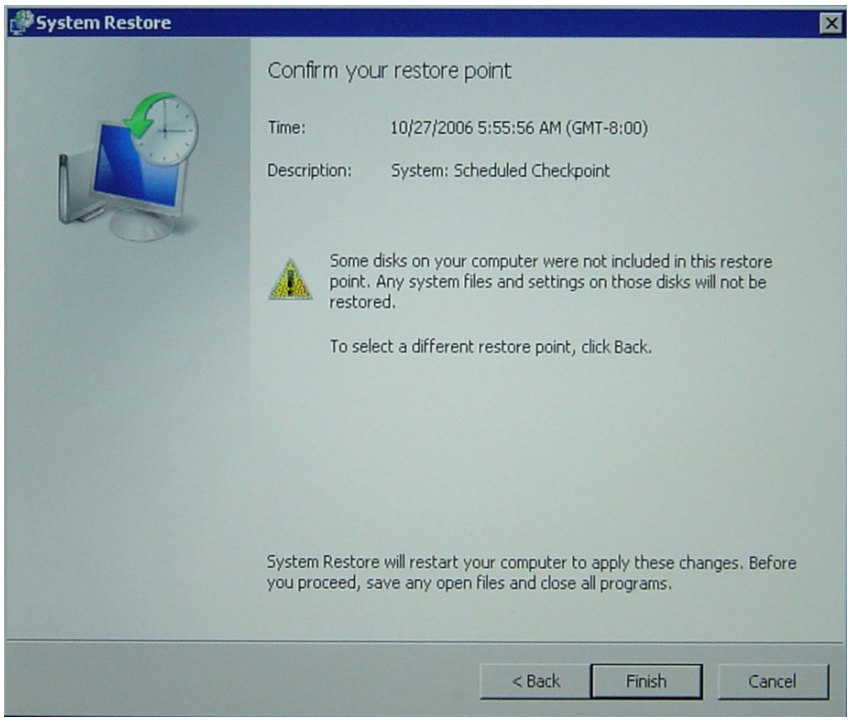
Sélectionner le point de restauration que l'ont veut utiliser



Confirmer les disques (ou partitions) à restaurer

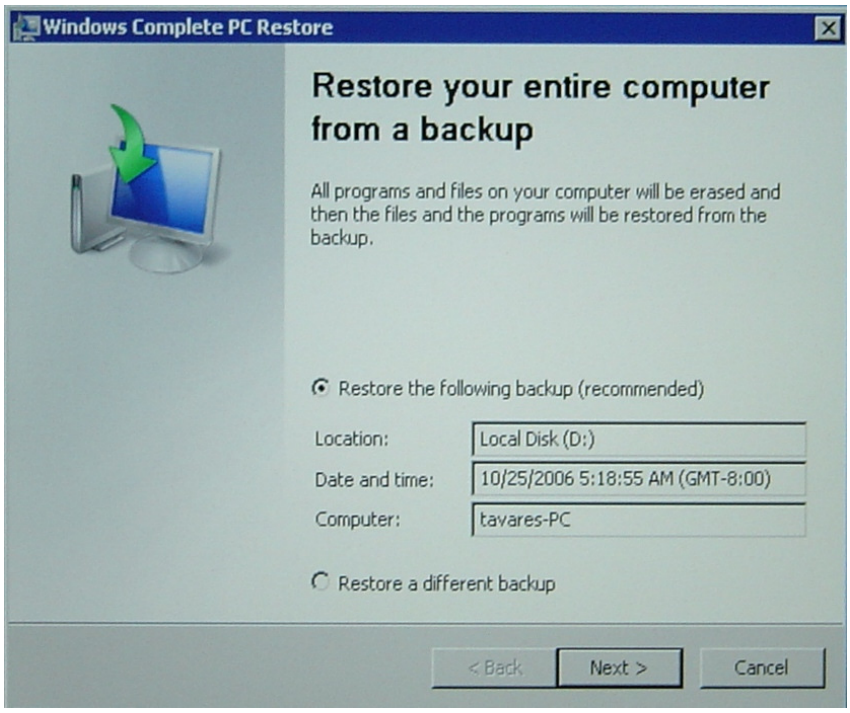


Confirmer les actions à effectuer

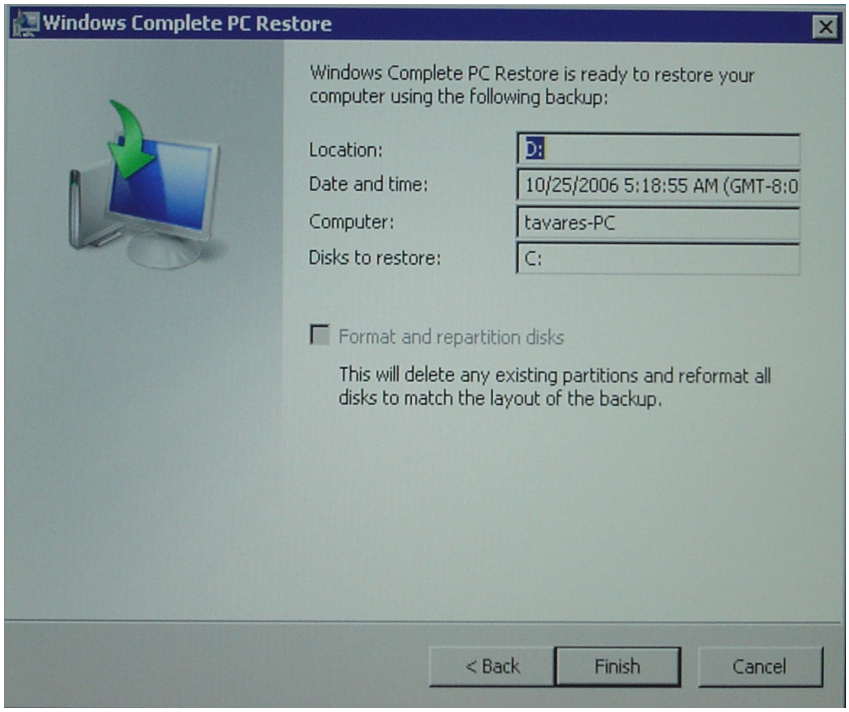


Windows Complete PC Restore

Sélectionner le *backup* (image) à restaurer



Confirmer la restauration



1.7.6 Limitations de *Windows PE*

Les limitations de *Windows PE* sont les suivantes :

- Minimum de 256MB de RAM, afin de pouvoir copier *Windows PE* en mémoire RAM et disposer d'un peu de RAM restante afin de lancer des programmes tiers.
  De toute manière il faut environ 1 GB de mémoire RAM afin de pouvoir utiliser *Vista* (recommandé par *Microsoft*).
  J'ai effectué une installation de *Vista* sur un pc équipé de 768MB de RAM, l'installation s'est effectuée sans problème et *Windows Vista* fonctionnait relativement bien.

- *Windows PE* ne peut pas être installé sur une disquette.

- *Reboot* (redémarrage) toutes les 24h, pour empêcher l'utilisation de *Windows PE* comme système d'exploitation.

- Il n'est pas possible pour un pc du réseau d'accéder aux fichiers d'un ordinateur qui exécute *Windows PE* (Service serveur non disponible).

- *Windows PE* ne supporte pas d'autres protocoles à part TCP/IP et NETBIOS.

- Requiert un affichage compatible *VESA* (*Video Electronics Standard Association*) pour utiliser la plus grande résolution pouvant être détectée.
  Si *Windows PE* n'arrive pas à déterminer les paramètres d'affichage, son affichage par défaut sera de 640x480 pixels.

- Il est possible de créer des versions personnalisées de *Windows PE* sur *Windows XP Professional* et *Windows Server 2003*, mais pas sur *Windows XP Home*.

- Ne supporte pas *Microsoft .NET Framework*

- Les applications 16 bits ne fonctionnent pas sur *Windows PE* 32 bits, de même que les applications 32 bits ne fonctionnent pas sur *Windows PE* 64 bits.

De plus seulement un sous-ensemble des Win32 APIs est disponible dans *Windows PE*, ceci afin que *Windows PE* soit moins gourmand en MB.

La taille de *Windows PE* de base, fourni par *Microsoft* dans *Windows Automated Installation Kit* est de 153MB.

Les catégories suivantes de Win32 APIs ne sont pas comprises dans *Windows PE* :

- *Access Control*
- *NetShow Theater Administration*
- *OpenGL*
- *Power Options*
- *Printing and Print Spooler*
- *Still Image*
- *Tape Backup*
- *Terminal Services*

- *User Profile*
- *Window Station and Desktop*
- *Windows Management Instrumentation* (*WMI*)
- *Windows Multimedia*
- *Windows Shell*

Ces différentes APIs sont juste listées à titre d'information. Dans le cadre de ce projet, nous n'allons pas développer du code, elles ne sont donc pas importantes.

1.7.7 Outils disponibles dans *Windows PE*

Plusieurs *tools* (outils) sont disponibles dans *Windows PE*. Tous ces outils sont en lignes de commande.

Les outils disponibles sont les suivants :

- ***Diskpart***, permet de gérer les disques durs (formater, créer/supprimer des partitions, affecter une lettre à une partition ou un disque, etc.)
- ***Drvinst***, permet d'ajouter des drivers à une image de *Windows PE*.
- ***Factory***, permet de mettre à jour des drivers, installer des applications.
- ***Mkimg***, permet de copier vers un répertoire l'ensemble des fichiers nécessaires à *Windows PE*. Ceci nous permettra de personnaliser *Windows PE* et d'en créer éventuellement une image ISO.
- ***Netcfg***, permet de configurer l'accès réseau.
- ***Oscdimg***, permet de sélectionner le répertoire où l'on a personnalisé *Windows PE* et d'en créer une ISO.
- ***Sys***, permet d'utiliser *Windows PE* afin de préinstaller *Windows 95*, *Windows 98* et *Windows Millenium*. Pour pouvoir être installés, ces 3 systèmes ont besoin d'un secteur *boot* sur le disque dur, cet outil permet de créer ces secteurs *boot*.

Dans le cadre de ce travail, l'outil ***Oscdimg*** sera le seul utilisé, afin de créer un CD *bootable* de *Windows PE*.

Pour plus d'informations, consulter la rubrique *Windows Preinstallation Environment User's Guide* disponible dans l'aide de *Windows AIK*.

1.8 *ImageX*

1.8.1 Qu'est ce que *ImageX* ?

ImageX est un outil fourni par *Microsoft* permettant de créer, installer et gérer les images *WIM*.

Cet outil est disponible dans *Windows Automated Installation Kit*.

1.8.2 Commandes *ImageX*

Le tableau suivant montre un aperçu des commandes *ImageX* :

/append	Ajouter une image à un fichier *WIM* existant
/apply	Installe une image sur le disque/partition spécifié
/capture	Capture (copie exacte) d'une installation vers un nouveau fichier WIM
/check	Contrôle l'intégrité de l'image *WIM*
/commit	Applique les changements effectués sur une image montée
/compress	Spécifie le mode de compression (*none*, *fast*, *maximum*)
/config	Utilise le fichier spécifié afin de configurer des options avancées
/delete	Efface une image contenue dans un fichier *WIM* (fichier avec plusieurs images)
/dir	Affiche la liste de fichier et dossiers contenus dans une image
/export	Transfère une image contenue dans un fichier *WIM* vers un autre fichier *WIM*
/info	Affiche quelques informations de notre fichier WIM (informations contenues dans un fichier XML)
/ref	Défini les références WIM pour une opération /apply
/split	Découpe un fichier WIM en plusieurs parties de plus petite taille. Ces différentes parties pourront uniquement être lues.
/verify	Contrôle qu'il n'y ait pas d'erreurs sur les fichiers et que ces fichiers ne soient pas dupliqués
/mount	Monte une image en lecture uniquement
/mountrw	Monte une image en lecture - écriture
/unmount	Démonte une image qui a été préalablement montée
/?	Affiche les commandes d'*ImageX*

Remarque : Les commandes n'ont pas toutes été listées.

1.8.3 Créer une image avec *ImageX*

But : Montrer comment créer (capturer) une image avec *ImageX*, tester les différents modes de compression disponibles ainsi que le temps nécessaire à la création d'une image pour chacun de ces modes.

Contexte

Windows Vista RC1 Build 5728 a été installée sur une machine DELL équipée d'un Pentium 4 2.8GHz FSB[800] 1MB cache, 1024RAM DDR 400MHz, disque dur IDE 7200 tours/minute.

BDD 2007 ainsi que *Windows AIK* on été installés, afin de ne pas avoir une simple installation standard *Windows*.

Pour effectuer les différentes captures d'images, le pc a été démarré sous *Windows PE* à l'aide d'un CD *bootable* (CD avec lequel il est possible de démarrer directement un pc lors de sa mise sous tension).
Voir paragraphe 1.5.4.5.1 page 24 pour créer un CD *bootable Windows PE* personnalisé.

Tests

L'occupation totale de l'installation sur disque dur est d'environ 10.6 GB

Les lignes de commande utilisées pour capturer les images sont les suivantes :

`cd X:\Program Files`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`Imagex /compress none /check /capture c: d:\none.wim "Vista compression none" /verify`	Capture la partition **`c:`** (*Windows Vista*) Le paramètre **`/compress`** none signifie que l'on utilise aucune compression. Le paramètre **`/check`** vérifie l'intégrité de l'image. Le paramètre **`/verify`** vérifie l'image *WIM* créée pour éviter les erreurs.
`Imagex /compress fast /check /capture c: d:\fast.wim "Vista compression fast" /verify`	Idem mais avec une compression rapide.
`Imagex /compress maximum /check /capture c: d:\max.wim "Vista compression max" /verify`	Idem mais avec une compression maximum.

Les résultats obtenus sont les suivants :

Compression	Taille de base (partition)	Taille de l'image *WIM*	Temps nécessaire pour créer l'image *WIM*
`none`	10.6 GB	5.55 GB	18 minutes
`fast`	10.6 GB	3.34 GB	21 minutes
`maximum`	10.6 GB	3.18 GB	50 minutes

La compression **`fast`** paraît être le meilleur compromis entre taille de l'image *WIM* créée et temps nécessaire à sa création.

1.8.4 Créer un CD *bootable Windows PE* personnalisé avec *ImageX*

Il est utile d'avoir un CD *bootable* de *Windows PE* afin de pouvoir démarrer n'importe quel ordinateur à l'aide de ce CD, puis éventuellement utiliser *ImageX* afin de capturer une installation ou installer une image.

Windows PE est disponible dans *Windows AIK*. Il existe aussi une image *WIM* de *Windows PE* sur le DVD d'installation de *Windows Vista* (nommée boot.wim), cependant cette image à déjà été personnalisée par *Microsoft*.

Pour créer un CD *bootable Windows PE* personnalisé avec *ImageX*, consulter le paragraphe 1.5.4.5.1 page 24.

1.9 Sysprep

Avant de rentrer dans le vif du sujet, *Sysprep* m'a posé quelques problèmes lors de son utilisation sur *Windows Vista RC1* et *RC2*. Parfois des erreurs sont survenues et par la suite : impossible d'utiliser correctement *Sysprep*, la seule solution a été de réinstaller *Windows Vista* !

1.9.1 Qu'est ce que *Sysprep* ?

Sysprep signifie *System Preparation*, c'est un outil qui comme son nom l'indique, est utilisé pour préparer un système dans le but d'en faire une image que l'on puisse déployer sur d'autres ordinateurs.

Certaines entreprises qui vendent des ordinateurs utilisent aussi *sysprep* pour préparer le système afin de lancer *Windows Welcome* (voir ci-dessous l'explication de *Windows Welcome*) au prochain démarrage.

Cet outil est disponible par défaut dans une installation de *Windows Vista*, **`sysprep.exe`** se trouve dans le répertoire **`C:\Windows\System32\sysprep`**

Sysprep nous permet de :

- Retirer toutes les données spécifiques système d'une installation *Windows*, telles que les SIDs, adresses IP, etc. Ceci nous permet de pouvoir ensuite capturer une image de notre *Windows* et de l'installer à travers tout un réseau, y compris sur des ordinateurs ayant une toute autre configuration matérielle (uniquement avec *Windows Vista*)

- Configurer *Windows* pour *booter* dans le mode *Audit*. Ce mode nous permet d'installer des applications tierces ainsi que des drivers, ou encore d'examiner la fonctionnalité de l'ordinateur.

- Configurer *Windows* pour *booter* sur *Windows Welcome*.
  Windows Welcome est un programme qui est lancé lors de la première utilisation de *Windows*. C'est lui qui nous demande de créer notre premier compte utilisateur et de définir quelques paramètres supplémentaires tels que la langue du clavier, la langue d'installation… *Windows Welcome* effectue aussi un test des performances de l'ordinateur.

- Faire un *Reset* (retour a zéro) de l'activation demandée par *Windows*. Cependant ceci ne peut se faire qu'un maximum de 3 fois.

1.9.2 Commandes *Sysprep*

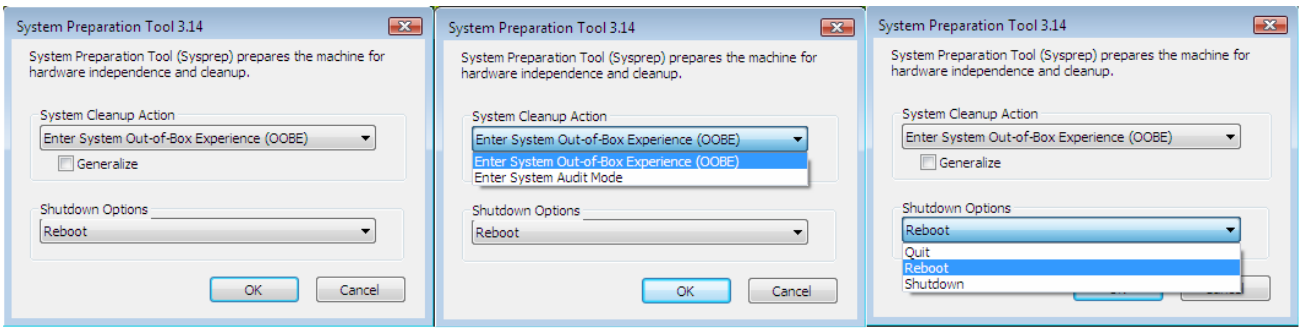
Le tableau suivant montre les différentes commandes de *Sysprep* :

/audit	Redémarre le pc en mode *Audit*. Si une *answer file* (fichier de réponse *Unattend.xml*) est utilisée, le mode audit de *Windows Setup* exécutera les phases de configuration *AuditSystem* et *AuditUser*. Voir paragraphe 1.9.7 page 78
/generalize	Enlève toutes les données spécifiques système de l'installation *Windows*. Ceci inclut les *event logs*, les SIDs ainsi que les autres informations uniques (adresses IP etc.)
/oobe	Exécutera *Windows Welcome* lors du prochain démarrage. Si une *answer file* est utilisée, les configurations *oobeSystem* seront exécutées avant le lancement de *Windows Welcome*.
/reboot	Redémarre le pc, cette option est utilisée après l'option **/generalize** ou **/audit**
/shutdown	Eteint le pc lorsque *Sysprep* à fini d'effectuer son traitement.
/quiet	Exécute *Sysprep* sans afficher de messages à l'écran.
/unattend:answerfile	Applique les configurations contenues dans le fichier de réponse. Il faut spécifier l'endroit ou se trouve le fichier de réponse à la place de "answerfile"

1.9.3 Exécution de *Sysprep*

Sysprep ne peut être exécute qu'avec les droits Administrateur.

Si *sysprep* est exécuté sans aucun paramètre, une fenêtre GUI s'ouvre et nous demande de sélectionner les paramètres *sysprep*. Cependant, les paramètres ne sont pas tous accessible via cette fenêtre GUI, tels que l'option **/quiet** ou l'option **/unattend**.



Le lancement de *sysprep* en ligne de commandes est donc préférable.

1.9.4 Créer une image *Build-to-Plan* (*BTP*)

1.9.4.1 Qu'est ce qu'une image *BTP*?

Une image *BTP* consiste à créer une image de référence pour pouvoir être installée sur d'autres ordinateurs.
Il faut donc installer *Windows* et lui ajouter/retirer ce que l'on souhaite, puis faire une capture de notre *Windows* personnalisé.

1.9.4.2 Comment créer une image *BTP*?

Voici les différentes étapes à effectuer :

Installer *Windows Vista* sur un pc et la personnaliser pour le déploiement.

Lorsque la personnalisation est terminée, effectuer les commandes suivantes :

cd C:\Windows\System32\sysprep	Dans le répertoire contenant l'exécutable **sysprep.exe**
sysprep /oobe /generalize	L'option **/oobe** nous permet de lancer *Windows Welcome* au prochain démarrage. L'option **/generalize** enlève toutes les données spécifiques système de l'installation *Windows*. Ceci inclut les *event logs*, les SIDs ainsi que les autres informations uniques (adresses IP etc.)

Démarrer à l'aide de *Windows PE* ou d'un autre système d'exploitation afin d'effectuer une capture de notre système vers une image au format WIM, ceci à l'aide d'*ImageX*.

Les lignes de commande pour créer une image de notre système :

cd X:	Dans le répertoire contenant l'exécutable **imagex.exe**
imagex /compress fast /check /capture c: d:\vistasyspreped.wim "Vista syspreped" /verify	Capture la partition **c:** (*Windows Vista*) Le paramètre **/compress fast** signifie que l'on utilise une compression rapide. Le paramètre **/check** vérifie l'intégrité de l'image. Le paramètre **/verify** vérifie l'image *WIM* créée pour éviter les erreurs.

Remarque : Les comptes utilisateurs ne sont pas effacés avec *sysprep*. Pour personnaliser les comptes utilisateurs il faut utiliser le fichier *Unattend.xml*

1.9.5 Créer une image *Build-to-Order* (*BTO*)

1.9.5.1 Qu'est ce qu'une image *BTO*?

Une image *BTO* est similaire à une image *BTP* mis à part qu'avec une image *BTO* il est possible de faire des changements additionnels qui seront uniques pour le pc en question. Une image BTO fera ces modifications dans le mode *Audit*.

Il faut commencer par installer une image de référence, puis dans le mode *Audit* on effectue des mises à jour additionnelles à notre installation *Windows*, qui sont uniques pour le pc que l'on installe.

En général, ce sont des applications ou des mises à jour demandées par la personne qui utilisera le pc.

1.9.5.2 Comment créer une image *BTO*?

Installer l'image de référence *Windows* sur le pc qui a besoin d'installations supplémentaires.

Lorsque l'installation est terminée, effectuer les commandes suivantes :

`cd C:\Windows\System32\sysprep`	Dans le répertoire contenant l'exécutable **`sysprep.exe`**
`sysprep /audit /generalize /reboot`	Avec l'option **`/audit`**, le pc entrera en mode *Audit* la prochaine fois qu'il sera démarré. L'option **`/généralise`** enlève toutes les données spécifiques système de l'installation *Windows*. Ceci inclut les *event logs*, les SIDs ainsi que les autres informations uniques (adresses IP etc.) L'option **`/reboot`** fera un *reboot* du pc à la fin du traitement.

Lorsque le pc démarre en mode *Audit*, installer les applications supplémentaires, puis exécuter les commandes :

`cd C:\Windows\System32\sysprep`	Dans le répertoire contenant l'exécutable **`sysprep.exe`**
`sysprep /oobe /shutdown`	L'option **`/oobe`** nous permet de lancer *Windows Welcome* au prochain démarrage. L'option **`/shutdown`** éteint le pc à la fin du traitement.

Si besoin, démarrer avec *Windows PE* ou un autre système d'exploitation et en faire une image à l'aide d'*ImageX* :

`cd X:\`	Dans le répertoire contenant l'exécutable **`imagex.exe`**
`imagex /compress fast /check /capture c: d:\vistasyspreped.wim "Vista syspreped" /verify`	Capture la partition **`c:`** (*Windows Vista*) Le paramètre **`/compress fast`** signifie que l'on utilise une compression rapide. Le paramètre **`/check`** vérifie l'intégrité de l'image. Le paramètre **`/verify`** vérifie l'image *WIM* créée pour éviter les erreurs.

1.9.6 Comparaison entre *Build-to-Plan* et *Build-to-Order*

Les images *BTP* sont plus souvent utilisées que les images *BTO*.
Ceci s'explique par le fait que, pour créer une image *BTO*, il faut d'abord installer une image *BTP* puis la modifier pour en faire une image *BTO*.
Les images *BTO* sont simplement des ajouts aux images *BTP*, et donc par conséquent, un supplément (option) aux images *BTP*.

Tout au long de ce travail de diplôme, seules les images *BTP* sont utilisées.

1.9.7 Le mode *Audit*

Avant d'aller plus loin, je tiens à préciser que ce mode n'est pas vraiment utile dans le cadre de ce diplôme.
La seule raison qui puisse pousser un ingénieur système à utiliser ce mode serait l'installation rapide d'une application à la demande d'un utilisateur spécifique, sans avoir à passer par *Windows Welcome* (nécessitant de 5 à 15minutes dépendant de la configuration *hardware*).

1.9.7.1 Qu'est ce que c'est ?

Le mode *Audit* est un mode particulier dans *Windows Vista*, il est notamment utilisé pour installer d'autres drivers et applications spécifiques à 1 seul ordinateur.

1.9.7.2 Que peut-on faire avec ce mode ?

Le mode *Audit* permet d'éviter le lancement de *Windows Welcome*, et ainsi arriver plus rapidement sur le système d'exploitation afin de personnaliser une image.

Comme indiqué ci-dessus, ce mode permet d'installer des *drivers* (pilotes), d'exécuter des *scripts*, et de tester la validité de l'installation de *Windows* avant de confier l'ordinateur à son utilisateur.

Le mode *Audit* n'a pas besoin des paramètres demandés par *Windows Welcome* pour être utilisé.

Il est aussi possible de vérifier que le démarrage de *Windows Welcome* se déroule comme prévu, et que les informations telles que le support de notre entreprise soient correctement affichées.

1.9.7.3 Comment démarrer en mode *Audit* ?

Les différentes manières de *booter* sur le mode *Audit* sont les suivantes :

- Lors d'une installation de base de *Windows*, à l'écran de *Windows Welcome*, appuyer sur **Ctrl+Shift+F3**
- Lors d'une installation dite *unattended* (personnalisée), il faut ajouter le composant *Microsoft-Windows-Deployment* dans la phase de configuration *oobeSystem* (voir en annexes).

  Dans les paramètres *Reseal* | *Mode*, spécifier *Audit*.

  Lorsque *Windows* aura terminé l'installation, le pc redémarrera en mode *Audit*.
- Exécuter les commandes :

`cd C:\Windows\System32\sysprep`	Dans le répertoire contenant l'exécutable **`sysprep.exe`**
`sysprep /audit`	Lors du prochain démarrage, le pc *bootera* en mode *Audit*

Remarque : En mode *Audit*, le pc sera logué avec le compte Administrateur.

1.9.8 Limitations de *Sysprep*

Les limitations de *Sysprep* sont les suivantes :

- Seule la version de *Sysprep* fournie avec le système d'exploitation peut être utilisée.
- *Sysprep* doit toujours être exécuté à partir de son emplacement de base, qui est **`C:\Windows\System32\sysprep`**
- *Sysprep* ne doit pas être utilisé sur des installations mises à niveau (par exemple *Windows XP* qui a été migré vers *Windows Vista*). *Sysprep* peut seulement être utilisé sur des installations propres.
- Si *Sysprep* est utilisé sur un système NTFS qui contient des fichiers ou dossiers cryptés, les données contenues dans ces fichiers seront perdues.
- *Sysprep* converti le nom de PC (*computer name*) en majuscules, ce nom ne sera donc pas effacé.
- En exécutant *Sysprep*, *Windows Welcome* nous demandera une clé *Windows Vista*. Il est possible d'utiliser un fichier XML qui est le fichier *Unattend.xml* configuré avec la clé de *Windows*, de manière à ne pas la rentrer une nouvelle fois. Cette information peut être ajoutée dans *Microsoft-Shell-Setup*, à l'aide de *Windows Image Manager*.
- *Sysprep* ne doit pas être utilisé lorsque le partage de fichiers de *Windows Media Player* est activé.
- Si l'on utilise la commande **`imagex /apply`**, pour installer une image *Windows* sur un pc, nous avons quelques restrictions à respecter.
  Par exemple, si on capture une image *Windows* sur une partition D, l'image devra toujours être déployée vers une partition D.

  Il faudra respecter les limitations suivantes :

  - Le numéro de partition ou *Windows Vista* est installé doit correspondre.
  - Le type de partition (*Primary*, *Extended*, *Logical*) doit correspondre.
  - Si la partition est configurée comme Active sur le pc de référence (sur lequel l'image a été produite), la partition de l'ordinateur qui va recevoir l'image devra aussi être configurée comme Active.
  - S'il y a une autre partition Active pour *Bootmgr* (*Boot Manager*) et BCD (*Boot Configuration Data*) sur le système de référence, il faut aussi capturer cette partition et la déployer sur le pc auquel on veut installer notre image.

  Ces limitations s'appliquent uniquement à la commande **`imagex /apply`**.

  Certaines applications qui sont installées avant que l'image de *Windows* ne soit déployée, peuvent être configurées pour demander une certaine partition (C, D, etc.) Il se peut que certaines applications ne fonctionnent plus correctement si la lettre de partition système ne correspond pas à la lettre que ces applications demandent.

Il est donc recommandé d'installer l'image de *Windows* ainsi que les diverses application sur la même partition.

- *Sysprep* ne doit pas être utilisé sur des ordinateurs faisant partie d'un domaine. L'option **`/generalize`** retire bien l'ordinateur du domaine, cependant les mots de passe pour les différents comptes utilisateurs ne sont pas retirés.
  Si l'image est déployée, les mots de passe pour les différents comptes utilisateurs doivent être spécifiés dans le fichier *Unattend.xml*, le cas contraire l'installation ne se fera pas correctement.

1.9.9 Dépendances de *Sysprep*

- *Sysprep* ne peut être utilisé qu'après avoir démarré Windows.
- Un programme tel qu'*ImageX* est nécessaire afin de capturer une image ou installer une image.

1.10 Fichier *Unattend.xml*

1.10.1 Qu'est ce que c'est ?

Un fichier *Unattend.xml* est un fichier de réponse (ou "*answer file*" en anglais) permettant de personnaliser et automatiser une installation *Windows*.

Ce fichier est au format *XML* (*Extensible Markup Language*).

1.10.2 Comment créer un tel fichier ?

Ce fichier peut être créé à l'aide de *Windows System Image Manager* (voir en annexes).

Etant donné qu'il s'agit d'un fichier XML, il peut évidemment aussi être créé ou modifié à la main via un simple éditeur de texte tel que *notepad*.
Cependant, cette méthode de création demande une bonne connaissance de sa structure, ce qui n'est pas toujours le cas pour les différents utilisateurs d'un tel fichier.

1.10.3 Sa structure

Un fichier *Unattend.xml* est structuré en diverses phases de configuration (ou *passes* en anglais).

Ces différentes phases sont expliquées dans le chapitre suivant

Chaque phase de configuration est délimitée par des balises XML, exemple :

```
<settings pass="specialize">
…
…
</settings>
```

Toutes les configurations contenues entre ces 2 balises sont effectuées dans leur phase respective (dans cet exemple cette phase est : **specialize**).

A l'intérieur de ces balises, nous trouverons d'autres balises qui font référence à des configurations bien précises.

1.10.4 Les différentes phases de configuration lors d'une installation

Lors de l'installation d'une image, **7 phases** de configuration sont possibles :

- ***Windows PE***, qui permet de configurer les diverses options de *Windows PE*, ainsi que les options de *Windows Setup*.

  Ces options peuvent être inclure la clef de *Vista*, configurer un disque (formater, partitionner, etc.)

- ***OfflineServicing***, qui permet d'effectuer des manipulations sur notre image, tels qu'effectuer des *Updates* (mises à jour), ajouter des *packages* ou *language packs* (packs de langue (français, anglais) pour *Windows Vista*), effectuer des mises à jour de la sécurité, etc.

- ***Specialize***, qui permet d'appliquer des informations système spécifiques, tels que configurer les options réseau, les options de domaine, *Internet Explorer*, etc.

- ***Generalize***, qui permet de configurer **`sysprep\generalize`** afin de garder certaines options *Windows* que l'on souhaite garder dans notre image.

  Cette phase s'exécute uniquement si la commande **`sysprep\generalize`** est effectuée.

- ***AuditSystem***, qui permet d'appliquer des configurations au système, tels que l'ajout de *drivers* (pilotes), applications ou mises à jour de façon automatisée à l'aide d'un fichier *unattend* ("fichier de réponse" ou "*answer file*" en anglais).

  Cette phase s'exécute uniquement si l'on démarre en mode *Audit*, elle s'effectue avant la phase *AuditUser* et avant qu'un utilisateur soit *logué* (avant qu'il ait entré son nom d'utilisateur et son mot de passe) en mode *Audit*.

- ***AuditUser***, qui permet d'exécuter des *scripts* (ensemble de lignes de commande), des applications ou d'autres exécutables de façon automatisée à l'aide d'un fichier *unattend*.

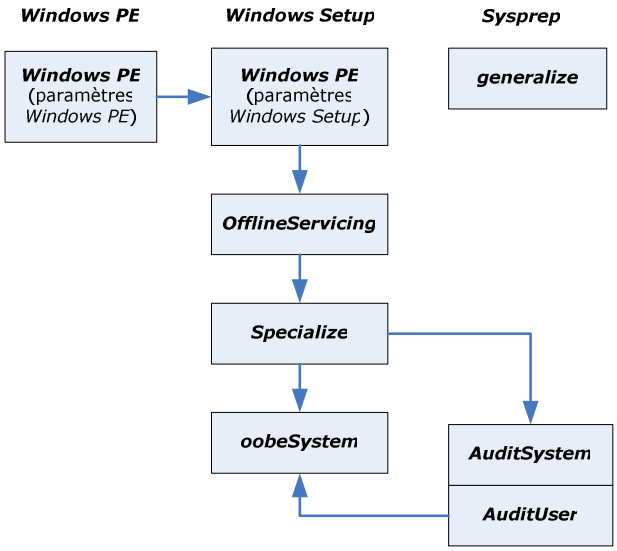
  Cette phase s'exécute après la phase *AuditSystem*.

- ***oobeSystem*** (*oobe* signifie *Out-Of-Box-Experience*), cette phase permet de configurer et appliquer des paramètres durant le premier *boot* de la machine (appelé *first-boot expérience*, ou encore *Windows Welcome*).

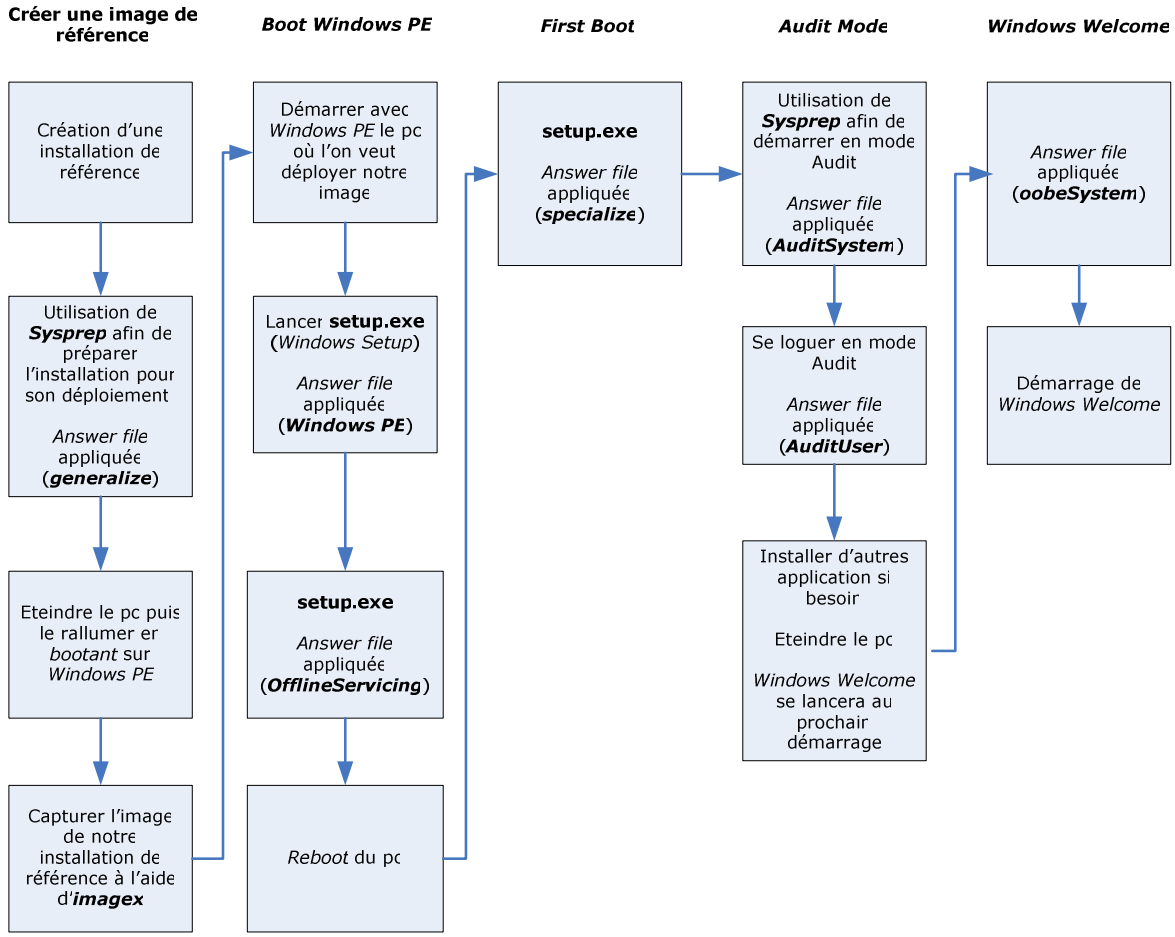
  Ces paramètres peuvent être la désactivation de la *Sidebar* (barre latérale par défaut dans *Vista*), création de comptes utilisateurs, affecter un mot de passe au compte Administrateur, etc.

  Les différents paramètres contenus dans le fichier *unattend.xml* sont appliqués avant que l'utilisateur ne puisse se *loguer* pour la première fois sur *Windows*.

Le diagramme suivant montre les relations entre les différentes phases ainsi que les phases de configuration possibles pour les différents programmes (*Windows PE*, *Windows Setup* et *Sysprep*) :



Le diagramme suivant montre les différentes phases lors d'un scénario *Build-To-Order* :



Lors d'un scénario *Build-to-Plan*, ce diagramme est aussi valable à l'exception de toute la partie ***Audit Mode*** qui n'existe pas.

1.10.5 *Answer Files* dans *Windows XP*

Dans *Windows XP*, il faut configurer divers fichiers pour automatiser et configurer une installation *Windows*.

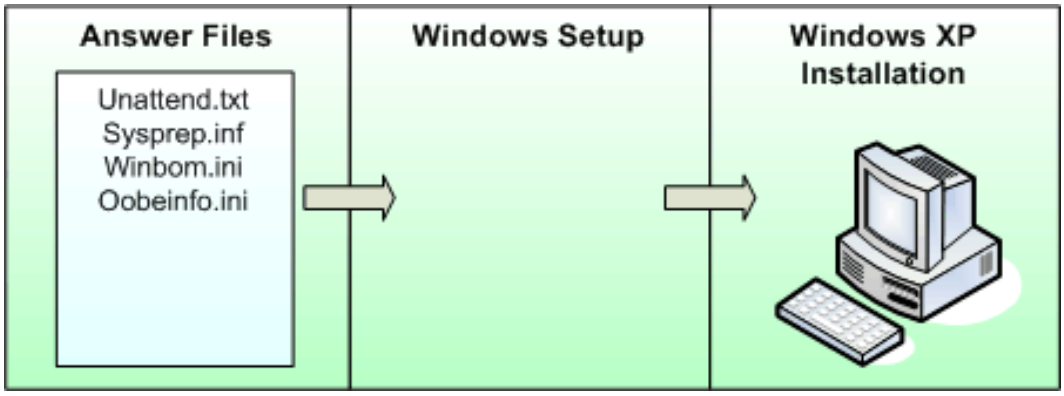


Image prise dans la présentation de M.CAU Lionel du 5 et 6 Avril 2006 *Microsoft TechDays*

Il faut **différents outils** pour configurer ces divers fichiers, de plus il faut pouvoir les **synchroniser** entre eux, ce qui rend cette tâche encore plus difficile.

Il y a aussi des **paramètres en doublons** dans ces divers fichiers, ce qui fait que si dans un fichier on spécifie une certaine valeur, et que dans l'autre fichier se trouve une valeur différente, cela provoquera une erreur !
Il est possible de perdre beaucoup de temps afin d'identifier et corriger cette erreur !

Ces différents fichiers disposés un peu partout sur l'ordinateur ainsi que leur configuration est **archaïque**.

1.10.6 *Answer Files* dans *Windows Vista*

Dans *Windows Vista* un seul fichier XML est nécessaire pour toute la configuration.

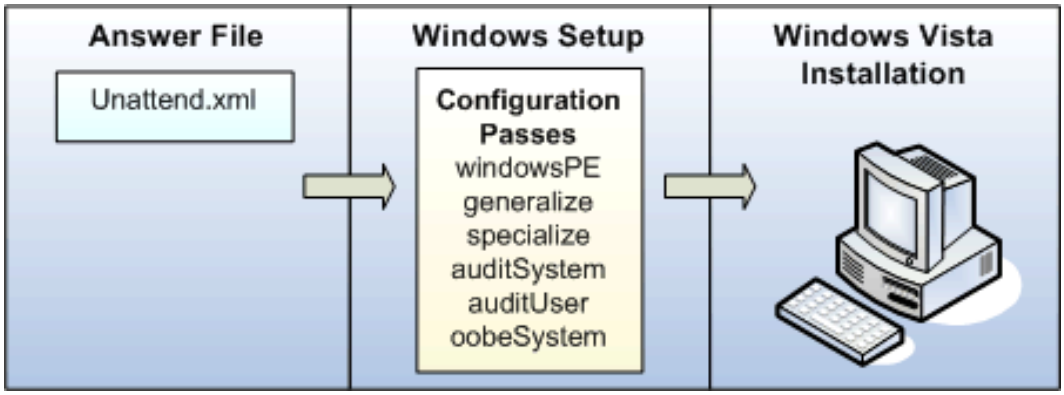


Image prise dans la présentation de M.CAU Lionel du 5 et 6 Avril 2006 *Microsoft TechDays*

Un seul outil est nécessaire pour créer le fichier *Unattend.xml* : *Windows System Image Manager* (voir en annexes).

Lors de chaque phase de *Windows Setup*, le fichier XML sera parcouru afin de détecter s'il y a des configurations à effectuer pour la phase en question.
Si tel est le cas, les configurations pour cette phase seront effectuées.

1.10.7 Comparaison entre les anciens fichiers de réponse et les nouvelles phases de configuration

Voici un tableau montrant la relation entre les divers fichiers de réponse nécessaires pour les versions antérieures à *Windows Vista* et les nouvelles phases de configuration dans *Vista*.

Anciens fichiers de réponse	**Phases de configuration *Vista***
Unattend.txt	*generalize, specialize*
Sysprep.inf	*generalize, specialize*
Winbom.ini WINPE	*windows PE*
Winbom.ini FACTORY	*auditSystem, auditUser*
Winbom.ini OOBE	*oobeSystem*
Oobeinfo.ini	*oobeSystem*

Un seul fichier *Unattend.xml* pour les différentes phases de configuration dans *Vista*.

1.10.8 Du point de vue sécurité

Dans *Windows Vista*, un seul fichier XML (*Unattend.xml*) est nécessaire pour la personnalisation et automatisation d'une installation *Vista*, cette unicité du fichier de réponse garanti une meilleure sécurité.

Les composants de *Vista* sont modulaires, c'est à dire qu'ils sont installés et exécutés indépendamment. Ceci nous permet de retirer ou désactiver ceux dont on n'a pas besoin, tel que retirer *Windows Media Player*, désactiver la demande d'assistance à distance, etc.

Moins il y a de programmes (lignes de code), moins il y a de failles.

Pour plus de précision concernant les différents composants, consulter les annexes.

Comme nous le verrons plus tard, il est possible avec un fichier de réponse, de configurer automatiquement lors d'une installation des comptes utilisateurs les mots de passe qui leur sont associés.
Si le fichier *Unattend.xml* est créé avec *Windows System Image Manager*, ces mots de passe n'apparaîtront pas en clair dans le fichier, ils seront chiffrés !

J'ai effectué quelques recherches afin de voir quelle est la méthode de chiffrement utilisée, cependant je n'ai trouvé aucune information, que ce soit sur le site de *Microsoft*, sur l'aide disponible dans *Windows AIK* ou encore dans les *newsgroups*…

1.11 Serveur *WDS* (*Windows Deployment Services*)

1.11.1 Introduction

Windows Deployment Services (*WDS*) est un serveur utilisé pour le déploiement.

WDS est la nouvelle version de *RIS* (*Remote Installation Services*).

Ce programme est gratuit et fourni par *Microsoft*. Il est utilisé pour déployer plus rapidement des images *Windows* à distance, ceci à travers le réseau.

WDS est disponible dans *Windows AIK* (lui même disponible dans *BDD 2007*), il peut être installé sur un *Windows Server 2003*.

1.11.2 Fonctionnement

Windows Deployment Services est en réalité plusieurs *components* qui fonctionnent ensemble pour permettre le déploiement d'images à travers le réseau.

Ces *components* sont organisés en 3 catégories :

- ***Server components***, tels qu'un serveur *PXE* (*Pre-Boot Execution Environment*), un serveur *TFTP* (*Trivial File Transfert Protocol*).
  Ces components sont utilisés pour permettre à un pc distant de *booter* sur le réseau (avec *PXE*) puis de télécharger la ou les images à installer via *TFTP*.

- ***Client components***, tels qu'une interface graphique (*GUI*) qui s'affiche lors de l'exécution de *Windows PE* et communique avec les *server components* afin de sélectionner et installer une image.

- ***Management components***, qui sont plusieurs *tools* (outils) utilisés pour gérer les serveurs, images système, et comptes utilisateurs.

1.11.3 Les différents modes

Un serveur *WDS* peut être utilisé en plusieurs modes :

- ***Legacy Mode***, ce mode est très semblable à *RIS*, l'environnement de *boot* doit être *OSChooser*, les images compatibles doivent être au format *RISETUP* ou *RIPREP*.
  Ce mode ne nous intéresse donc pas.

- ***Mixed Mode***, c'est un mode mixte qui accepte comme environnement de *boot* *OSChooser* ou *Windows PE*.

  Dans le cas ou l'environnement de *boot* est *OSChooser*, les images doivent être au format *RISETUP* ou *RIPREP*.

  Dans le cas ou l'environnement de *boot* est *Windows PE*, les images doivent être au format *WIM*.

  Ce mode est intéressant lorsque nous avons divers systèmes d'exploitation à déployer dans notre réseau.
  Remarque : Un serveur en *Mixed mode* ne peut être installé que sur *Windows*

Server 2003.

- ***Native Mode***, l'environnement de *boot* doit être *Windows PE* et les images doivent être au format *WIM*.

  Remarque : Ce mode est le seul mode accepté lors de l'installation de *WDS* sur *Windows Server Longhorn*.

Le mode par défaut *Mixed Mode* a été utilisé au laboratoire.

1.11.4 Pré-requis à l'installation

Un serveur *WDS* ne peut être installé que si notre réseau comprend :

- ***Windows Server 2003 SP1*** avec ***RIS*** installé. Ce serveur *RIS* n'a pas besoin d'être configuré, il est uniquement nécessaire pour que l'*update* (mise à jour) vers un serveur *WDS* puisse s'effectuer.

- ***DHCP***, il faut un serveur *DHCP* car *WDS* utilise *PXE* (qui lui même utilise *DHCP*).

- ***DNS***, un serveur *DNS* doit être présent pour exécuter *WDS*.

- ***Active Directory***, un serveur *WDS* doit être membre d'un domaine *Active Directory*.

- Une **partition *NTFS* sur** le **serveur *WDS*** car *WDS* à besoin d'une partition *NTFS* pour stocker la ou les images.

Problèmes possibles :

Il est possible d'obtenir des erreurs lors de l'envoi sur le réseau d'images *WIM* découpées en plusieurs parties.

Il est donc préférable de regrouper les différentes parties en un seul fichier *WIM* avant de faire ceci.

1.11.5 *Booter* un pc client avec *PXE*

Pour qu'un pc client (sur lequel on veut installer une image) puisse se connecter à notre serveur *WDS*, ce pc doit supporter le mode de *boot PXE*, ce qui veut dire qu'il doit être capable de *booter* à partir de sa carte réseau.

Si le pc supporte ce mode, ce mode doit être activé dans le BIOS et sélectionné comme mode de *boot* principal (ou disponible comme choix).

Dans notre laboratoire, le pc utilisé afin de *booter* en PXE puis se connecter au serveur WDS est un DELL OPTIPLEX GX270. Sa configuration est disponible en annexe.

Lors du *boot* sur *PXE*, notre pc client attend que notre serveur DHCP lui attribue une adresse IP.

Lorsque notre pc client à reçu une adresse IP valide, un message à l'écran apparaît nous demandant de confirmer le *boot PXE* en appuyant F12 une nouvelle fois (sur le poste Dell utilisé au labo).

L'image de *boot* que l'on a spécifié sur notre serveur se lance.
Si plusieurs images de *boot* sont disponibles sur notre serveur, nous aurons un menu de *boot* (voir paragraphe suivant) nous permettant de choisir l'image de *Windows PE* à télécharger pour son lancement.

Problèmes possibles en utilisant *PXE* avec *WDS* :

Les divers systèmes d'exploitation disponibles sur le serveur *WDS* peuvent ne pas apparaître dans la fenêtre "*Image Selection*" du client *WDS*.

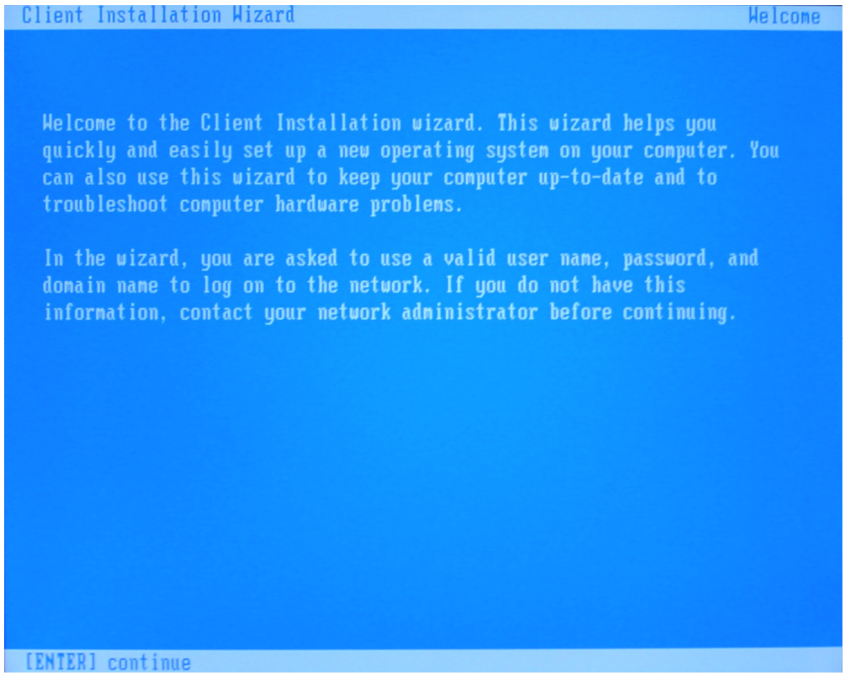
Les causes possibles à ce problème sont :

- Le compte utilisé n'a pas les permissions suffisantes pour lire les images *WIM* disponibles sur le serveur *WDS*.

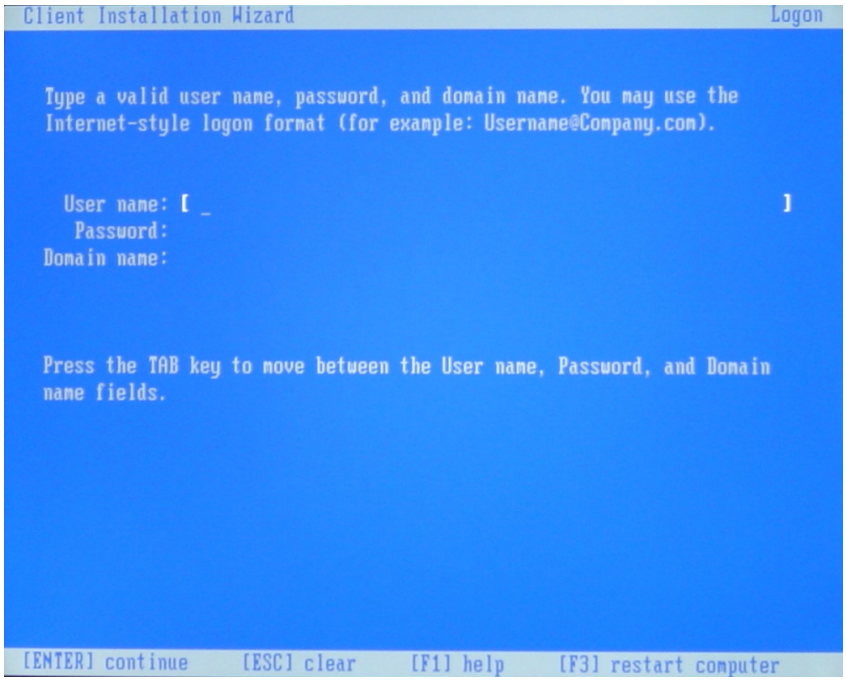
- L'architecture du client (x86, Itanium, x64) ne correspond pas avec le type d'architecture des images disponibles sur le serveur *WDS*.
  Par exemple un client qui a *booté* sur un *Windows PE* basé x86 avec un client *WDS* installé pourra uniquement consulter les images d'installation de type x86.

Les pc clients basés x64 peuvent ne pas recevoir d'image *Windows PE* x64, ceci peut être du à une mauvaise configuration BIOS.

Si le client ne parvient pas à trouver de serveur *WDS*, le message suivant sera affiché :



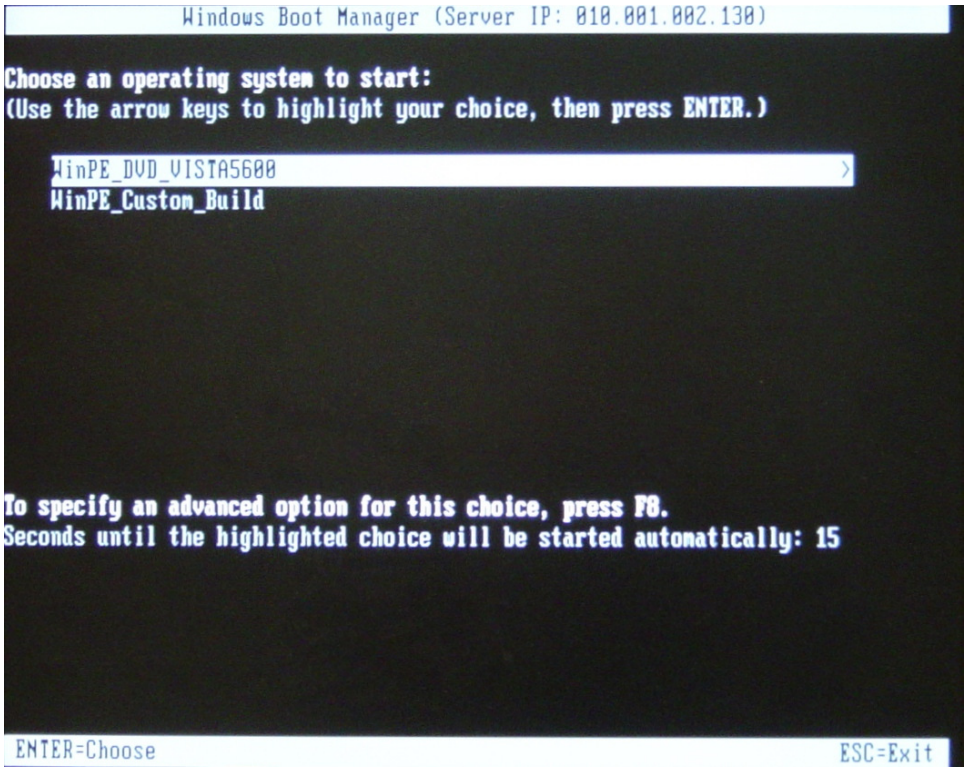
Ceci nous permettra de spécifier un nom d'utilisateur, un mot de passe et le nom de domaine sur lequel se connecter.



En principe, si tout est correctement configuré, ces 2 dernières captures d'écran n'apparaîtront jamais. Si elles apparaissent c'est qu'il y a vraisemblablement un problème de configuration.

1.11.6 Le menu de *boot*

Si plusieurs images de *boot* (images de *Windows PE*) sont installées sur le serveur *WDS*, lors de la connexion d'un client à notre serveur, un menu lui sera affiché permettant de choisir l'image de *boot* souhaitée :



Ce menu de *boot* permet à *WDS* de pouvoir séparer les différentes architectures (x86, x64…) ainsi que de gérer différentes images pour chaque architecture.

Pour configurer ce menu, il faut être membre du groupe *Local Administrators* sur le serveur *WDS*.

Limitations de configuration du menu de *boot* :

- Le menu de *boot* ne peut afficher que 13 images différentes.
- Les caractères de type *double-byte* (caractères encodés sur 2 octets) ne peuvent être affichés correctement par le menu de *boot*.

Le nom des images doit contenir uniquement des chiffres et des lettres, il ne peut pas y avoir d'espaces.

1.11.7 Utilisation des fichiers *Unattend.xml*

La création d'un fichier *Unattend.xml* est décrite en annexes.

L'utilisation des fichiers *Unattend.xml* sur un serveur *WDS* est décrite dans le paragraphe 1.5.6.2.5 et 1.5.6.2.6, page 44.

De plus pour pouvoir utiliser des fichiers *Unattend.xml*, il faut être membre du groupe *Local Administrators* sur le serveur *WDS* et avoir les droits suffisants pour ajouter un pc à un domaine.

1.11.8 Du point de vue sécurité

Pour se connecter à notre serveur *WDS*, il faut être membre du domaine et donc disposer d'un nom d'utilisateur et d'un mot de passe afin de s'authentifier.

De plus, il est possible d'associer un compte à certaines images, par exemple nous pouvons créer un compte pour une secrétaire.

Ce compte aura le droit de lecture uniquement sur des images créées pour les secrétaires.

De ce fait, une secrétaire ne pourra pas, par exemple, installer sur son poste une image créée pour un ingénieur (ayant évidemment plus de droits et d'outils qu'une simple image secrétaire) !

1.11.9 Comparaison avec les solutions alternatives

Comparons maintenant notre serveur *WDS* aux autres possibilités que l'on trouve sur le marché.

Microsoft propose quant à lui un autre serveur d'images, appelé serveur *SMS* (*System Management Server*), qui est censé améliorer le déploiement.
SMS permet entre autres d'installer une application sur des systèmes *Windows* déjà opérationnels, il utilise aussi des outils tels qu'*ImageX* de manière totalement transparente.
Ce serveur *SMS* n'a pas été testé, c'est un serveur payant, contrairement au serveur *WDS* qui est gratuit.

D'autres entreprises produisent aussi des solutions de déploiement, comme par exemple *Norton Ghost*, qui lui aussi est payant. Cependant, *Norton Ghost* possède son propre système d'images, et donc certaines propriétés du format *WIM* sont perdues, tels que la possibilité d'avoir plusieurs images dans le même fichier, l'installation non destructive, ou encore la possibilité de gérer les images sans devoir les réinstaller (mode *Offline*).

1.12 Liens utiles

Webcast (vidéo) sur le déploiement (en anglais):
http://www.microsoft.com/belux/technet/fr/itsshowtime/sessionh.aspx?videoid=208

Quelques *Virtual Labs* (laboratoires virtuels) qui peuvent être utiles :
http://www.microsoft.com/france/technet/traincert/virtuallab/default.mspx

Virtual Lab sur la création d'images et automatisation :
http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?EventID=1032305600&EventCategory=3&culture=en-US&CountryCode=US

Virtual Lab sur l'utilisation de *Windows System Image Manager* :
http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?EventID=1032305603&EventCategory=3&culture=en-US&CountryCode=US

2 *PowerShell*

2 semaines d'étude

2.1 Introduction

Dans cette deuxième partie de mon travail de diplôme, je vais étudier le nouveau *shell* (interface en lignes de commande) gratuit proposé par *Microsoft* nommé *PowerShell*.

Je vais donc voir ce qu'il est possible de faire avec ce *shell*, noter les différences par rapport à MS-DOS ainsi qu'aux *shells* d'*Unix*/*Linux*, puis essayer de trouver quelques scénarios intéressants dans le cadre des images *Vista* ainsi que de leur déploiement. Nous verrons aussi si ce *shell* peut être utilisé pour des études orientées *Forensics*.

2.2 Pourquoi avoir créé *PowerShell* ?

PowerShell a été créé pour remplacer l'actuelle interface en ligne de commandes MS-DOS disponibles dans les différentes versions *Windows*.

En créant *PowerShell*, le but de *Microsoft* était de faire un langage de *script* puissant, avec autant de fonctionnalités que ceux qui existent sous *Unix* (et *Linux*) avec le même niveau de sécurité (ceci est expliqué en page 97, paragraphe 2.8).

Lorsque des administrateurs utilisant des systèmes *Unix* ou *Linux* doivent travailler avec des systèmes *Windows*, ils n'avaient pas les mêmes possibilités avec MS-DOS qu'avec les *shells* de *Unix*/*Linux* auxquels ils étaient habitués.
Ils devaient souvent recourir à des programmes tiers afin d'essayer de faire les mêmes opérations que sur les *shells Unix*/*Linux*, et ceci leur posait de gros problèmes ! (bugs, matériels non supportés, etc.).

L'idée a donc été de faire un *shell* pouvant rivaliser avec les *shells* d'*Unix* et *Linux*, pour qu'un administrateur *Linux* puisse rapidement utiliser *PowerShell* et ceci avec les mêmes commandes auxquelles il est habitué sous *Unix* ou *Linux* (*cd*, *pwd*, *ls*, etc.) ainsi qu'en matière de *scripts*.

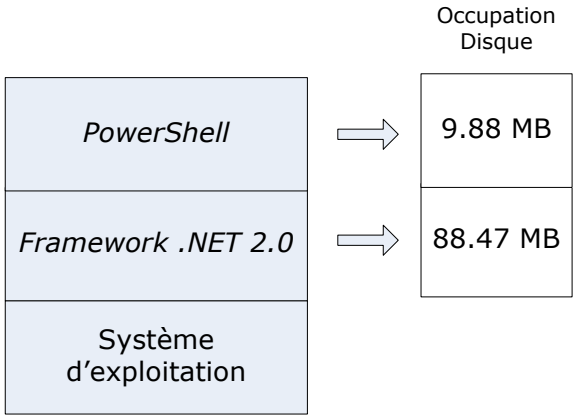
PowerShell s'inspire du monde *Unix* qui reste une référence en matière de *scripts*, cependant nous verrons dans le paragraphe 1.7 que *PowerShell* en diffère sur plusieurs points.

2.3 Qu'est ce que *PowerShell* ?

Windows PowerShell, anciennement *Microsoft Command Shell –MSH-* (nom de code *Monad*) est une interface en ligne de commandes (*shell*) et un langage de *script*.

PowerShell est basé sur la programmation orientée objet et le *framework Microsoft .NET 2.0*.

Si nous regardons *Powershell* au niveau couches :



A l'origine, il était prévu que *PowerShell* soit inclus dans *Windows Vista*, mais finalement les 2 logiciels sont disjoints. *Microsoft* prévoit de fournir *PowerShell* par défaut dans *Microsoft Exchange Server 2007*.

Selon *Microsoft*, *PowerShell* est compatible avec toutes les versions *Windows* qui supportent le *Framework .NET 2.0*, cependant au moment de mon étude *PowerShell*, je n'ai pas réussi à l'installer sous *Windows Vista RC1/RC2*. Les seules versions disponibles sur le site de *Microsoft* étaient destinées à *Windows XP SP2* et *Windows Server 2003 SP1*.

Le 14 Novembre 2006, *Microsoft* proposa en téléchargement la version finale de *PowerShell* 1.0 RTW, dont une version pour *Windows Vista RC1*, pas de version compatible *Vista RC2 !*

Seule la version *PowerShell RC2* pour *Windows XP SP2* a été utilisée tout au long de cette étude.

PowerShell est compatible avec les précédents langages de *script* tels que *batch* (format .BAT) cependant, ces *scripts* ne sont pas interprétés directement par *PowerShell*, ils sont redirigés vers cmd.exe pour les exécuter.

Remarque : La modification de variables d'environnement dans un *script* .cmd afin d'affecter l'exécution d'autres *scripts* n'est pas supportée dans *PowerShell*).

2.4 Qu'est ce qu'un *script* ?

Un *script* est un fichier contenant une suite logique de lignes de commandes ainsi que du code, ils sont utilisés pour automatiser certaines tâches et ainsi gagner du temps.

Un *script* à proprement parler contient des lignes de code qui lui sont propres, telles que des bouches, des tests, des conditions, etc. (consulter la page 115, paragraphe 2.14 pour quelques exemples).

On les utilise par exemple afin d'éviter de réécrire à chaque fois les mêmes commandes. On stocke donc les différentes commandes dans un fichier (*script*) puis on exécute ce *script* en temps voulu.

Remarque : Dans *PowerShell* les fichiers de *script* ont l'extension ***.ps1**

2.5 Exécuter un *script* avec PowerShell?

Un script peut être exécuté en effectuant dans *PowerShell* les commandes suivantes :

`cd X:`	Dans le répertoire contenant le *script* à lancer.
`.\nom_du_script.ps1`	Le point "." signifie que le *script* se trouve dans le répertoire courant. Remplacer **`nom_du_script`** par le nom du *script* à exécuter.

2.6 Comment installer *PowerShell* ?

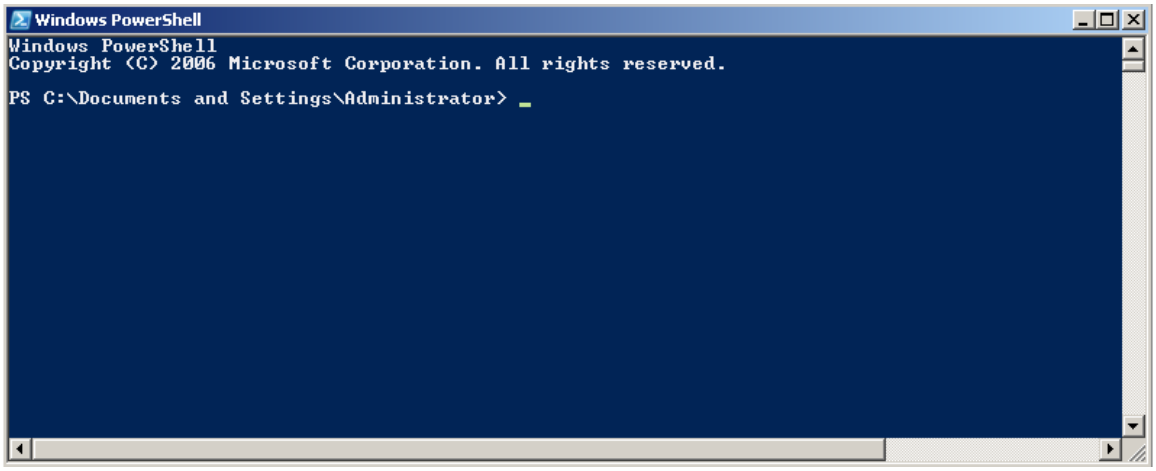
Avant de pouvoir installer *PowerShell*, il faut avoir installé au préalable *Microsoft Framework .NET 2.0* (exécutable de 22.4MB, 88.47MB d'occupation disque après installation), qui peut se télécharger à l'adresse :
http://www.microsoft.com/downloads/details.aspx?FamilyId=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=fr

La version utilisée tout au long de ce mémoire a été *PowerShell RC2* compatible *Windows XP SP2* (exécutable de 1.6MB, 9.88MB d'occupation disque après installation).
Suite à la sortie de la version *RTW*, la *RC2* n'est donc plus disponible au téléchargement.

Comme indiqué dans le chapitre 2.3 page 95, *Microsoft* proposa le 14 Novembre 2006 la version finale de *PowerShell* 1.0 *RTW* (pour *Windows Vista RC1 Build 5600*, *Windows XP SP2* ou *Windows Server 2003*), ces versions peuvent être téléchargées à l'adresse :
http://www.microsoft.com/windowsserver2003/technologies/management/powershell/download.mspx

2.7 A quoi ressemble-t-il ?

PowerShell est une fenêtre d'invite de commandes, tout comme cmd.exe



2.8 Comparaison de *PowerShell* et les *shells* sous *Unix/Linux*

2.8.1 Echanges d'informations, passage de paramètres

Dans la plupart des *shell*, les échanges entre composants se font sous forme de texte.

Sous *Unix* les échanges se font sous forme d'un *pipeline* (tuyau) au format texte entre la sortie d'un composant et l'entrée d'un autre composant, ce qui implique que parfois il faut réordonner les différentes informations pour les rendre compatibles au composant qui les reçoit, ce qui peut facilement alourdir l'écriture des lignes de commandes.

Sous *PowerShell*, les échanges de font sous forme d'objet .NET, ce qui ne demande pas de ré-ordonnancement des différentes informations.

2.8.2 Le typage (informations de type)

Les différents paramètres ont chacun un type (entier, chaîne de caractères, etc.).

PowerShell a été créé pour gérer efficacement les erreurs de typage. Le typage permet d'éviter de passer des types incompatibles d'uns stade à l'autre du *pipeline*.

Par exemple, si la première commande renvoie un texte et que la deuxième nécessite en entrée un objet particulier, une erreur sera renvoyée.

2.8.3 Les *cmd-let* (ou *cmdlet*)

Dans *PowerShell*, les commandes sont appelées *cmd-let* (prononcées *commandlet*).

Ce sont en réalité des instances d'une certaine classe .NET, de ce fait ces *cmd-let* sont créées avec relativement peu de lignes de code qui manipulent et renvoient des instances de classe .NET.

Ces cmd-let sont constituées d'un verbe puis d'un nom, le tout séparé par un trait d'union.

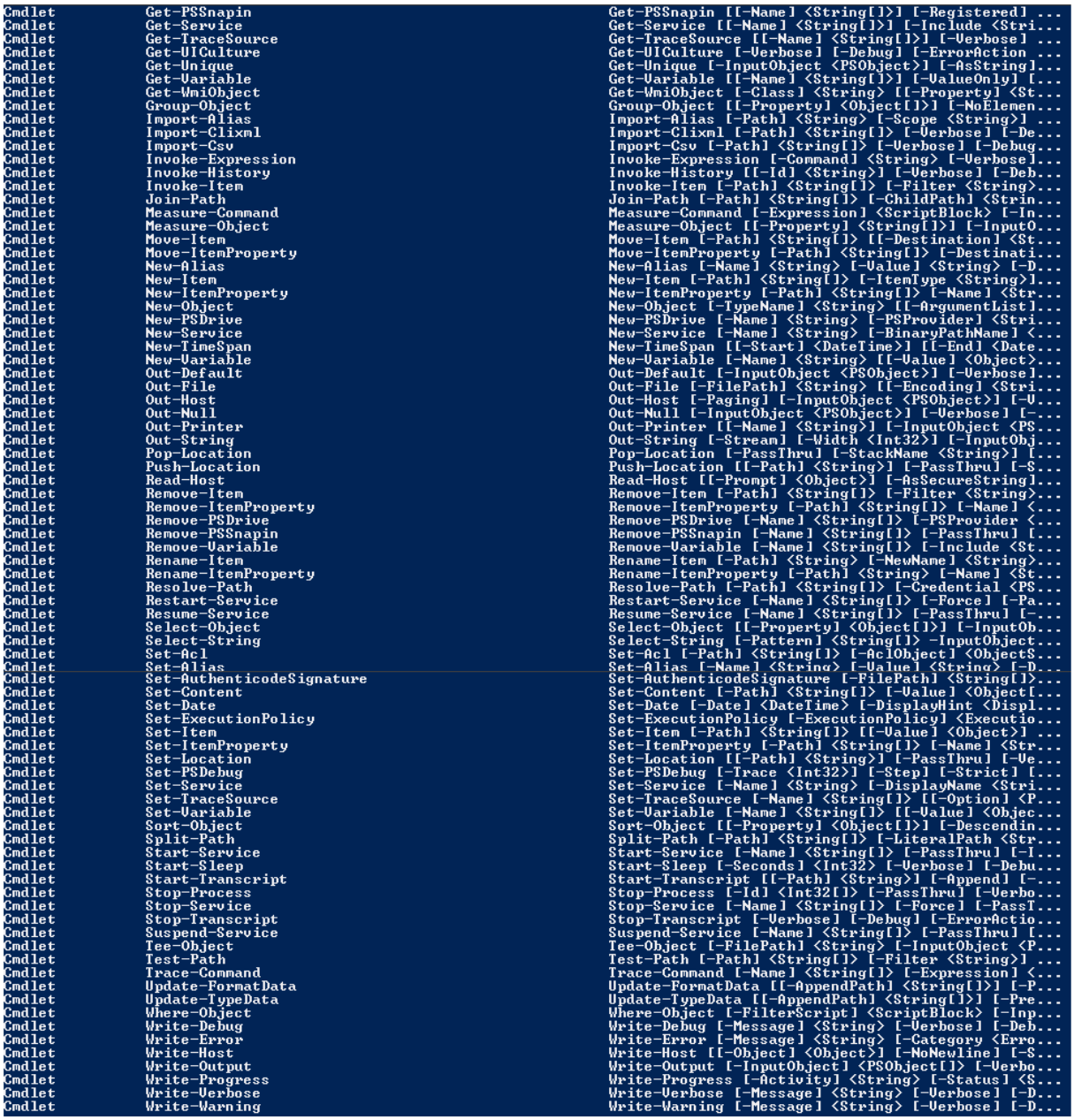
Ceci rend l'apprentissage de *PowerShell* relativement facile, en lisant la commande nous avons déjà une idée de ce qu'elle réalise.

Par exemple, la *cmd-let* **`Get-Command`**, comme son verbe l'indique, permet de faire un **`Get`** (lecture). Le nom **`Command`** quand à lui signifie commandes.

Cette *cmd-let* va donc lister toutes les commandes :

```
PS C:\Documents and Settings\Administrator> get-command

CommandType     Name                                               Definition
-----------     ----                                               ----------
Cmdlet          Add-Content                                        Add-Content [-Path] <String[]> [-Value] <Object[...
Cmdlet          Add-History                                        Add-History [[-InputObject] <PSObject[]>] [-Pass...
Cmdlet          Add-Member                                         Add-Member [-MemberType] <PSMemberTypes> [-Name]...
Cmdlet          Add-PSSnapin                                       Add-PSSnapin [-Name] <String[]> [-PassThru] [-Ve...
Cmdlet          Clear-Content                                      Clear-Content [-Path] <String[]> [-Filter <Strin...
Cmdlet          Clear-Item                                         Clear-Item [-Path] <String[]> [-Force] [-Filter ...
Cmdlet          Clear-ItemProperty                                 Clear-ItemProperty [-Path] <String[]> [-Name] <S...
Cmdlet          Clear-Variable                                     Clear-Variable [-Name] <String[]> [-Include <Str...
Cmdlet          Compare-Object                                     Compare-Object [-ReferenceObject] <PSObject[]> [...
Cmdlet          ConvertFrom-SecureString                           ConvertFrom-SecureString [-SecureString] <Secure...
Cmdlet          Convert-Path                                       Convert-Path [-Path] <String[]> [-Verbose] [-Deb...
Cmdlet          ConvertTo-Html                                     ConvertTo-Html [[-Property] <Object[]>] [-InputO...
Cmdlet          ConvertTo-SecureString                             ConvertTo-SecureString [-String] <String> [[-Sec...
Cmdlet          Copy-Item                                          Copy-Item [-Path] <String[]> [[-Destination] <St...
Cmdlet          Copy-ItemProperty                                  Copy-ItemProperty [-Path] <String[]> [-Destinati...
Cmdlet          Export-Alias                                       Export-Alias [-Path] <String> [[-Name] <String[]>...
Cmdlet          Export-Clixml                                      Export-Clixml [-Path] <String> [-Depth <Int32>] ...
Cmdlet          Export-Console                                     Export-Console [[-Path] <String>] [-Force] [-NoC...
Cmdlet          Export-Csv                                         Export-Csv [-Path] <String> -InputObject <PSObje...
Cmdlet          ForEach-Object                                     ForEach-Object [-Process] <ScriptBlock[]> [-Inpu...
Cmdlet          Format-Custom                                      Format-Custom [[-Property] <Object[]>] [-Depth <...
Cmdlet          Format-List                                        Format-List [[-Property] <Object[]>] [-GroupBy <...
Cmdlet          Format-Table                                       Format-Table [[-Property] <Object[]>] [-AutoSize...
Cmdlet          Format-Wide                                        Format-Wide [[-Property] <Object>] [-AutoSize] [...
Cmdlet          Get-Acl                                            Get-Acl [[-Path] <String[]>] [-Audit] [-Filter <...
Cmdlet          Get-Alias                                          Get-Alias [[-Name] <String[]>] [-Exclude <String...
Cmdlet          Get-AuthenticodeSignature                          Get-AuthenticodeSignature [-FilePath] <String[]>...
Cmdlet          Get-ChildItem                                      Get-ChildItem [[-Path] <String[]>] [[-Filter] <S...
Cmdlet          Get-Command                                        Get-Command [[-ArgumentList] <Object[]>] [-Verb ...
Cmdlet          Get-Content                                        Get-Content [-Path] <String[]> [-ReadCount <Int6...
Cmdlet          Get-Credential                                     Get-Credential [-Credential] <PSCredential> [-Ve...
Cmdlet          Get-Culture                                        Get-Culture [-Verbose] [-Debug] [-ErrorAction <A...
Cmdlet          Get-Date                                           Get-Date [[-Date] <DateTime>] [-Year <Int32>] [-...
Cmdlet          Get-EventLog                                       Get-EventLog [-LogName] <String> [-Newest <Int32...
Cmdlet          Get-ExecutionPolicy                                Get-ExecutionPolicy [-Verbose] [-Debug] [-ErrorA...
Cmdlet          Get-Help                                           Get-Help [[-Name] <String>] [-Category <String[]>...
Cmdlet          Get-History                                        Get-History [[-Id] <Int64[]>] [[-Count] <Int32>]...
Cmdlet          Get-Host                                           Get-Host [-Verbose] [-Debug] [-ErrorAction <Acti...
Cmdlet          Get-Item                                           Get-Item [-Path] <String[]> [-Filter <String>] [...
Cmdlet          Get-ItemProperty                                   Get-ItemProperty [-Path] <String[]> [[-Name] <St...
Cmdlet          Get-Location                                       Get-Location [-PSProvider <String[]>] [-PSDrive ...
Cmdlet          Get-Member                                         Get-Member [[-Name] <String[]>] [-InputObject <P...
Cmdlet          Get-PfxCertificate                                 Get-PfxCertificate [-FilePath] <String[]> [-Verb...
Cmdlet          Get-Process                                        Get-Process [[-Name] <String[]>] [-Verbose] [-De...
Cmdlet          Get-PSDrive                                        Get-PSDrive [[-Name] <String[]>] [-Scope <String...
Cmdlet          Get-PSProvider                                     Get-PSProvider [[-PSProvider] <String[]>] [-Verb...
```



```
Cmdlet          Get-PSSnapin                       Get-PSSnapin [[-Name] <String[]>] [-Registered] ...
Cmdlet          Get-Service                        Get-Service [[-Name] <String[]>] [-Include <Stri...
Cmdlet          Get-TraceSource                    Get-TraceSource [[-Name] <String[]>] [-Verbose] ...
Cmdlet          Get-UICulture                      Get-UICulture [-Verbose] [-Debug] [-ErrorAction ...
Cmdlet          Get-Unique                         Get-Unique [-InputObject <PSObject>] [-AsString]...
Cmdlet          Get-Variable                       Get-Variable [[-Name] <String[]>] [-ValueOnly] [...
Cmdlet          Get-WmiObject                      Get-WmiObject [-Class] <String> [[-Property] <St...
Cmdlet          Group-Object                       Group-Object [[-Property] <Object[]>] [-NoElemen...
Cmdlet          Import-Alias                       Import-Alias [-Path] <String> [-Scope <String>] ...
Cmdlet          Import-Clixml                      Import-Clixml [-Path] <String[]> [-Verbose] [-De...
Cmdlet          Import-Csv                         Import-Csv [-Path] <String[]> [-Verbose] [-Debug...
Cmdlet          Invoke-Expression                  Invoke-Expression [-Command] <String> [-Verbose]...
Cmdlet          Invoke-History                     Invoke-History [[-Id] <String>] [-Verbose] [-Deb...
Cmdlet          Invoke-Item                        Invoke-Item [-Path] <String[]> [-Filter <String>...
Cmdlet          Join-Path                          Join-Path [-Path] <String[]> [-ChildPath] <Strin...
Cmdlet          Measure-Command                    Measure-Command [-Expression] <ScriptBlock> [-In...
Cmdlet          Measure-Object                     Measure-Object [[-Property] <String[]>] [-InputO...
Cmdlet          Move-Item                          Move-Item [-Path] <String[]> [[-Destination] <St...
Cmdlet          Move-ItemProperty                  Move-ItemProperty [-Path] <String[]> [-Destinati...
Cmdlet          New-Alias                          New-Alias [-Name] <String> [-Value] <String> [-D...
Cmdlet          New-Item                           New-Item [-Path] <String[]> [-ItemType <String>]...
Cmdlet          New-ItemProperty                   New-ItemProperty [-Path] <String[]> [-Name] <Str...
Cmdlet          New-Object                         New-Object [-TypeName] <String> [[-ArgumentList]...
Cmdlet          New-PSDrive                        New-PSDrive [-Name] <String> [-PSProvider] <Stri...
Cmdlet          New-Service                        New-Service [-Name] <String> [-BinaryPathName] <...
Cmdlet          New-TimeSpan                       New-TimeSpan [[-Start] <DateTime>] [[-End] <Date...
Cmdlet          New-Variable                       New-Variable [-Name] <String> [[-Value] <Object>...
Cmdlet          Out-Default                        Out-Default [-InputObject <PSObject>] [-Verbose]...
Cmdlet          Out-File                           Out-File [-FilePath] <String> [[-Encoding] <Stri...
Cmdlet          Out-Host                           Out-Host [-Paging] [-InputObject <PSObject>] [-U...
Cmdlet          Out-Null                           Out-Null [-InputObject <PSObject>] [-Verbose] [-...
Cmdlet          Out-Printer                        Out-Printer [[-Name] <String>] [-InputObject <PS...
Cmdlet          Out-String                         Out-String [-Stream] [-Width <Int32>] [-InputObj...
Cmdlet          Pop-Location                       Pop-Location [-PassThru] [-StackName <String>] [...
Cmdlet          Push-Location                      Push-Location [[-Path] <String>] [-PassThru] [-S...
Cmdlet          Read-Host                          Read-Host [[-Prompt] <Object>] [-AsSecureString]...
Cmdlet          Remove-Item                        Remove-Item [-Path] <String[]> [-Filter <String>...
Cmdlet          Remove-ItemProperty                Remove-ItemProperty [-Path] <String[]> [-Name] <...
Cmdlet          Remove-PSDrive                     Remove-PSDrive [-Name] <String[]> [-PSProvider <...
Cmdlet          Remove-PSSnapin                    Remove-PSSnapin [-Name] <String[]> [-PassThru] [...
Cmdlet          Remove-Variable                    Remove-Variable [-Name] <String[]> [-Include <St...
Cmdlet          Rename-Item                        Rename-Item [-Path] <String> [-NewName] <String>...
Cmdlet          Rename-ItemProperty                Rename-ItemProperty [-Path] <String> [-Name] <St...
Cmdlet          Resolve-Path                       Resolve-Path [-Path] <String[]> [-Credential <PS...
Cmdlet          Restart-Service                    Restart-Service [-Name] <String[]> [-Force] [-Pa...
Cmdlet          Resume-Service                     Resume-Service [-Name] <String[]> [-PassThru] [-...
Cmdlet          Select-Object                      Select-Object [[-Property] <Object[]>] [-InputOb...
Cmdlet          Select-String                      Select-String [-Pattern] <String[]> -InputObject...
Cmdlet          Set-Acl                            Set-Acl [-Path] <String[]> [-AclObject] <ObjectS...
Cmdlet          Set-Alias                          Set-Alias [-Name] <String> [-Value] <String> [-D...
Cmdlet          Set-AuthenticodeSignature          Set-AuthenticodeSignature [-FilePath] <String[]>...
Cmdlet          Set-Content                        Set-Content [-Path] <String[]> [-Value] <Object[...
Cmdlet          Set-Date                           Set-Date [-Date] <DateTime> [-DisplayHint <Displ...
Cmdlet          Set-ExecutionPolicy                Set-ExecutionPolicy [-ExecutionPolicy] <Executio...
Cmdlet          Set-Item                           Set-Item [-Path] <String[]> [[-Value] <Object>] ...
Cmdlet          Set-ItemProperty                   Set-ItemProperty [-Path] <String[]> [-Name] <Str...
Cmdlet          Set-Location                       Set-Location [[-Path] <String>] [-PassThru] [-Ve...
Cmdlet          Set-PSDebug                        Set-PSDebug [-Trace <Int32>] [-Step] [-Strict] [...
Cmdlet          Set-Service                        Set-Service [-Name] <String> [-DisplayName <Stri...
Cmdlet          Set-TraceSource                    Set-TraceSource [-Name] <String[]> [[-Option] <P...
Cmdlet          Set-Variable                       Set-Variable [-Name] <String[]> [[-Value] <Objec...
Cmdlet          Sort-Object                        Sort-Object [[-Property] <Object[]>] [-Descendin...
Cmdlet          Split-Path                         Split-Path [-Path] <String[]> [-LiteralPath <Str...
Cmdlet          Start-Service                      Start-Service [-Name] <String[]> [-PassThru] [-I...
Cmdlet          Start-Sleep                        Start-Sleep [-Seconds] <Int32> [-Verbose] [-Debu...
Cmdlet          Start-Transcript                   Start-Transcript [[-Path] <String>] [-Append] [-...
Cmdlet          Stop-Process                       Stop-Process [-Id] <Int32[]> [-PassThru] [-Verbo...
Cmdlet          Stop-Service                       Stop-Service [-Name] <String[]> [-Force] [-PassT...
Cmdlet          Stop-Transcript                    Stop-Transcript [-Verbose] [-Debug] [-ErrorActio...
Cmdlet          Suspend-Service                    Suspend-Service [-Name] <String[]> [-PassThru] [...
Cmdlet          Tee-Object                         Tee-Object [-FilePath] <String> [-InputObject <P...
Cmdlet          Test-Path                          Test-Path [-Path] <String[]> [-Filter <String>] ...
Cmdlet          Trace-Command                      Trace-Command [-Name] <String[]> [-Expression] <...
Cmdlet          Update-FormatData                  Update-FormatData [[-AppendPath] <String[]>] [-P...
Cmdlet          Update-TypeData                    Update-TypeData [[-AppendPath] <String[]>] [-Pre...
Cmdlet          Where-Object                       Where-Object [-FilterScript] <ScriptBlock> [-Inp...
Cmdlet          Write-Debug                        Write-Debug [-Message] <String> [-Verbose] [-Deb...
Cmdlet          Write-Error                        Write-Error [-Message] <String> [-Category <Erro...
Cmdlet          Write-Host                         Write-Host [[-Object] <Object>] [-NoNewline] [-S...
Cmdlet          Write-Output                       Write-Output [-InputObject] <PSObject[]> [-Verbo...
Cmdlet          Write-Progress                     Write-Progress [-Activity] <String> [-Status] <S...
Cmdlet          Write-Verbose                      Write-Verbose [-Message] <String> [-Verbose] [-D...
Cmdlet          Write-Warning                      Write-Warning [-Message] <String> [-Verbose] [-D...
```

Pour consulter l'aide d'une *cmd-let*, il suffit de taper **`help nom_de_la_cmdlet -full`**, pour obtenir l'aide complète associée à cette *cmdlet* en question.

2.8.4 Les arguments de commande

Si l'utilisateur ne rentre pas tous les arguments nécessaires à une commande, *Powershell* demandera de les fournir. Il n'y aura donc pas d'erreur renvoyée comme sous *Unix/Linux* ou encore cmd.exe

2.8.5 Les alias

Un alias est un nom ou terme facile à mémoriser qui est utilisé à la place d'un autre nom ou mot.

Dans le cas de *PowerShell*, un alias est un nom que l'ont choisi pour remplacer une commande (cmd-let) *PowerShell*.

Par exemple, les alias disponibles par défaut dans *PowerShell* tels que "dir" ou encore "ls" sont utilisés pour remplacer la commande **`Get-ChildItem`**, ceci rend *PowerShell* facilement utilisable pour une personne habituée à utiliser les commandes traditionnelles d'*Unix/Linux* ou encore *Windows*.

D'autres alias peuvent être spécifiés à l'aide de la commande **`Set-Alias`**.

Remarque : Il est aussi possible de définir des alias dans la plupart des *shell* d'*Unix* et *Linux*.

2.8.6 La gestion d'aide

La gestion d'aide a été uniformisée : la commande **`help`** permet d'accéder à l'ensemble des aides disponibles et consulter l'aide associée à une méthode.

Il existe plusieurs niveaux d'aide pour la commande **`help`** :

- **`help`**, qui affiche l'aide de base (*Name*, *Synopsis* (brève description), *Detailed Description*, *Related Links*, *Remarks*).
- **`help -full`**, qui fournit l'aide totale disponible pour une commande (*Name*, *Synopsis* (brève description), *Syntax*, *Detailed Description*, *Parameters*, *Input Type*, *Return Type*, *Notes*, *Examples*, *Related Links*).
- **`help -detailed`**, qui fournit une aide détaillée pour une commande (*Name*, *Synopsis*, *Syntax*, *Detailed Description*, *Parameters*, *Examples*, *Remarks*).
- **`help -examples`**, qui affiche quelques exemples pour l'utilisation de la commande concernée.
- **`help -parameter nom_du_parametre`**, qui affiche l'aide associée à un paramètre de commande spécifique.

2.8.7 Les boucles

Dans *PowerShell*, il est possible de créer des *scripts* contenant des boucles, ceci très facilement.

De plus il est possible de créer une boucle analysant les données reçues, pour afficher uniquement certains paramètres ou pour effectuer des opérations avec les informations reçues.

Certains exemples sont disponibles en page 115, paragraphe 2.14

2.8.8 Affichage des données

Lors de l'utilisation d'une commande dans *PowerShell*, il est possible d'afficher les informations reçues de plusieurs manières différentes, par exemple en ligne ou en colonne, avec un format ressemblant à Unix ou MS-DOS.

Ces différents affichages peuvent être spécifiés à l'aide des commandes **`Format-List`**, **`Format-Table`**, **`Format-Wide`** et **`Format-Custom`**.

Exemple :

Lors de l'utilisation de la commande **`get-command`** (qui liste les commandes disponibles dans *PowerShell*), les informations sont affichées comme suit :

```
CommandType     Name                                               Definition
-----------     ----                                               ----------
Cmdlet          Add-Content                                        Add-Content [-Path] <String[]> [-Value] <Object[...
Cmdlet          Add-History                                        Add-History [[-InputObject] <PSObject[]>] [-Pass...
Cmdlet          Add-Member                                         Add-Member [-MemberType] <PSMemberTypes> [-Name]...
Cmdlet          Add-PSSnapin                                       Add-PSSnapin [-Name] <String[]> [-PassThru] [-Ve...
Cmdlet          Clear-Content                                      Clear-Content [-Path] <String[]> [-Filter <Strin...
Cmdlet          Clear-Item                                         Clear-Item [-Path] <String[]> [-Force] [-Filter ...
Cmdlet          Clear-ItemProperty                                 Clear-ItemProperty [-Path] <String[]> [-Name] <S...
Cmdlet          Clear-Variable                                     Clear-Variable [-Name] <String[]> [-Include <Str...
Cmdlet          Compare-Object                                     Compare-Object [-ReferenceObject] <PSObject[]> [...
Cmdlet          ConvertFrom-SecureString                           ConvertFrom-SecureString [-SecureString] <Secure...
Cmdlet          Convert-Path                                       Convert-Path [-Path] <String[]> [-Verbose] [-Deb...
Cmdlet          ConvertTo-Html                                     ConvertTo-Html [[-Property] <Object[]>] [-InputO...
Cmdlet          ConvertTo-SecureString                             ConvertTo-SecureString [-String] <String> [[-Sec...
Cmdlet          Copy-Item                                          Copy-Item [-Path] <String[]> [[-Destination] <St...
Cmdlet          Copy-ItemProperty                                  Copy-ItemProperty [-Path] <String[]> [-Destinati...
Cmdlet          Export-Alias                                       Export-Alias [-Path] <String> [[-Name] <String[]...
Cmdlet          Export-Clixml                                      Export-Clixml [-Path] <String> [-Depth <Int32>] ...
Cmdlet          Export-Console                                     Export-Console [[-Path] <String>] [-Force] [-NoC...
Cmdlet          Export-Csv                                         Export-Csv [-Path] <String> -InputObject <PSObje...
Cmdlet          ForEach-Object                                     ForEach-Object [-Process] <ScriptBlock[]> [-Inpu...
Cmdlet          Format-Custom                                      Format-Custom [[-Property] <Object[]>] [-Depth <...
Cmdlet          Format-List                                        Format-List [[-Property] <Object[]>] [-GroupBy <...
Cmdlet          Format-Table                                       Format-Table [[-Property] <Object[]>] [-AutoSize...
Cmdlet          Format-Wide                                        Format-Wide [[-Property] <Object>] [-AutoSize] [...
```

Ils sont donc affichés en ligne, cet affichage par défaut correspond à la commande **`Format-Table`**.

Ces données peuvent être redirigées vers une commande **`Format-List`** par exemple, ce qui modifiera l'affichage :

**`Get-command	Format-List`**	Redirection des informations reçues pour être affichées à l'aide de la commande **`Format-List`**

```
Name            : Add-Content
CommandType     : Cmdlet
Definition      : Add-Content [-Path] <String[]> [-Value] <Object[]> [-PassThru] [-Filter <String>] [-Include <String[
                  ]>] [-Exclude <String[]>] [-Force] [-Credential <PSCredential>] [-Verbose] [-Debug] [-ErrorAction <A
                  ctionPreference>] [-ErrorVariable <String>] [-OutVariable <String>] [-OutBuffer <Int32>] [-WhatIf] [
                  -Confirm] [-Encoding <FileSystemCmdletProviderEncoding>]
                  Add-Content [-LiteralPath] <String[]> [-Value] <Object[]> [-PassThru] [-Filter <String>] [-Include <
                  String[]>] [-Exclude <String[]>] [-Force] [-Credential <PSCredential>] [-Verbose] [-Debug] [-ErrorAc
                  tion <ActionPreference>] [-ErrorVariable <String>] [-OutVariable <String>] [-OutBuffer <Int32>] [-Wh
                  atIf] [-Confirm] [-Encoding <FileSystemCmdletProviderEncoding>]

Path            :
AssemblyInfo    :
DLL             : C:\WINDOWS\assembly\GAC_MSIL\Microsoft.PowerShell.Commands.Management\1.0.0.0__31bf3856ad364e35\Micr
                  osoft.PowerShell.Commands.Management.dll
HelpFile        : Microsoft.PowerShell.Commands.Management.dll-Help.xml
ParameterSets   : {Path, LiteralPath}
ImplementingType : Microsoft.PowerShell.Commands.AddContentCommand
Verb            : Add
Noun            : Content

Name            : Add-History
CommandType     : Cmdlet
Definition      : Add-History [[-InputObject] <PSObject[]>] [-Passthru] [-Verbose] [-Debug] [-ErrorAction <ActionPrefe
                  rence>] [-ErrorVariable <String>] [-OutVariable <String>] [-OutBuffer <Int32>]

Path            :
AssemblyInfo    :
DLL             : C:\WINDOWS\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Managemen
                  t.Automation.dll
HelpFile        : System.Management.Automation.dll-Help.xml
ParameterSets   : {__AllParameterSets}
ImplementingType : Microsoft.PowerShell.Commands.AddHistoryCommand
Verb            : Add
Noun            : History

Name            : Add-Member
CommandType     : Cmdlet
Definition      : Add-Member [-MemberType] <PSMemberTypes> [-Name] <String> [[-Value] <Object>] [[-SecondValue] <Objec
                  t>] -InputObject <PSObject> [-Force] [-PassThru] [-Verbose] [-Debug] [-ErrorAction <ActionPreference
                  >] [-ErrorVariable <String>] [-OutVariable <String>] [-OutBuffer <Int32>]
```

Nous remarquons que nous obtenons un tout autre affichage !

Ces commandes peuvent aussi être redirigées vers la commande **Format-Wide** :

`Get-command | Format-Wide`	Redirection des informations reçues pour être affichées à l'aide de la commande **`Format-Wide`**

La commande **`Format-Custom`** affiche les informations sous forme de code orienté objet.

2.8.9 Les *providers*

Les *providers* sont des interfaces logicielles qui permettent d'accéder à différentes structures de données de nature différentes (système de fichiers, base de registre, certificats, etc.)

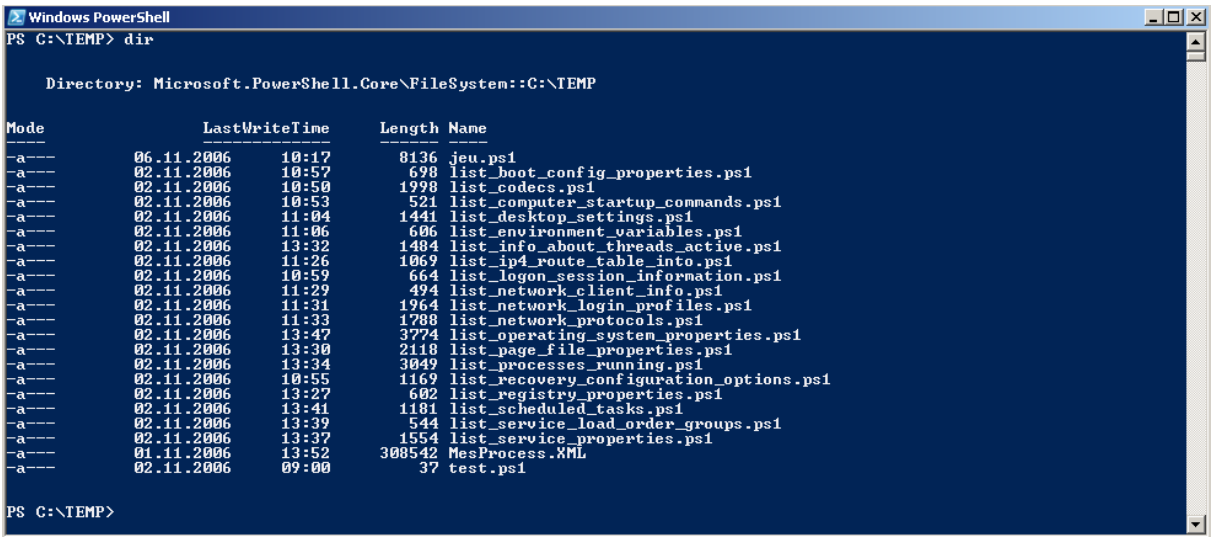
Les *providers* permettent de créer des disques (virtuellement) pouvant être accédés par des *cmd-let* déjà existants.

Ils nous permettent donc de parcourir et naviguer sur différents types de structure de donnée tels que le système de fichiers, la base de registre, *Active Directory* et *WMI* (*Windows Management Instrumentation*) avec les mêmes commandes !

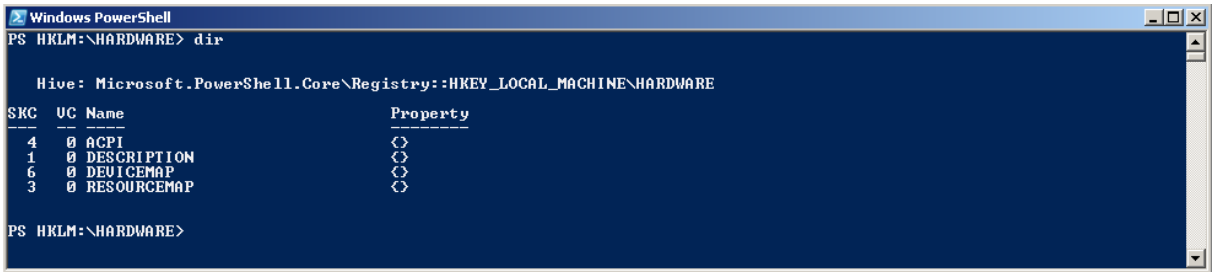
Exemple :

Il est possible d'utiliser la même commande pour lister les dossiers et fichier d'un répertoire ou énumérer le contenu d'une clef de la base de registre.

"Dir" du système de fichiers :



"Dir" de la base de registre :



Ceci nous permet donc de naviguer très facilement dans des structures de données très différentes les unes des autres.

Remarque : Il est possible de créer son propre *provider*. Pour cela consulter le lien http://msdn2.microsoft.com/en-us/library/ms714636.aspx

2.9 Que faire avec *PowerShell* ?

PowerShell peut être utilisé en local pour :

- **Créer des *scripts***, *powerShell* possède son propre langage de programmation.
- **Modifier le système de fichiers** (créer/supprimer des répertoires ou fichiers, modifier la base de registre.
- **Lancer des programmes**, par exemple pour lancer l'exécution de *notepad*, taper *notepad* dans l'invite de commandes.
- **Lister diverses informations système** présentes sur l'ordinateur local ainsi que sur les ordinateurs du réseau (ils doivent pour cela exécuter le service *RPC* (*Remote Procedure Call*) ainsi que *WMI*.
  De plus le compte utilisé doit faire partie du groupe *Administrators* sur le pc distant.
  Il n'est pas nécessaire d'avoir *PowerShell* installé sur les ordinateurs distants, il faut uniquement *WMI*, qui est déjà par défaut dans les installations *Xp* et *Vista*.

Toutes ces différentes possibilités ont été testées et fonctionnent correctement.

En ce qui concerne l'accès aux informations sur un pc distant en utilisant *PowerShell*, j'ai effectué une demande depuis un poste *Windows XP* vers un poste *Windows Vista* (*PowerShell* a uniquement été installé sur le poste *Windows XP*).
Les pare-feu *Windows* ont du être désactivés et les différents ordinateurs ont dû être ajoutés à un domaine.

2.10 Sécurité dans *Powershell* ?

Etant donné que *Powershell* peut exécuter toutes sortes de *scripts*, des personnes malintentionnées pourraient créer des *scripts* malicieux.

Pour palier à ce problème, *Powershell* a été conçu avec 4 niveaux de sécurité :

- Mode **`Restricted`**, aucun *script* ne peut être exécuté. C'est le mode par défaut dans la version RC2.
- Mode **`AllSigned`**, seuls les *scripts* signés peuvent être lancés.
- Mode **`RemoteSigned`**, tous les *scripts* locaux peuvent être exécutés, les *scripts* téléchargés doivent être signés pour être exécutés.
- Mode **`Unrestricted`**, n'importe quel *script* peut être exécuté.

La politique de sécurité (ou stratégie de sécurité d'exécution) appliquée aux *scripts* ou à toutes autres commandes saisies devant être exécutées, est régie par la clef de registre **`ExecutionPolicy`** se trouvant dans :

`HKLM\Software\Microsoft\Powershell\1\ShellIds\Microsoft.Powershell\`

De plus, les extensions de script *PowerShell* ne sont pas associées au *shell* dans l'explorateur. Pour être exécutés, les *scripts* doivent être lancés directement dans *Powershell*.
Pour plus d'informations, consulter l'aide disponible dans *PowerShell* :
`help about_signing|more`

2.10.1 Changer le niveau de sécurité *ExecutionPolicy*

Pour changer le niveau de sécurité de l'utilisation de *scripts*, lancer la commande suivante dans *PowerShell* :

`Set-ExecutionPolicy <policy-name>`	Change le niveau de sécurité

Exemple :

`Set-ExecutionPolicy Unrestricted`	Tous les *scripts* peuvent être exécutés.

2.10.2 Différents types de certificats

Afin d'ajouter une signature à un *script*, il est nécessaire de disposer d'un certificat.

Il y a 2 types de certificats possibles pour signer un *script* :

- **Certificat créé par une autorité de certification (*Trusted Root*)**, qui vérifie l'identité de la personne demandant un certificat. Si un certificat nous est délivré de cette manière, il nous sera possible de créer des *scripts* pouvant s'exécuter sur tous les ordinateurs *Windows* faisant confiance à cette même autorité de certification.

- **Certificat auto-signé**, ce qui nous permet de signer nos propres *scripts*. Ces scripts pourront s'exécuter sur notre ordinateur.
  Cette méthode est expliquée en détail dans le paragraphe suivant.

2.10.3 Créer un certificat auto-signé

Il est possible de créer un certificat auto-signé à l'aide de l'outil **`MakeCert.exe`** (outil en lignes de commandes CLI), disponible gratuitement dans le Kit de développement logiciel *Microsoft .NET Framework SDK 2.0* (exécutable de 377MB, 442MB d'occupation disque après installation), qui peut être téléchargé à l'adresse :
http://www.microsoft.com/downloads/details.aspx?familyid=FE6F2099-B7B4-4F47-A244-C96D69C35DEC&displaylang=fr

Après installation de ce Kit, ouvrir la fenêtre d'invite de commandes *SDK* (Start – Program Files – Microsoft .NET Framework SDK v2.0 – Invite de commandes du Kit de développement SDK) puis effectuer les commandes suivantes :

`makecert -n "CN=PowerShell Local Certificate Root" -a sha1 -eku 1.3.6.1.5.5.7.3.3 -r -sv root.pvk root.cer -ss Root -sr localMachine`	Crée une autorité de certification locale à la machine.
`makecert -pe -n "CN=PowerShell User" -ss MY -a sha1 -eku 1.3.6.1.5.5.7.3.3 -iv root.pvk -ic root.cer`	Génère un certificat personnel (nommé **`My`**) à partir de l'autorité de certification locale.

Lors de l'exécution de ces 2 commandes, l'outil **`MakeCert.exe`** nous demandera de choisir un mot de passe afin de générer la clé privée.

Remarque : Lors de la création d'un certificat auto-signé, il est conseillé d'activer la protection forte de la clé privée (activation décrite en page suivante), ceci empêche les programmes malicieux de pouvoir *signer* des scripts à notre insu !

2.10.4 Vérification que le certificat auto-signé a bien été créé

Dans *PowerShell*, exécuter la commande suivante :

`get-childitem cert:\CurrentUser\My -codesigning`	Vérifie si le certificat nommé **`My`** existe. Si tel est le cas *PowerShell* affichera son *Thumbprint* (signature numérique)

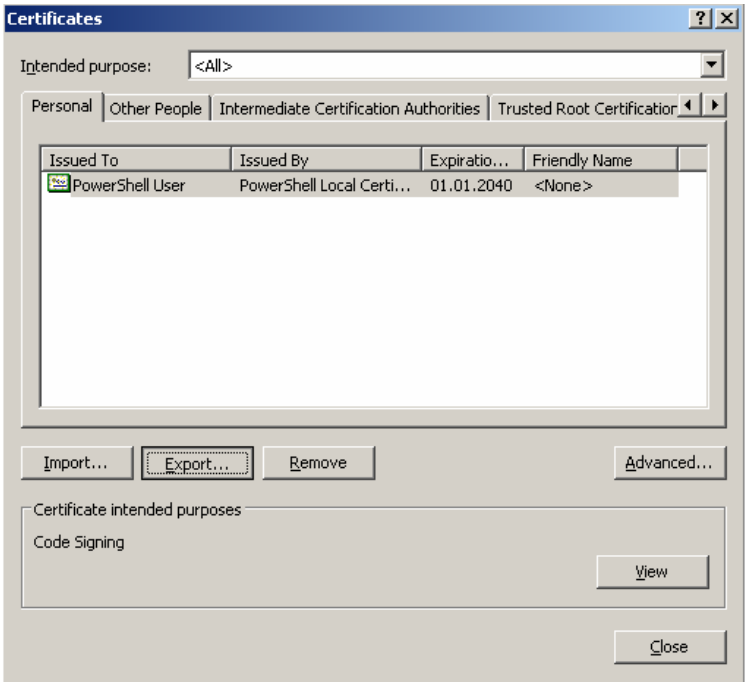
2.10.5 Activer la protection forte de la clé privée

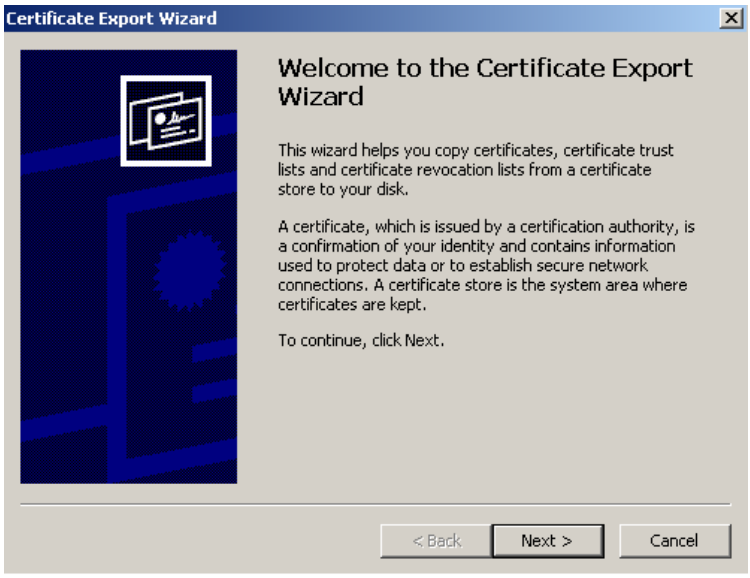
L'activation de la protection forte de la clé privée peut se faire à l'aide de l'outil **`certmgr.exe`** (outil graphique GUI), qui est compris dans *Microsoft Framework .NET SDK 2.0*.

Exécuter dans *PowerShell* les commandes suivantes :

`cd 'C:\Program Files\Microsoft.NET\SDK\v2.0\BIN'`	Dans le répertoire contenant l'outil certmgr.exe
`.\certmgr.exe`	Lancement de **`certmgr.exe`** `.\` signifie que l'on lance un programme dans le répertoire courant

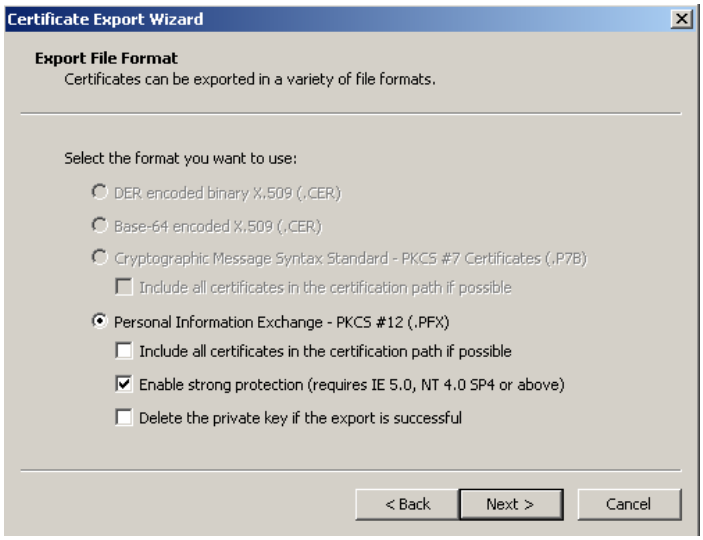
La fenêtre suivante s'affiche à l'écran, sélectionner le certificat créé puis cliquer sur "Export…" :



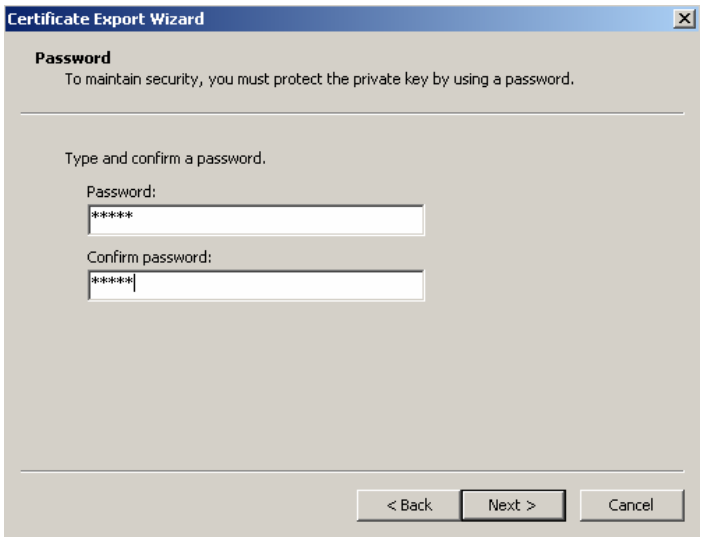


Exporter la clé privée :

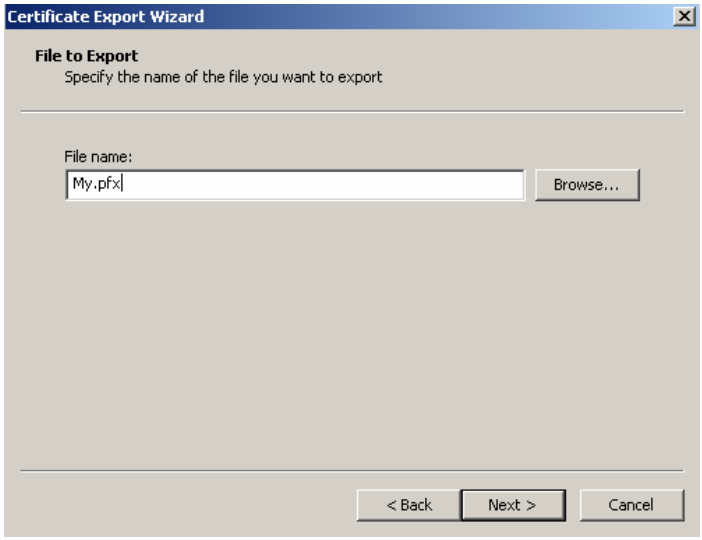
Activer la protection forte :



Définir un mot de passe afin de protéger la clé privée :



Spécifier le nom du fichier à exporter :



Cliquer sur "Finish" pour terminer l'exportation :

2.10.6 Signer un *script*

Effectuer sous *PowerShell* les commandes suivantes :

`$cert = @(Get-ChildItem cert:\CurrentUser\My -codesigning)[0]`	Spécifie l'utilisation du certificat nommé **`My`**
`Set-AuthenticodeSignature file.ps1 $cert`	Signe le *script* nommé **`file.ps1`**

2.11 Utilisation intéressante de *PowerShell* pour ce projet?

Après avoir étudié les différentes possibilités qu'offre *PowerShell*, peut-on utiliser ce nouveau *shell* pour faciliter la création d'images ou leur déploiement ?

En accédant avec *PowerShell* à un ordinateur distant, il est uniquement possible de faire des opérations de lecture, on ne peut pas modifier le système de fichier ni la base de registre, ce qui n'est donc pas très intéressant dans le cadre de ce projet.

Pour la création et installation d'images à l'aide d'*ImageX*, il faut uniquement exécuter une ligne de commande sous *Windows PE*, il est donc inutile de créer un *script* uniquement pour ceci.

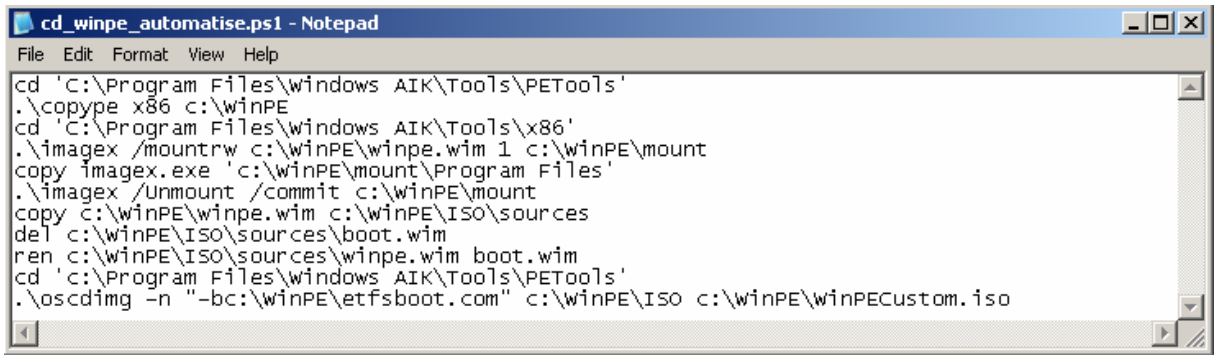
Pour le déploiement d'images, en utilisant un serveur *WDS*, nous devons *booter* chaque pc à l'aide de *PXE* pour nous connecter à notre serveur *WDS* et choisir une image. *PowerShell* ne permet pas d'éveiller (démarrer) sur *PXE* un pc du réseau à distance, il n'est donc pas intéressant de ce point de vue.

La **seule utilité que pourrait avoir *PowerShell* dans le cadre des images *Vista***, est d'automatiser le processus de création d'un CD ou une clé USB *bootable Windows PE*. Ceci peut se faire en créant un *script* approprié, que l'on pourra lancer pour effectuer ces diverses opérations automatiquement.

2.11.1 Créer un script afin d'automatiser le processus de création d'un CD *bootable Windows PE*

Dans ce paragraphe, nous allons créer un *script* afin d'automatiser la création d'une ISO *bootable* de *Windows PE* (*Windows PE* personnalisé en incluant l'outil *ImageX*).
Cette ISO pourra ensuite être gravée directement sur CD ou DVD.

En réalité, ce n'est pas un "vrai" *script*, car il n'y a pas vraiment de code, il s'agit juste ici d'une suite de commandes à exécuter. On aurait aussi pu créer un fichier .BAT avec ces commandes, qui s'exécuterait dans cmd.exe. Le *script PowerShell* est donc le suivant :



Ce *script* est un peu différent des commandes exécutées normalement sur cmd.exe, ceci car les divers outils ne sont pas associés à *PowerShell* lors de la commande **copype** (contrairement à l'exécution de cette commande dans cmd.exe permettant l'utilisation des divers outils tels que *imagex* sans spécifier leur emplacement).

Explications :

cd 'C:\Program Files\ Windows AIK\Tools\PETools'	Dans le répertoire contenant les fichiers de *Windows PE*
.\copype x86 c :\WinPE	Prépare les fichiers de *Windows PE* pour traitement vers **c :\WinPE**
cd 'C:\Program Files\ Windows AIK\Tools\x86'	Dans le répertoire contenant *imagex*
.\imagex /mountrw c:\WinPE\winpe.wim 1 c:\WinPE\mount	"monte" l'image de *Windows PE* (**winpe.wim**) vers le répertoire **c:\WinPE\mount.** L'option **/mountrw** monte l'image en *read-write* (lecture-écriture). C'est vers ce répertoire que imagex.exe va être copié.
copy imagex.exe 'c:\WinPE\mount\Program Files'	Copie **imagex.exe** vers **c:\WinPE\mount\Program Files**
.\imagex /Unmount /commit c:\WinPE\mount	Démonte l'image et enregistre les modifications avec l'option **/commit**
copy c:\WinPE\winpe.wim c:\WinPE\ISO\sources	Copie l'image personnalisée de *Windows PE* qui vient d'être créée vers le répertoire de création ISO
del c:\WinPE\ISO\sources\boot.wim	Supprime l'ancienne version de *Windows PE*
ren c:\WinPE\ISO\sources\winpe.wim boot.wim	Renomme notre image personnalisée de *Windows PE*
cd 'C:\Program Files\ Windows AIK\Tools\PETools'	Dans le répertoire contenant l'outil OSCDIMG
.\oscdimg -n "-bc:\WinPE\etfsboot.com" c:\WinPE\ISO c:\WinPE\WinPECustom.iso	Utilise l'outil **oscdimg** afin de créer une

	ISO *bootable* de *Windows PE*, pouvant être gravée sur CD ou DVD. Noter les guillemets, voir la remarque suivante pour plus d'informations.

Le temps d'exécution de ce *script PowerShell* (.ps1) est de 1 minute 11 secondes (*script* exécuté sur un pc avec processeur Intel Pentium 4 1.7GHz, 256MB RAM, disque dur IDE 7200tours/minute).

Remarque : Lors de la commande **`.\oscdimg -n "-bc:\WinPE\etfsboot.com" c:\WinPE\ISO c:\WinPE\WinPECustom.iso`**, il est nécessaire (avec *PowerShell*) de mettre des guillemets pour le deuxième paramètre de cette commande.
Si cette commande est effectuée sans guillemets, *PowerShell* va séparer le deuxième paramètre comme suit :

`-bc:\WinPE\etfsboot.com` deviendra **`-bc:  \WinPE\etfsboot.com`**

Cette option sera donc séparée en deux, et son exécution ne pourra fonctionner.

Ces séparations ont lieu lors de l'exécution de commandes contenant des paramètres commençant par des signes négatifs "-".

Il faut donc entourer avec des guillemets tous les paramètres commençant par des signes négatifs "-".

2.12 Utilité de *PowerShell* pour des études orientées *Forensics*?

Pour pouvoir utiliser *PowerShell*, il faut l'installer sur un ordinateur, de plus il faut avoir une installation préalable de *Microsoft Framework .NET 2.0*.

Il n'est donc pas possible de mettre *PowerShell* sur un CD et de l'exécuter directement depuis le CD et donc par conséquent, on ne peut pas l'utiliser pour des études orientées *Forensics*.

2.13 Utilisation de *PowerShell* afin d'automatiser la sécurisation d'un poste *Windows XP*

Il est relativement facile d'automatiser la sécurisation d'un poste *Windows XP* en utilisant un domaine. Cependant, toutes les entreprises ne disposent pas forcément d'un domaine, notamment les petites entreprises.

Le but de ce paragraphe est de montrer comment cette tache peut être automatisée à l'aide de *PowerShell*.

Habituellement les ingénieurs souhaitant sécuriser un poste *XP* utilisent *MMC* (*Microsoft Management Console*), qui est une interface graphique. Il leur faut ensuite aller chercher chaque Policy et lui donner la valeur souhaitée (exemple : ***Start*** – ***Settings*** – ***Control Panel*** – ***Administrative Tools*** – ***Local Security Policy*** – ***Security Settings*** – ***Local Policies*** – ***Security Options***, puis mettre la règle ***Network security : Do not store LAN Manager hash on next password change*** sur ***Enabled***), ce qui constitue une tâche pouvant prendre un certain temps et être très répétitive (par exemple sécuriser 20 postes *XP* de cette manière).

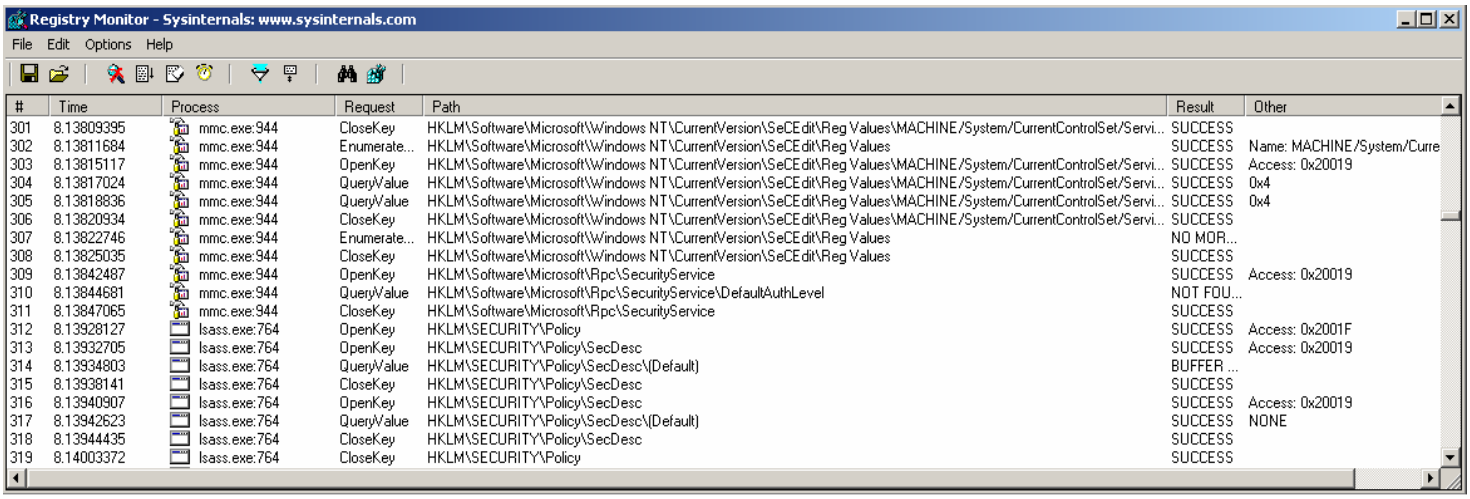
La liste complète des règles à effectuer pour sécuriser un poste *Windows XP* est disponible au laboratoire, consulter le classeur "**Sécuriser un poste client *Windows XP SP2***"

Avec *PowerShell*, il n'est pas possible d'accéder à *MMC* directement, cependant il est possible d'utiliser l'outil *RegMon* (*Registry Monitor*), qui est un outil gratuit créé par *SysInternals* (cette entreprise a été rachetée par *Microsoft*) nous permettant de voir en temps réel les clés qui sont modifiées dans la base de registre.

Donc, en utilisant *RegMon* et en changeant dans *MMC* la valeur d'une règle, il nous sera possible de voir en temps réel toutes les clés qui seront modifiées dans la base de registre, et avec *PowerShell* il sera alors possible de créer un *script* accédant à la base de registre afin de modifier ces mêmes clés.
Ce *script* pourra par la suite être exécuté sur chaque poste client à sécuriser.

L'outil *Regmon* peut être téléchargé à l'adresse :
http://www.microsoft.com/technet/sysinternals/ProcessesAndThreads/Regmon.mspx

Cet outil dispose d'une interface graphique nous montrant tous les accès à la base de registre ainsi que le processus qui effectue ces accès :

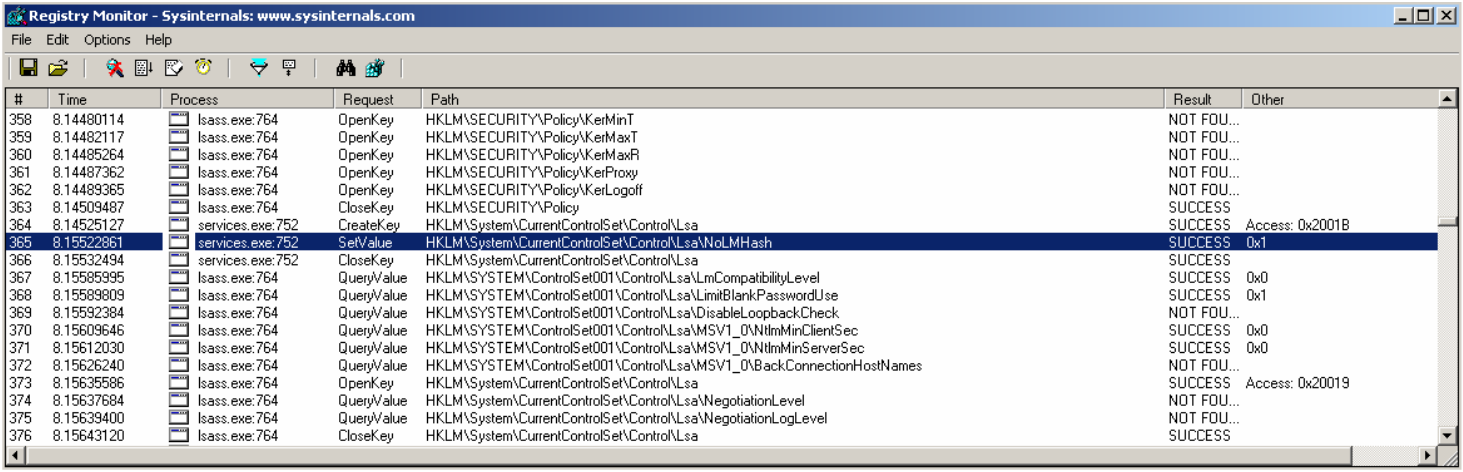


#	Time	Process	Request	Path	Result	Other
301	8.13809395	mmc.exe:944	CloseKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values\MACHINE/System/CurrentControlSet/Servi...	SUCCESS	
302	8.13811684	mmc.exe:944	Enumerate...	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values	SUCCESS	Name: MACHINE/System/Curre
303	8.13815117	mmc.exe:944	OpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values\MACHINE/System/CurrentControlSet/Servi...	SUCCESS	Access: 0x20019
304	8.13817024	mmc.exe:944	QueryValue	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values\MACHINE/System/CurrentControlSet/Servi...	SUCCESS	0x4
305	8.13818836	mmc.exe:944	QueryValue	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values\MACHINE/System/CurrentControlSet/Servi...	SUCCESS	0x4
306	8.13820934	mmc.exe:944	CloseKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values\MACHINE/System/CurrentControlSet/Servi...	SUCCESS	
307	8.13822746	mmc.exe:944	Enumerate...	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values	NO MOR...	
308	8.13825035	mmc.exe:944	CloseKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\SeCEdit\Reg Values	SUCCESS	
309	8.13842487	mmc.exe:944	OpenKey	HKLM\Software\Microsoft\Rpc\SecurityService	SUCCESS	Access: 0x20019
310	8.13844681	mmc.exe:944	QueryValue	HKLM\Software\Microsoft\Rpc\SecurityService\DefaultAuthLevel	NOT FOU...	
311	8.13847065	mmc.exe:944	CloseKey	HKLM\Software\Microsoft\Rpc\SecurityService	SUCCESS	
312	8.13928127	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy	SUCCESS	Access: 0x2001F
313	8.13932705	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Access: 0x20019
314	8.13934803	lsass.exe:764	QueryValue	HKLM\SECURITY\Policy\SecDesc\(Default)	BUFFER ...	
315	8.13938141	lsass.exe:764	CloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
316	8.13940907	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Access: 0x20019
317	8.13942623	lsass.exe:764	QueryValue	HKLM\SECURITY\Policy\SecDesc\(Default)	SUCCESS	NONE
318	8.13944435	lsass.exe:764	CloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
319	8.14003372	lsass.exe:764	CloseKey	HKLM\SECURITY\Policy	SUCCESS	

Dans la colonne *Request*, nous pouvons voir les différentes requêtes qui sont effectuées par les processus.

La requête qui modifie une valeur dans la base de registre est la requête **SetValue**.

Nous devons donc parcourir les données affichées par *RegMon* à la recherche d'une requête de ce type.



#	Time	Process	Request	Path	Result	Other
358	8.14480114	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\KerMinT	NOT FOU...	
359	8.14482117	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\KerMaxT	NOT FOU...	
360	8.14485264	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\KerMaxR	NOT FOU...	
361	8.14487362	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\KerProxy	NOT FOU...	
362	8.14489365	lsass.exe:764	OpenKey	HKLM\SECURITY\Policy\KerLogoff	NOT FOU...	
363	8.14509487	lsass.exe:764	CloseKey	HKLM\SECURITY\Policy	SUCCESS	
364	8.14525127	services.exe:752	CreateKey	HKLM\System\CurrentControlSet\Control\Lsa	SUCCESS	Access: 0x2001B
365	8.15522861	services.exe:752	SetValue	HKLM\System\CurrentControlSet\Control\Lsa\NoLMHash	SUCCESS	0x1
366	8.15532494	services.exe:752	CloseKey	HKLM\System\CurrentControlSet\Control\Lsa	SUCCESS	
367	8.15585995	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\LmCompatibilityLevel	SUCCESS	0x0
368	8.15589809	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\LimitBlankPasswordUse	SUCCESS	0x1
369	8.15592384	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\DisableLoopbackCheck	NOT FOU...	
370	8.15609646	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\MSV1_0\NtlmMinClientSec	SUCCESS	0x0
371	8.15612030	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\MSV1_0\NtlmMinServerSec	SUCCESS	0x0
372	8.15626240	lsass.exe:764	QueryValue	HKLM\SYSTEM\ControlSet001\Control\Lsa\MSV1_0\BackConnectionHostNames	NOT FOU...	
373	8.15635586	lsass.exe:764	OpenKey	HKLM\System\CurrentControlSet\Control\Lsa	SUCCESS	Access: 0x20019
374	8.15637684	lsass.exe:764	QueryValue	HKLM\System\CurrentControlSet\Control\Lsa\NegotiationLevel	NOT FOU...	
375	8.15639400	lsass.exe:764	QueryValue	HKLM\System\CurrentControlSet\Control\Lsa\NegotiationLogLevel	NOT FOU...	
376	8.15643120	lsass.exe:764	CloseKey	HKLM\System\CurrentControlSet\Control\Lsa	SUCCESS	

En parcourant les données affichées par *RegMon*, on constate qu'une seule clé est modifiée dans la base de registre lorsque l'on change la règle ***Network security : Do not store LAN Manager hash on next password change***

Cette clé se trouve dans **`HKLM\System\CurrentControlSet\Control\Lsa\`** et se nomme **`NoLMHash`**.

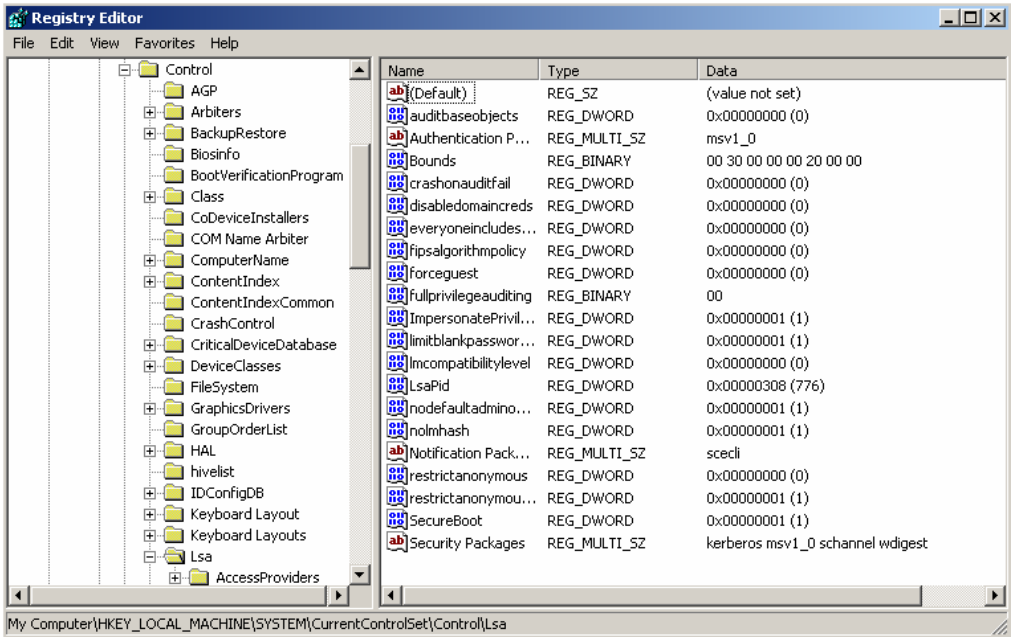
Il nous suffit maintenant de créer un petit *script* avec *PowerShell* afin de changer automatiquement la valeur de cette clé dans la base de registre.

Le *script* sera donc le suivant :

Explications :

`cd HKLM:\System\CurrentControlSet\Control\Lsa`	Dans la base de registre contenant la clé à modifier
`Set-itemproperty . -name nolmhash -value 1`	Modifie la clé nommée **`nolmhash`** avec la valeur 1. Le point "." après **`Set-itemproperty`** signifie que la clé spécifiée se trouve dans le "répertoire" courant

Après exécution de ce petit *script*, vérifions dans la base de registre si la clé est bel et bien modifiée :



Nous constatons que la clé est bien passée à la valeur 1.

Ce principe est identique pour toutes les autres règles à modifier !

2.14 Autres exemples de *scripts*

2.14.1 Lister les codecs installés sur un pc

list_codecs.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_CodecFile" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "AccessMask: " $objItem.AccessMask
      write-host "Archive: " $objItem.Archive
      write-host "Caption: " $objItem.Caption
      write-host "Compressed: " $objItem.Compressed
      write-host "Compression Method: " $objItem.CompressionMethod
      write-host "Creation Class Name: " $objItem.CreationClassName
      write-host "Creation Date: " $objItem.CreationDate
      write-host "CS Creation Class Name: " $objItem.CSCreationClassName
      write-host "CS Name: " $objItem.CSName
      write-host "Description: " $objItem.Description
      write-host "Drive: " $objItem.Drive
      write-host "8.3 File Name: " $objItem.EightDotThreeFileName
      write-host "Encrypted: " $objItem.Encrypted
      write-host "Encryption Method: " $objItem.EncryptionMethod
      write-host "Extension: " $objItem.Extension
      write-host "File Name: " $objItem.FileName
      write-host "File Size: " $objItem.FileSize
      write-host "File Type: " $objItem.FileType
      write-host "FS Creation Class Name: " $objItem.FSCreationClassName
      write-host "FS Name: " $objItem.FSName
      write-host "Group: " $objItem.Group
      write-host "Hidden: " $objItem.Hidden
      write-host "Installation Date: " $objItem.InstallDate
      write-host "In Use Count: " $objItem.InUseCount
      write-host "Last Accessed: " $objItem.LastAccessed
      write-host "Last Modified: " $objItem.LastModified
      write-host "Manufacturer: " $objItem.Manufacturer
      write-host "Name: " $objItem.Name
      write-host "Path: " $objItem.Path
      write-host "Readable: " $objItem.Readable
      write-host "Status: " $objItem.Status
      write-host "System: " $objItem.System
      write-host "Version: " $objItem.Version
      write-host "Writeable: " $objItem.Writeable
      write-host
}
```

Nous pouvons constater l'instruction **`foreach`**, qui n'est rien d'autre qu'une boucle parcourant chaque codec installé sur le poste.

Les instructions **`write-host`** affichent les informations dans la fenêtre de *PowerShell*.

Exemple des informations codec affichées (il s'agit ici du codec mp3) :

```
AccessMask:  18809343
Archive:  True
Caption:  c:\windows\system32\l3codeca.acm
Compressed:  False
Compression Method:
Creation Class Name:  Win32_CodecFile
Creation Date:  20020829140000.000000+120
CS Creation Class Name:  Win32_ComputerSystem
CS Name:  U32
Description:  Fraunhofer IIS MPEG Layer-3 Codec
Drive:  c:
8.3 File Name:  c:\windows\system32\l3codeca.acm
Encrypted:  False
Encryption Method:
Extension:  acm
File Name:  l3codeca
File Size:  290816
File Type:  acm File
FS Creation Class Name:  Win32_FileSystem
FS Name:  NTFS
Group:  Audio
Hidden:  False
Installation Date:  20020829140000.000000+120
In Use Count:
Last Accessed:  20061113120125.921875+060
Last Modified:  20040804015612.000000+120
Manufacturer:  Fraunhofer Institut Integrierte Schaltungen IIS
Name:  C:\WINDOWS\system32\L3CODECA.ACM
Path:  \windows\system32\
Readable:  True
Status:  OK
System:  False
Version:  1, 9, 0, 0305
Writeable:  True
```

2.14.2 Lister les commandes de démarrage

list_computer_startup_commands.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_StartupCommand" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Caption: " $objItem.Caption
      write-host "Command: " $objItem.Command
      write-host "Description: " $objItem.Description
      write-host "Location: " $objItem.Location
      write-host "Name: " $objItem.Name
      write-host "Setting ID: " $objItem.SettingID
      write-host "User: " $objItem.User
      write-host
}
```

Exemple de la commande de démarrage de McAfee :

```
Caption:  McAfeeUpdaterUI
Command:  "C:\Program Files\Network Associates\Common Framework\UpdaterUI.exe" /StartedFromRunKey
Description:  McAfeeUpdaterUI
Location:  HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Name:  McAfeeUpdaterUI
Setting ID:
User:  All Users
```

2.14.3 Lister les variables d'environnement

list_environment_variables.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_Environment" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Caption: " $objItem.Caption
      write-host "Description: " $objItem.Description
      write-host "InstallationDate: " $objItem.InstallDate
      write-host "Name: " $objItem.Name
      write-host "Status: " $objItem.Status
      write-host "System Variable: " $objItem.SystemVariable
      write-host "User Name: " $objItem.UserName
      write-host "Variable Value: " $objItem.VariableValue
      write-host
}
```

Exemple d'une variable d'environnement affichée :

```
Caption:  <SYSTEM>\PROCESSOR_ARCHITECTURE
Description:  <SYSTEM>\PROCESSOR_ARCHITECTURE
InstallationDate:
Name:  PROCESSOR_ARCHITECTURE
Status:  OK
System Variable:  True
User Name:  <SYSTEM>
Variable Value:  x86
```

2.14.4 Lister la table d'adresses IP V4

list_ip4_route_table_into.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "V32"

$colItems = get-wmiobject -class "Win32_IP4RouteTable" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Age: " $objItem.Age
      write-host "Caption: " $objItem.Caption
      write-host "Description: " $objItem.Description
      write-host "Destination: " $objItem.Destination
      write-host "Information: " $objItem.Information
      write-host "Installation Date: " $objItem.InstallDate
      write-host "Interface Index: " $objItem.InterfaceIndex
      write-host "Mask: " $objItem.Mask
      write-host "Metric 1: " $objItem.Metric1
      write-host "Metric 2: " $objItem.Metric2
      write-host "Metric 3: " $objItem.Metric3
      write-host "Metric 4: " $objItem.Metric4
      write-host "Metric 5: " $objItem.Metric5
      write-host "Name: " $objItem.Name
      write-host "Next Hop: " $objItem.NextHop
      write-host "Protocol: " $objItem.Protocol
      write-host "Status: " $objItem.Status
      write-host "Type: " $objItem.Type
      write-host
}
```

Exemple affiché :

```
Age:  23871
Caption:  10.1.0.0
Description:  10.1.0.0 - 255.255.0.0 - 10.1.2.32
Destination:  10.1.0.0
Information:  0.0
Installation Date:
Interface Index:  2
Mask:  255.255.0.0
Metric 1:  20
Metric 2:  -1
Metric 3:  -1
Metric 4:  -1
Metric 5:  -1
Name:  10.1.0.0
Next Hop:  10.1.2.32
Protocol:  2
Status:
Type:  3
```

2.14.5 Lister les protocoles réseau

list_network_protocols.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_NetworkProtocol" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Caption: " $objItem.Caption
      write-host "Connectionless Service: " $objItem.ConnectionlessService
      write-host "Description: " $objItem.Description
      write-host "Guarantees Delivery: " $objItem.GuaranteesDelivery
      write-host "Guarantees Sequencing: " $objItem.GuaranteesSequencing
      write-host "Installation Date: " $objItem.InstallDate
      write-host "Maximum Address Size: " $objItem.MaximumAddressSize
      write-host "Maximum Message Size: " $objItem.MaximumMessageSize
      write-host "Message Oriented: " $objItem.MessageOriented
      write-host "Minimum Address Size: " $objItem.MinimumAddressSize
      write-host "Name: " $objItem.Name
      write-host "Pseudo-Stream Oriented: " $objItem.PseudoStreamOriented
      write-host "Status: " $objItem.Status
      write-host "Supports Broadcasting: " $objItem.SupportsBroadcasting
      write-host "Supports Connect Data: " $objItem.SupportsConnectData
      write-host "Supports Disconnect Data: " $objItem.SupportsDisconnectData
      write-host "Supports Encryption: " $objItem.SupportsEncryption
      write-host "Supports Expedited Data: " $objItem.SupportsExpeditedData
      write-host "Supports Fragmentation: " $objItem.SupportsFragmentation
      write-host "Supports Graceful Closing: " $objItem.SupportsGracefulClosing
      write-host "Supports Guaranteed Bandwidth: " $objItem.SupportsGuaranteedBandwidth
      write-host "Supports Multicasting: " $objItem.SupportsMulticasting
      write-host "Supports Quality of Service: " $objItem.SupportsQualityofService
      write-host
}
```

Exemple de protocole affiché (protocole TCP/IP)

```
Caption:  Tcpip
Connectionless Service:  True
Description:  TCP/IP Protocol Driver
Guarantees Delivery:  False
Guarantees Sequencing:  False
Installation Date:  20020829140000.000000+120
Maximum Address Size:  16
Maximum Message Size:  65467
Message Oriented:  True
Minimum Address Size:  16
Name:  MSAFD Tcpip [UDP/IP]
Pseudo-Stream Oriented:  False
Status:  OK
Supports Broadcasting:  True
Supports Connect Data:  False
Supports Disconnect Data:  False
Supports Encryption:  False
Supports Expedited Data:  False
Supports Fragmentation:
Supports Graceful Closing:  False
Supports Guaranteed Bandwidth:  True
Supports Multicasting:  True
Supports Quality of Service:  False
```

2.14.6 Lister les propriétés du système d'exploitation

list_operating_system_properties.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_OperatingSystem" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Boot Device: " $objItem.BootDevice
      write-host "Build Number: " $objItem.BuildNumber
      write-host "Build Type: " $objItem.BuildType
      write-host "Caption: " $objItem.Caption
      write-host "Code Set: " $objItem.CodeSet
      write-host "Country Code: " $objItem.CountryCode
      write-host "Creation Class Name: " $objItem.CreationClassName
      write-host "CS Creation Class Name: " $objItem.CSCreationClassName
      write-host "CSD Version: " $objItem.CSDVersion
      write-host "CS Name: " $objItem.CSName
      write-host "Current Time Zone: " $objItem.CurrentTimeZone
      write-host "Debug: " $objItem.Debug
      write-host "Description: " $objItem.Description
      write-host "Distributed: " $objItem.Distributed
      write-host "Encryption Level: " $objItem.EncryptionLevel
      write-host "Foreground Application Boost: " $objItem.ForegroundApplicationBoost
      write-host "Free Physical Memory: " $objItem.FreePhysicalMemory
      write-host "Free Space In Paging Files: " $objItem.FreeSpaceInPagingFiles
      write-host "Free Virtual Memory: " $objItem.FreeVirtualMemory
      write-host "Installation Date: " $objItem.InstallDate
      write-host "Large System Cache: " $objItem.LargeSystemCache
      write-host "Last Boot-Up Time: " $objItem.LastBootUpTime
      write-host "Local DateTime: " $objItem.LocalDateTime
      write-host "Locale: " $objItem.Locale
      write-host "Manufacturer: " $objItem.Manufacturer
      write-host "Maximum Number Of Processes: " $objItem.MaxNumberOfProcesses
      write-host "Maximum Process Memory Size: " $objItem.MaxProcessMemorySize
      write-host "Name: " $objItem.Name
      write-host "Number Of Licensed Users: " $objItem.NumberOfLicensedUsers
      write-host "Number Of Processes: " $objItem.NumberOfProcesses
      write-host "Number Of Users: " $objItem.NumberOfUsers
      write-host "Organization: " $objItem.Organization
      write-host "Operating System Language: " $objItem.OSLanguage
      write-host "Operating System Product Suite: " $objItem.OSProductSuite
      write-host "Operating System Type: " $objItem.OSType
      write-host "Other Type Description: " $objItem.OtherTypeDescription
      write-host "Plus Product ID: " $objItem.PlusProductID
      write-host "Plus Version Number: " $objItem.PlusVersionNumber
      write-host "Primary: " $objItem.Primary
      write-host "Product Type: " $objItem.ProductType
      write-host "Quantum Length: " $objItem.QuantumLength
      write-host "Quantum Type: " $objItem.QuantumType
      write-host "Registered User: " $objItem.RegisteredUser
      write-host "Serial Number: " $objItem.SerialNumber
      write-host "Service Pack Major Version: " $objItem.ServicePackMajorVersion
      write-host "Service Pack Minor Version: " $objItem.ServicePackMinorVersion
      write-host "Size Stored In Paging Files: " $objItem.SizeStoredInPagingFiles
      write-host "Status: " $objItem.Status
      write-host "Suite Mask: " $objItem.SuiteMask
      write-host "System Device: " $objItem.SystemDevice
      write-host "System Directory: " $objItem.SystemDirectory
      write-host "System Drive: " $objItem.SystemDrive
      write-host "Total Swap Space Size: " $objItem.TotalSwapSpaceSize
      write-host "Total Virtual Memory Size: " $objItem.TotalVirtualMemorySize
      write-host "Total Visible Memory Size: " $objItem.TotalVisibleMemorySize
      write-host "Version: " $objItem.Version
      write-host "Windows Directory: " $objItem.WindowsDirectory
      write-host
}
```

2.14.7 Lister les processus en cours d'exécution

list_processes_running.ps1 - Notepad

File Edit Format View Help

```
$strComputer = "."

$colItems = get-wmiobject -class "Win32_Process" -namespace "root\CIMV2" `
-computername $strComputer

foreach ($objItem in $colItems) {
      write-host "Caption: " $objItem.Caption
      write-host "Command Line: " $objItem.CommandLine
      write-host "Creation Class Name: " $objItem.CreationClassName
      write-host "Creation Date: " $objItem.CreationDate
      write-host "CS Creation Class Name: " $objItem.CSCreationClassName
      write-host "CS Name: " $objItem.CSName
      write-host "Description: " $objItem.Description
      write-host "Executable Path: " $objItem.ExecutablePath
      write-host "Execution State: " $objItem.ExecutionState
      write-host "Handle: " $objItem.Handle
      write-host "Handle Count: " $objItem.HandleCount
      write-host "Installation Date: " $objItem.InstallDate
      write-host "Kernel-Mode Time: " $objItem.KernelModeTime
      write-host "Maximum Working Set Size: " $objItem.MaximumWorkingSetSize
      write-host "Minimum Working Set Size: " $objItem.MinimumWorkingSetSize
      write-host "Name: " $objItem.Name
      write-host "Operating System Creation Class Name: " $objItem.OSCreationClassName
      write-host "Operating System Name: " $objItem.OSName
      write-host "Other Operation Count: " $objItem.OtherOperationCount
      write-host "Other Transfer Count: " $objItem.OtherTransferCount
      write-host "Page Faults: " $objItem.PageFaults
      write-host "Page File Usage: " $objItem.PageFileUsage
      write-host "Parent Process ID: " $objItem.ParentProcessId
      write-host "Peak Page File Usage: " $objItem.PeakPageFileUsage
      write-host "Peak Virtual Size: " $objItem.PeakVirtualSize
      write-host "Peak Working Set Size: " $objItem.PeakWorkingSetSize
      write-host "Priority: " $objItem.Priority
      write-host "Private Page Count: " $objItem.PrivatePageCount
      write-host "Process ID: " $objItem.ProcessId
      write-host "Quota Non-Paged Pool Usage: " $objItem.QuotaNonPagedPoolUsage
      write-host "Quota Paged Pool Usage: " $objItem.QuotaPagedPoolUsage
      write-host "Quota Peak Non-Paged Pool Usage: " $objItem.QuotaPeakNonPagedPoolUsage
      write-host "Quota Peak Paged Pool Usage: " $objItem.QuotaPeakPagedPoolUsage
      write-host "Read Operation Count: " $objItem.ReadOperationCount
      write-host "Read Transfer Count: " $objItem.ReadTransferCount
      write-host "Session ID: " $objItem.SessionId
      write-host "Status: " $objItem.Status
      write-host "Termination Date: " $objItem.TerminationDate
      write-host "Thread Count: " $objItem.ThreadCount
      write-host "User-Mode Time: " $objItem.UserModeTime
      write-host "Virtual Size: " $objItem.VirtualSize
      write-host "Windows Version: " $objItem.WindowsVersion
      write-host "Working Set Size: " $objItem.WorkingSetSize
      write-host "Write Operation Count: " $objItem.WriteOperationCount
      write-host "Write Transfer Count: " $objItem.WriteTransferCount
      write-host
}
```

2.15 Liens utiles

Des exemples de *scripts*, surtout pour effectuer des lectures d'information, telles que les informations réseau (adresses IP etc.), récupération du *Security Descriptor* d'un objet (site en anglais)
http://www.microsoft.com/technet/scriptcenter/topics/msh/cmdlets/index.mspx

Un *webcast* (vidéo) de *DFO Show*, expliquant les bases de *PowerShell*, très utile pour commencer à se familiariser avec cet outil (en anglais)
http://channel9.msdn.com/Showpost.aspx?postid=201005

Un test de *PowerShell* (effectué part *arstechnica*), montre une bonne partie des possibilités que ce nouveau *shell* offre, avec des exemples (en anglais)
http://arstechnica.com/guides/other/msh.ars/1

Une introduction à *PowerShell*, écrite par Laurent Dardenne (en français)
http://laurent-dardenne.developpez.com/articles/Windows/PowerShell/Introduction/

Documentation Pack de *PowerShell* (cette documentation est installée automatiquement en même temps que *PowerShell*, mais il existe aussi le lien suivant pour télécharger à part toute cette documentation). Documentation très utile pour créer des *scripts* :
http://www.microsoft.com/downloads/details.aspx?FamilyId=B4720B00-9A66-430F-BD56-EC48BFCA154F&displaylang=en

Il existe un *newsgroup* (groupe de discutions) pour *PowerShell*, nommé microsoft.public.windows.powershell, on peut y trouver des questions intéressantes et poser nos propres questions. Des représentants de *Microsoft* répondent parfois à nos questions, bien que leur présence soit relativement limitée…
Pour s'inscrire aux *newsgroups*, consulter
http://support.microsoft.com/default.aspx?scid=fh;FR;NEWSGROUPOE

3 Gestion des *logs* sous *Vista*

1 semaine d'étude

3.1 Introduction

Dans ce chapitre, je vais étudier la gestion des *logs* dans *Windows Vista*.

Je vais commencer par étudier le nouvel outil *MMC Event Viewer*, disponible dans *Vista*, voir les possibilités que nous offre cet outil pour la gestion des *logs*.

Ensuite je vais montrer comment centraliser les *logs* des différents postes *Windows Vista* disponibles sur le réseau, ceci car il est important pour une entreprise de pouvoir centraliser les *logs* afin de les traiter sur un seul et unique poste.

3.2 Qu'est ce qu'un *log* ?

Un *log* est une trace d'un certain évènement qui c'est produit par le passé.

Ce n'est rien d'autre que du texte, indiquant par exemple qu'un accès à été autorisé.

Cette trace comprend divers champs, par exemple l'heure où l'accès a été autorisé, la personne qui a effectué la demande d'accès, le degré d'importance du *log* qui dans ce cas ce sera probablement un *log* d'information (si l'accès se verra refusé, le degré d'importance du *log* sera probablement un *Warning* !), etc.

Tous les divers *logs* sont répertoriés dans un fichier, que l'on appelle fichier journalier. Ces fichiers au format brut sont très difficiles à lire, on perd facilement énormément de temps à chercher des *logs* spécifiques, c'est pourquoi il nous est indispensable de disposer d'outils permettant une lecture beaucoup plus claire et une analyse bien plus rapide et puissante.

3.3 *MMC Event Viewer*

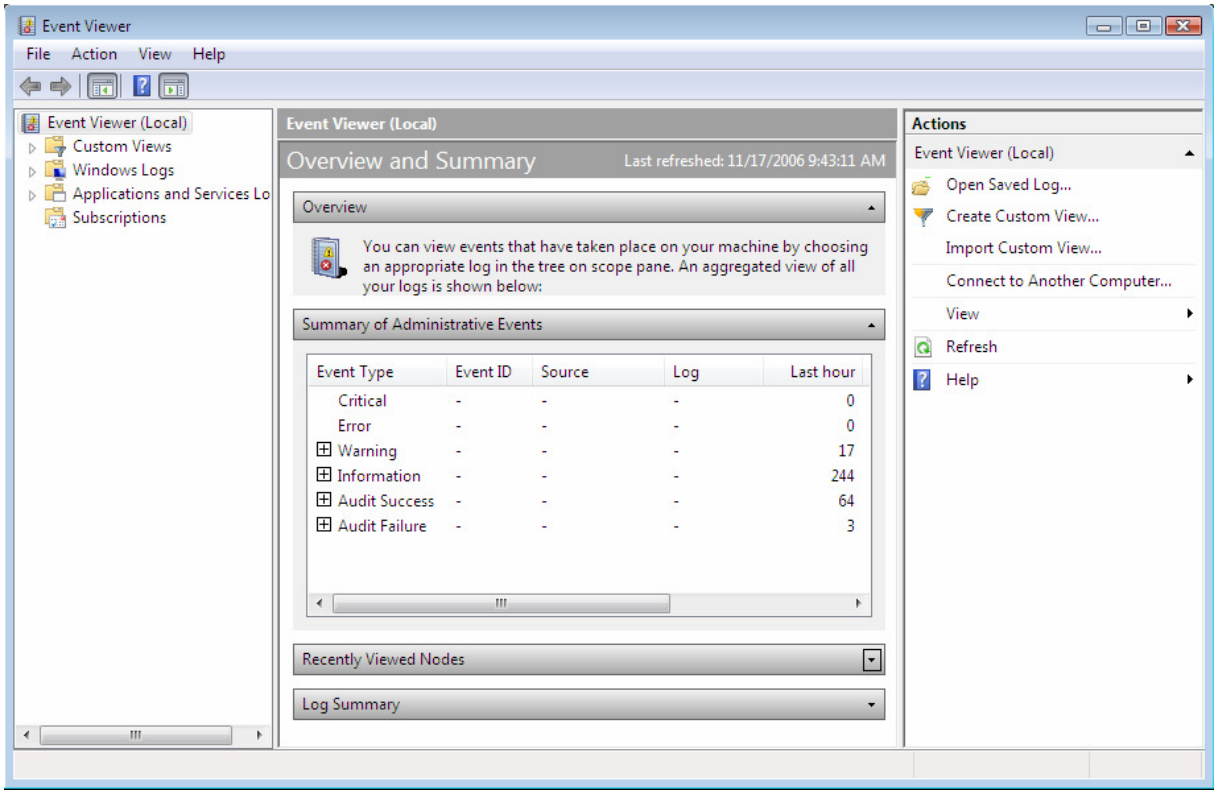
3.3.1 Qu'est ce que c'est ?

MMC (*Microsoft Management Console*) *Event Viewer* (observateur d'évènements), est un outil graphique (GUI) permettant de consulter les différents *logs* produits par l'ordinateur en question.

Il est aussi possible de consulter les *logs* d'autres ordinateurs, je reviendrai sur ce point plus tard dans ce mémoire.

3.3.2 A quoi ressemble-t-il ? Comment lancer son exécution ?

MMC Event Viewer peut être lancé à partir des *Administrative Tools* (*Start – Control Panel – Administrative Tools – Event Viewer*).



Lors du lancement de l'exécution de *MMC Event Viewer*, cet outil va automatiquement rechercher les *logs* présents sur le pc local et en affiche un résumé dans la fenêtre *Overview And Summary*, sous l'onglet *Summary Of Administrative Events*.

Ceci nous permet de voir immédiatement si nous avons des *logs* critiques, des erreurs, des avertissements (*Warnings*) etc.

Rien que ce mécanisme est déjà beaucoup plus parlant que de devoir parcourir un fichier brut de *logs* !

3.3.3 Que permet-il de faire ?

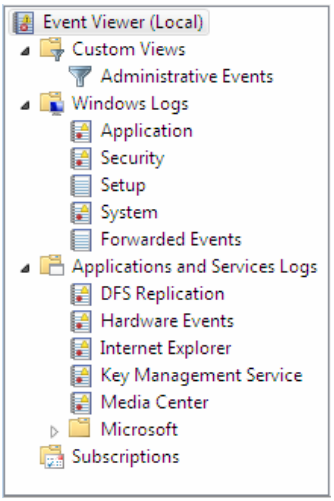
Avec *MMC Event Viewer*, il est possible de :

- Consulter les *logs* (en local) du poste *Vista*.
- Se connecter à un poste distant sur le réseau et consulter les *logs* de ce poste (poste *Vista* ou même antérieur ! voir page 145 pour plus d'informations).
- Filtrer les *logs* pour retrouver plus rapidement les informations recherchées.
- Associer une tâche à un événement spécifique, à l'aide de *Task Scheduler*.
- Centraliser les *logs* sur un poste *Vista*.

Tout ceci a été testé dans ce chapitre avec *Windows Vista Ultimate RC2 Build 5744*

3.3.4 Les *logs* dans *Event Viewer*

Commençons par analyser la fenêtre de gauche nommée *Event Viewer*.



Nous remarquons 3 principaux onglets :

- ***Custom Views*** nous permet de créer nos propres filtres, par défaut nous avons déjà un filtre nommé *Administrative Events.*
  Pour créer un filtre, consulter le paragraphe 3.3.8 page 139

- ***Windows Logs*** regroupe différentes catégories de *logs* :

  - ***Application*** *logs*, qui contient les *logs* produits par des application ou programmes.
    Par exemple le *log* d'une application qui a essayé de parcourir un fichier erroné.

  - ***Security*** *logs*, qui regroupe tous les *logs* liés à la sécurité.
    Par exemple une authentification réussie (ou échouée), la demande d'écriture non autorisée sur un fichier, ou encore la tentative de suppression non autorisée d'un fichier.

  - ***Setup*** *logs*, qui contient les *logs* relatifs au *setup* des applications.

  - ***System*** *logs*, qui regroupe les *logs* des différents composants du système, comme par exemple l'échec d'un *driver* (pilote) ou encore la défaillance d'un autre composants système pendant son démarrage.

  - ***Forwarded Events*** *logs*, qui est utilisé pour stocker les différents *logs* collectés sur des postes distants.

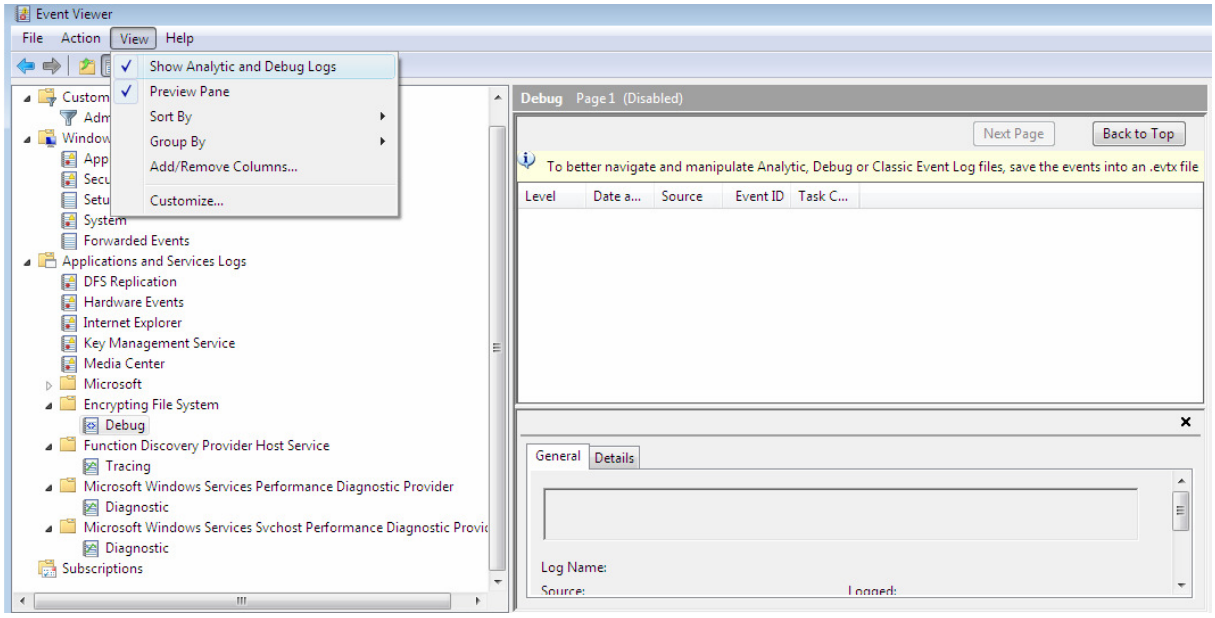
  Remarque : *Application*, *Security* et *System Logs* existent aussi dans les versions antérieures à *Windows Vista*.
  Setup et *Forwarded Events logs* sont quant à eux nouveaux par rapport aux anciennes versions de *Windows*.

- ***Application and Services Logs***, qui est une nouvelle catégorie de *logs*. Cette nouvelle catégorie enregistre les événements d'une application ou d'un composant unique plutôt que d'évènements qui pourraient avoir un impact au niveau système.

  Cette nouvelle catégorie est surtout destinée à des fins de dépannage, elle peut être utilisée aussi bien par des administrateurs que par des développeurs.

  Les administrateurs peuvent consulter par exemple consulter les *Hardware Events* dans le but d'effectuer un dépannage matériel, et les développeurs peuvent quand à eux consulter les *logs* de leur propre application.

  Il faut savoir que les *logs* de cette nouvelle catégorie sont souvent très nombreux et les problèmes peuvent être difficiles à identifier.



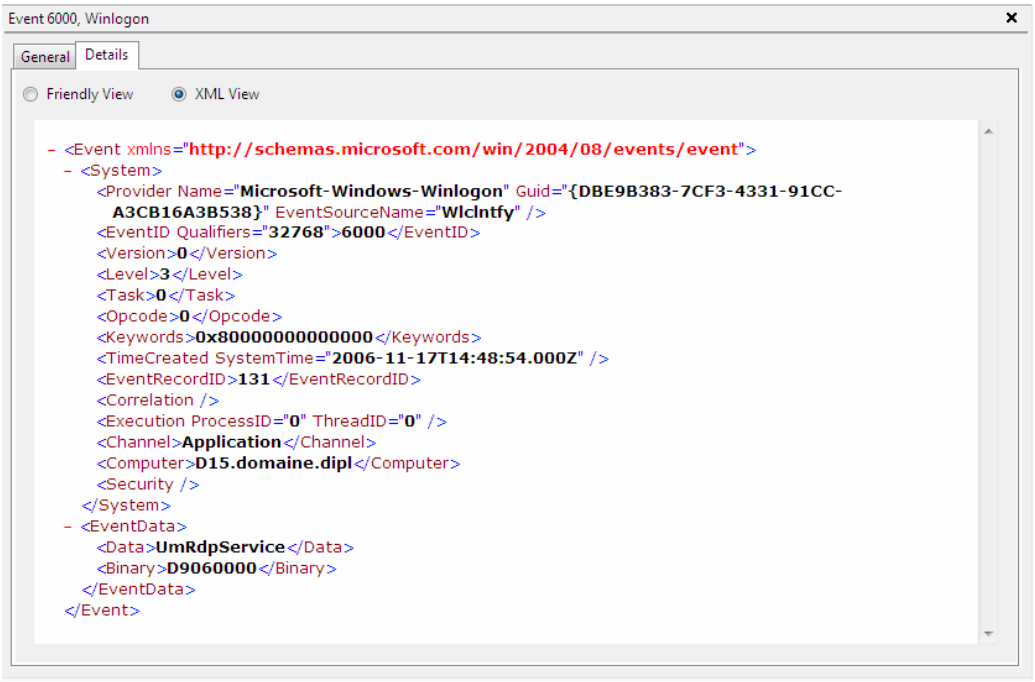
Il existe 2 sous-catégories de *logs* qui ne sont pas visibles par défaut : *Analytic Logs* et *Debug Logs*.

Analytic Logs sont des *logs* décrivant le fonctionnement d'un programme. Ces *logs* sont souvent très nombreux, ils évoquent les problèmes qui ne peuvent pas être résolus par l'utilisateur du programme en question.

Afin de rendre visible ces *logs*, il suffit d'aller dans le menu *View* puis de cocher *Show Analytic and Debug Logs*, ceci nous permet alors de consulter : *Encrypting File System* (pour faire un débogage), *Function Discovery Provider Host Service* (pour suivre les différents évènements), *Microsoft Windows Services Performance Diagnostic Provider* (pour effectuer un diagnostic) et *Microsoft Windows Services Svchost Performance Diagnostic* Provider (pour effectuer aussi un diagnostic).

Je ne vais pas m'étendre plus sur ces sous-catégories, il faut juste savoir qu'elles existent si un jour nous avons besoin de les utiliser.

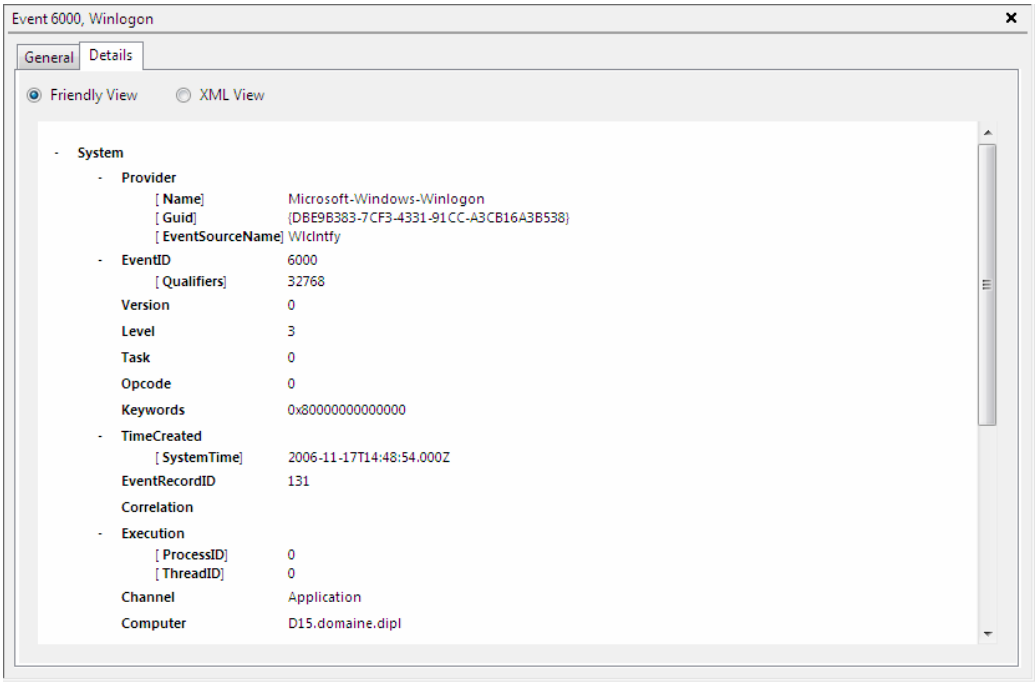
Une nouveauté par rapport aux anciennes versions d'*Event Viewer*, est la représentation des *logs* en format XML.



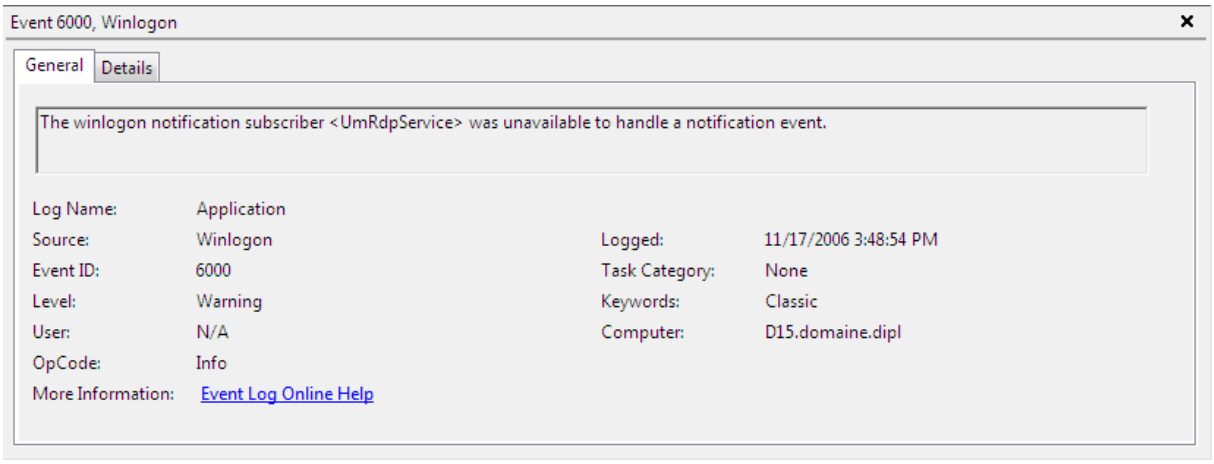
Il suffit pour cela de sélectionner un *log*, puis dans la fenêtre du bas d'*Event Viewer*, sélectionner l'onglet *Détails* puis *XML View*.

Il est possible pour les programmeurs de créer des programmes ou des *scripts* traitant les *logs* de cette manière, avec le format XML.

Il est aussi possible de les représenter sur une vue appelée *Friendly View*, qui est une vue supposée être plus lisible que la vue XML :



Il existe de plus une vue générale, qui est pour ma part la vue la plus claire et lisible en consultant un *log*. Il y a de plus une phrase de description du *log* en question, qui permet de savoir immédiatement ce que le *log* signifie.



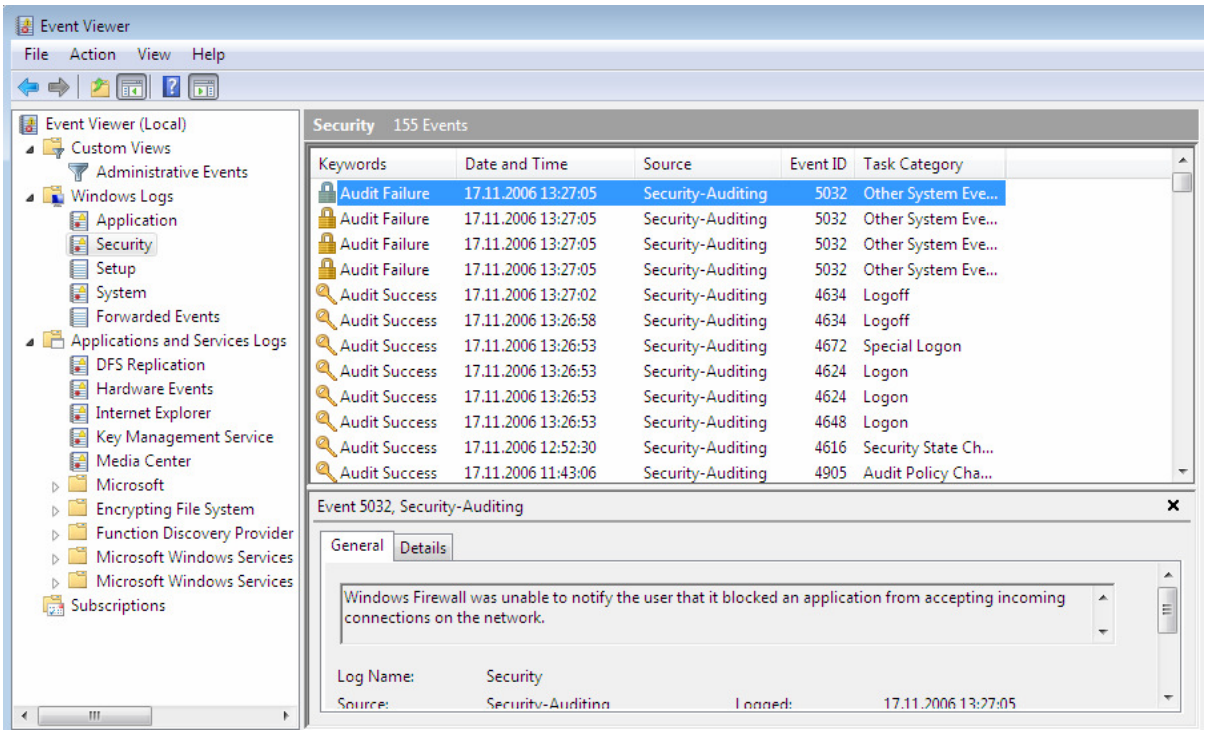
Nous pouvons remarquer aussi une nouveauté, qui est le lien nommé **`More Information`**. Ce lien nous permet d'accéder directement à une page web contenant de plus amples explications sur le *log*. Actuellement il n'y a pas beaucoup d'informations disponibles en consultant ces liens, espérons que ceci soit amélioré dans le futur !

3.3.5 Quels sont les champs des *logs* ?

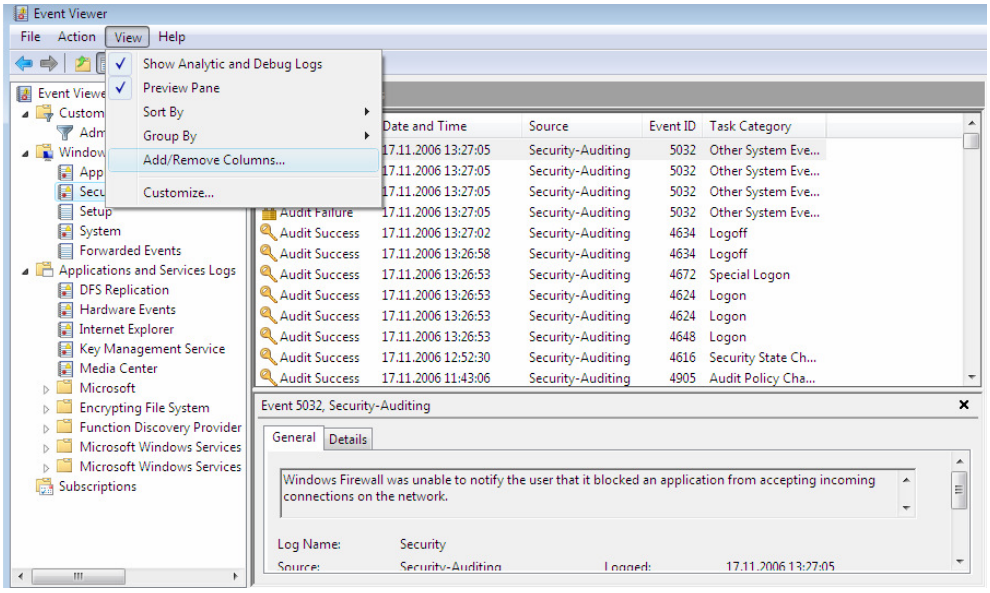
Chaque log dispose de divers champs qui lui sont associés, comme déjà cité précédemment.

Nous allons ici voir quels sont les champs associés aux *logs* disponibles dans *Event Viewer*.

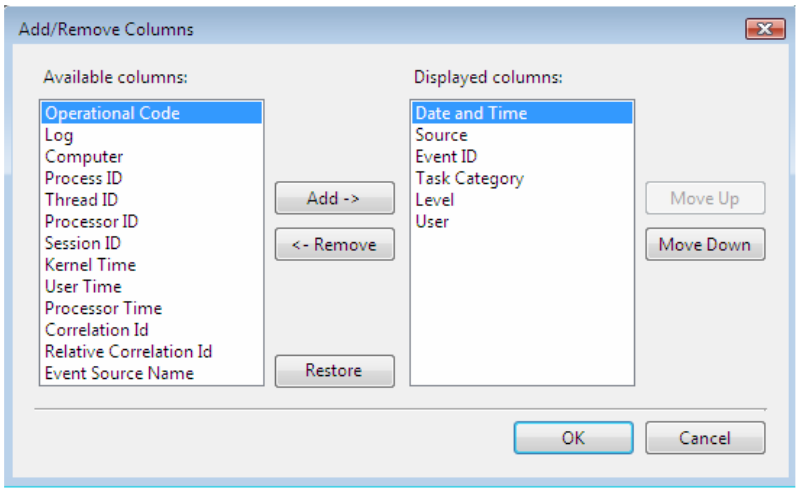
En sélectionnant par exemple les *logs* contenus dans *Security*, les champs affichés par défaut dans *Event Viewer* sont les suivants : *Keywords*, *Date and Time*, *Source*, *Event ID*, *Task Category*.



Il est possible d'ajouter d'autres champs, pour cela sélectionner le Menu *View*, puis *Add/Remove Columns…*



Il nous est alors possible d'afficher d'autres champs dans *Event Viewer*



Les champs disponibles sont les suivants :

- ***Operational Code***, qui contient une valeur indiquant l'activité ou un point dans une activité que l'application exécutait lorsque l'événement a eu lieu.
  Par exemple l'initialisation ou la fermeture.

- ***Log***, qui indique le genre de *log* (*Security*, *Application*, etc.)

- ***Computer***, qui indique le nom du poste qui a créé le *log*. Cette information est notamment utile lorsque les divers *logs* de divers postes du réseau sont centralisés sur une seule machine.

- ***Process ID***, qui contient l'identifiant unique du processus

- ***Thread ID***, qui contient l'identifiant unique du *Thread* (tâche)

- ***Processor ID***, qui contient l'identifiant unique du processeur

- ***Session ID***, qui contient l'identifiant unique de la session

- ***Kernel Time***, l'heure au niveau noyau où l'évènement a été produit

- ***User Time***, l'heure au niveau utilisateur où l'événement a été produit

- ***Processor Time***, l'heure au niveau du processeur où l'événement a été produit

- ***Correlation ID***, un identifiant unique de corrélation, aucune information supplémentaire n'a été trouvée sur ce champ, aucun *log* n'avais de valeur dans ce champ.

- ***Relative Correlation ID***, idem que le champ précédent, aucune information n'a été trouvée sur ce champ et il était vide pour tous les *logs* rencontrés sauf pour un *log* d'erreur.

- ***Event Source Name***, le nom de la source d'événement (par exemple *Winlogon* qui gère l'authentification pour entrer dans une session *Windows*, ou encore *Winmgmt* qui est *Windows Management*)

- ***Date and Time***, la date et l'heure où le *log* a été produit.

- ***Source***, qui indique l'application qui a enregistré l'événement. Il peut s'agir d'une application système, comme d'un programme ou encore d'un driver (pilote).

- ***Event ID***, qui contient l'identifiant unique de l'événement.

- ***Task Category***, indique la catégorie de tâche, par exemple un *Logon* (nom d'utilisateur et mot de passe) ou *Logoff* (lorsqu'on se déconnecte par exemple de notre session *Windows*).

- ***Level***, indique l'importance du log. Il y en a au total 5 : *Critical*, *Error*, *Warning*, *Information*, *Verbose*.
  Dans *Event Viewer*, chaque niveau d'importance du *log* est associé à un symbole, par exemple une croix rouge lors d'un log d'erreur, ou encore un triangle avec un point d'exclamation lors d'un log *Warning*.

- ***User***, qui contient le nom d'utilisateur associé au *log*.

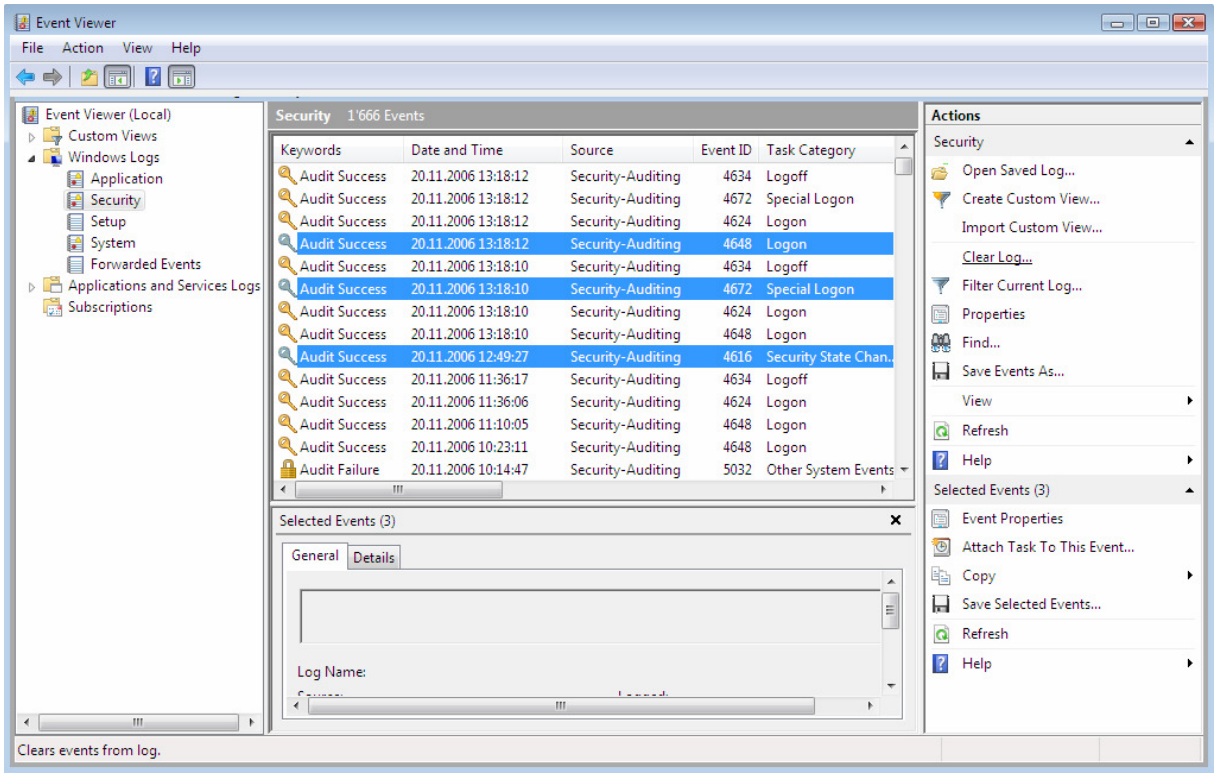
3.3.6 Gérer les *logs*

Dans ce paragraphe nous allons voir comment supprimer des *logs*, comment spécifier la taille maximum d'un fichier de *logs* et comment faire lorsque le fichier de *logs* est complet (est-ce qu'on sauve ce fichier? le supprime ?).

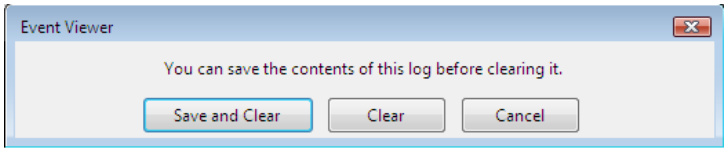
Ces actions peuvent aussi être effectuées à l'aide de lignes de commandes, consulter l'aide d'*Event Viewer* pour plus d'informations.

3.3.6.1 Supprimer des *logs*

Il est relativement facile de supprimer un *log*, il suffit pour cela de sélectionner les *logs* à supprimer, puis de cliquer sur *Clear Log…* (Dans la fenêtre *Actions*)



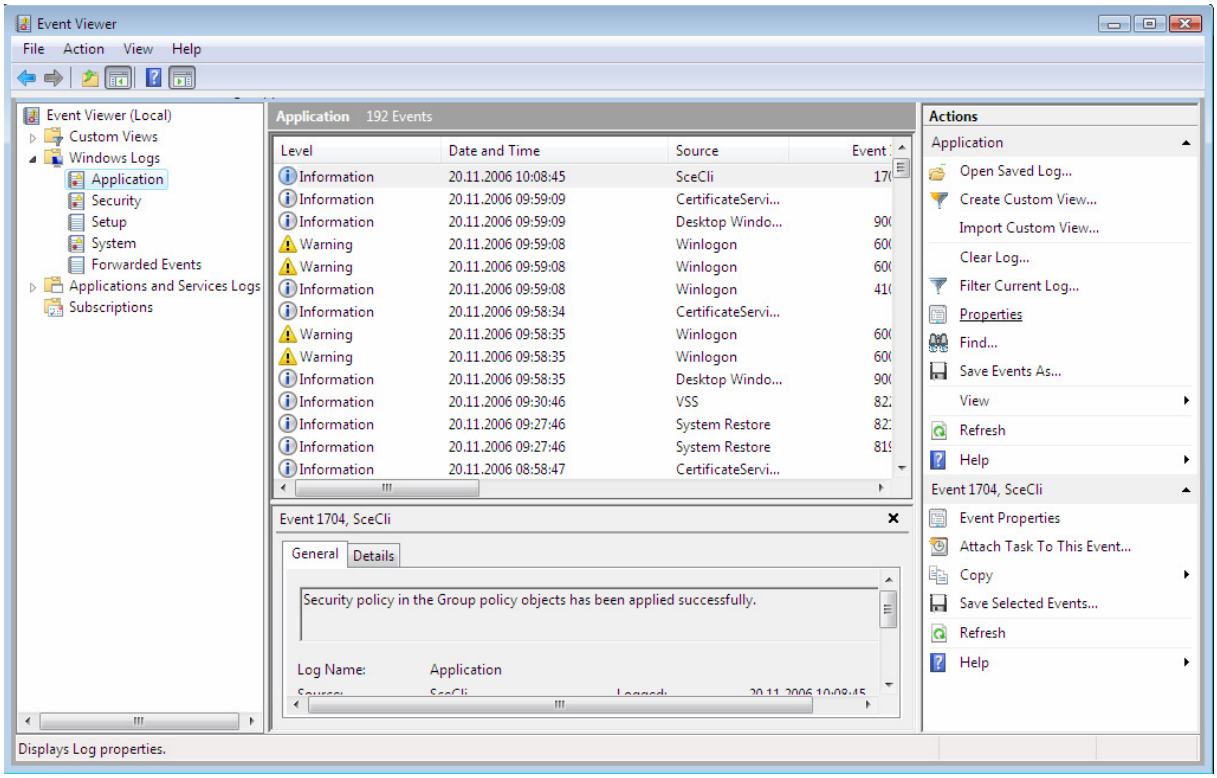
Nous pourrons alors confirmer la suppression en effectuant une copie des *logs* au préalable, ou supprimer définitivement les *logs*.



3.3.6.2 Taille maximum d'un fichier de *logs* et actions sur les fichiers journaliers complets

Il est possible de spécifier la taille maximum d'un fichier de *logs*, pour cela sélectionner la catégorie de *logs* (dans *Windows Logs* par exemple) puis sur le bouton *Properties* disponible dans la fenêtre *Actions*.

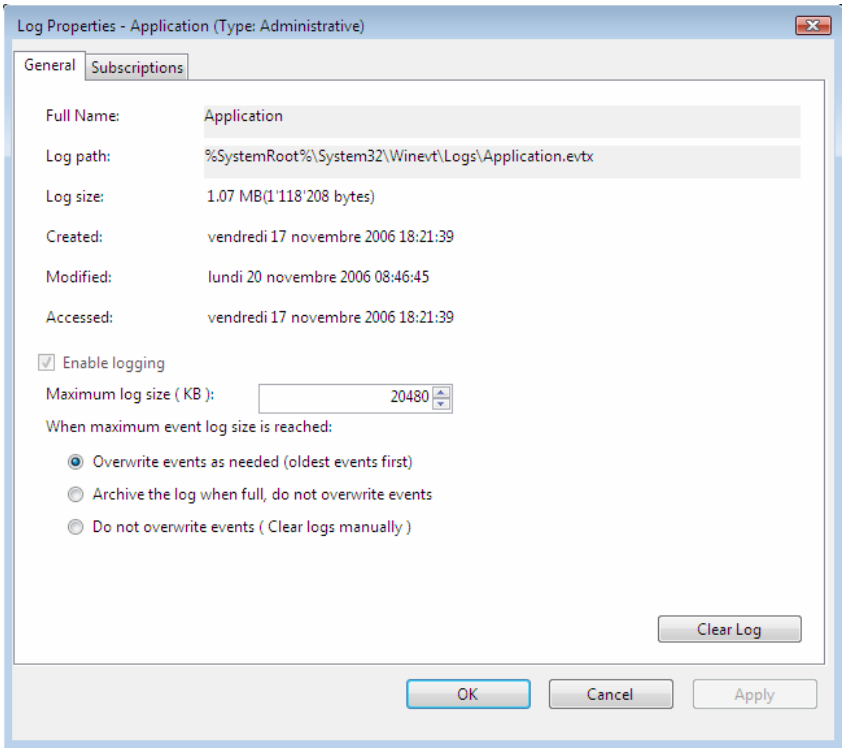
Dans l'exemple suivant la catégorie *Application* a été sélectionnée :



Nous pouvons alors spécifier la taille maximum du fichier de *logs* en question.

Dans cet exemple nous pouvons constater que le fichier de *logs* se trouve dans **`%SystemRoot%\System32\Winevt\Logs\Application.evtx`**

Les *logs* sont donc sauvés dans un fichier au format **evtx**.



Remarque : La taille maximum du fichier de *logs* doit être un multiple de 64 KBytes et ne peut être inférieure à 1024 KBytes.

Lorsque le fichier de *logs* a atteint sa taille maximale, nous pouvons spécifier un des 3 choix suivants :

- *Overwrite events as needed* (*oldest events first*), les *logs* les plus anciens seront remplacés par les nouveaux.

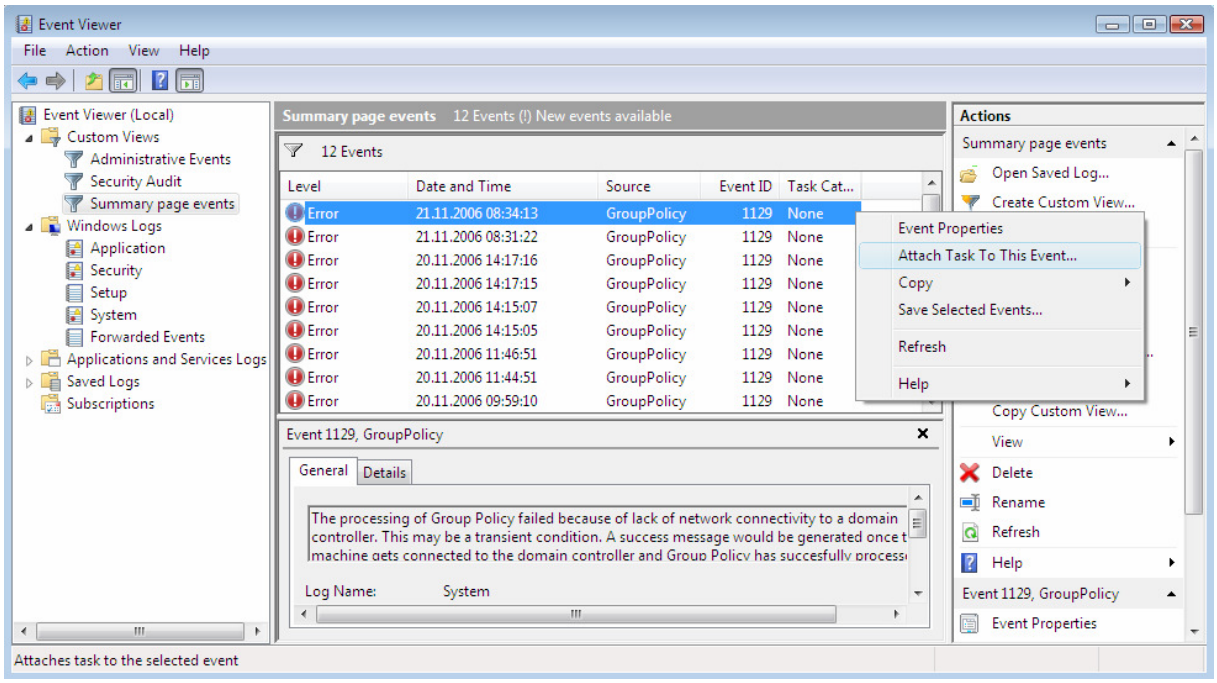
- *Archive the log when full, do not overwrite events*, le fichier de *logs* sera sauvegardé, puis un nouveau fichier sera créé, il n'y aura donc aucun *log* perdu.

- *Do not overwrite events* (*Clear logs manually*), les *logs* ne seront pas remplacés mais il faudra en effacer certains manuellement pour laisser de la place aux suivants.

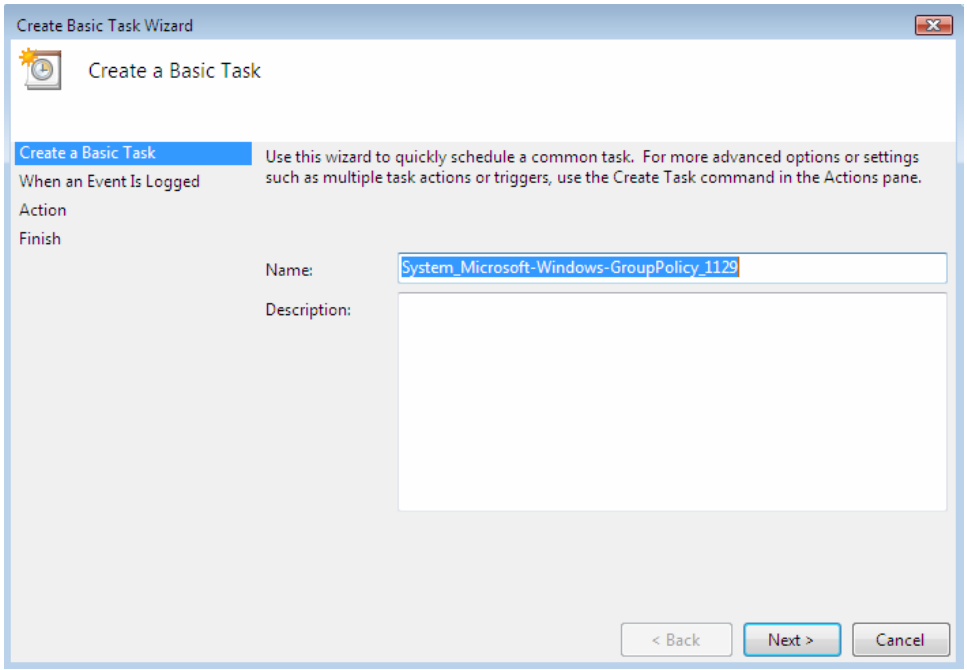
3.3.7 Task Scheduler

Lorsqu'un événement spécifique se produit, il est possible d'associer une action à cet événement à l'aide de *Task Scheduler*.

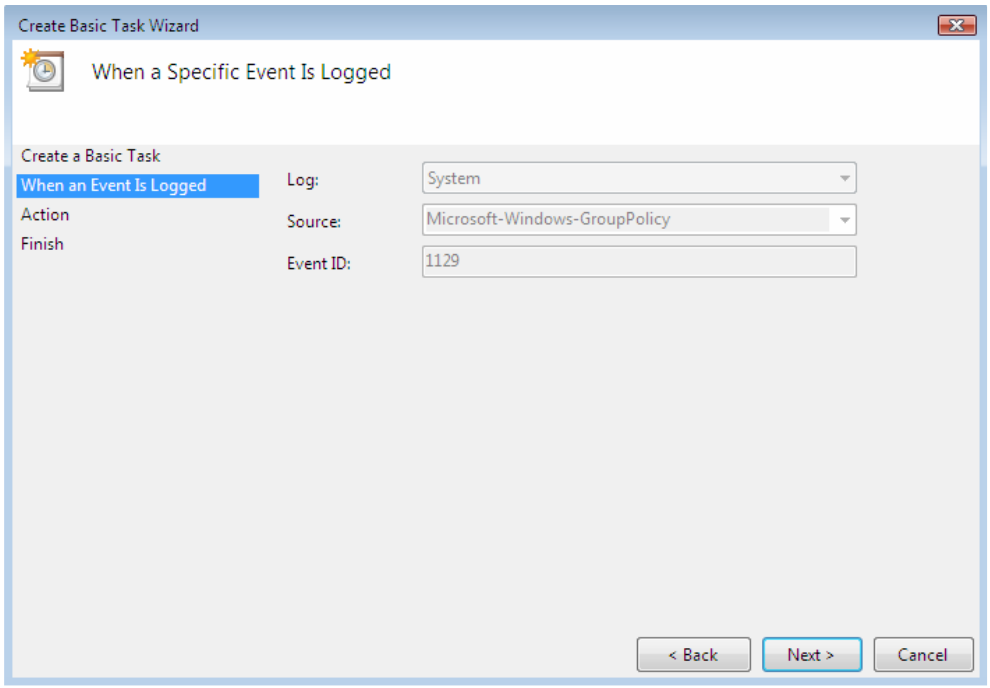
Pour cela, sélectionner le *log*, effectuer un clic droit puis sélectionner *Attach Task To This Event…*



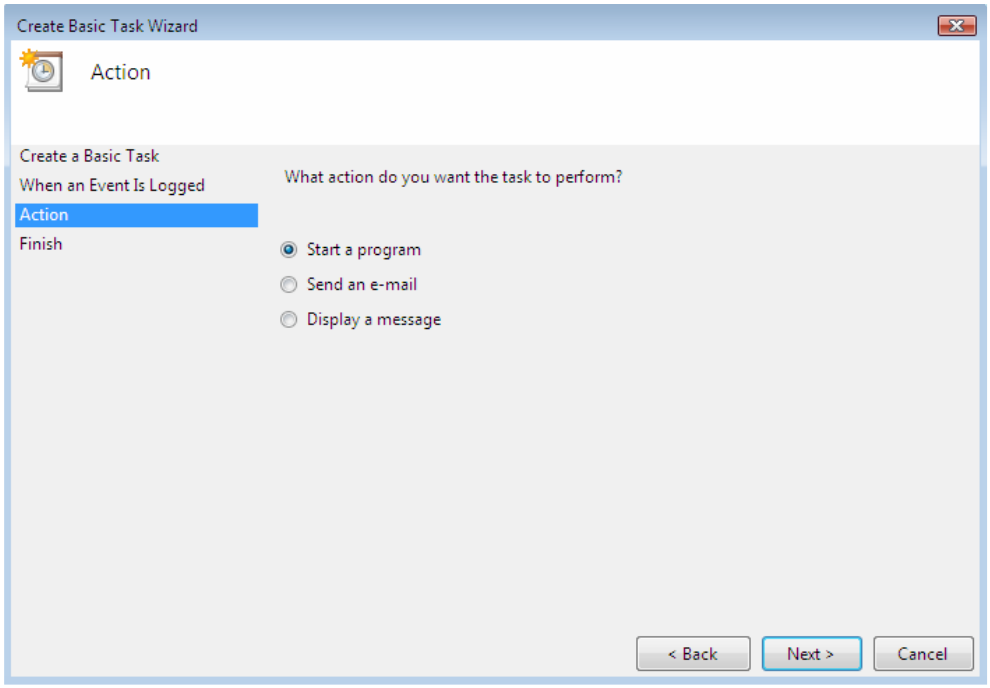
Il est alors possible de spécifier un nom pour la tâche qui va être créée, puis une description :



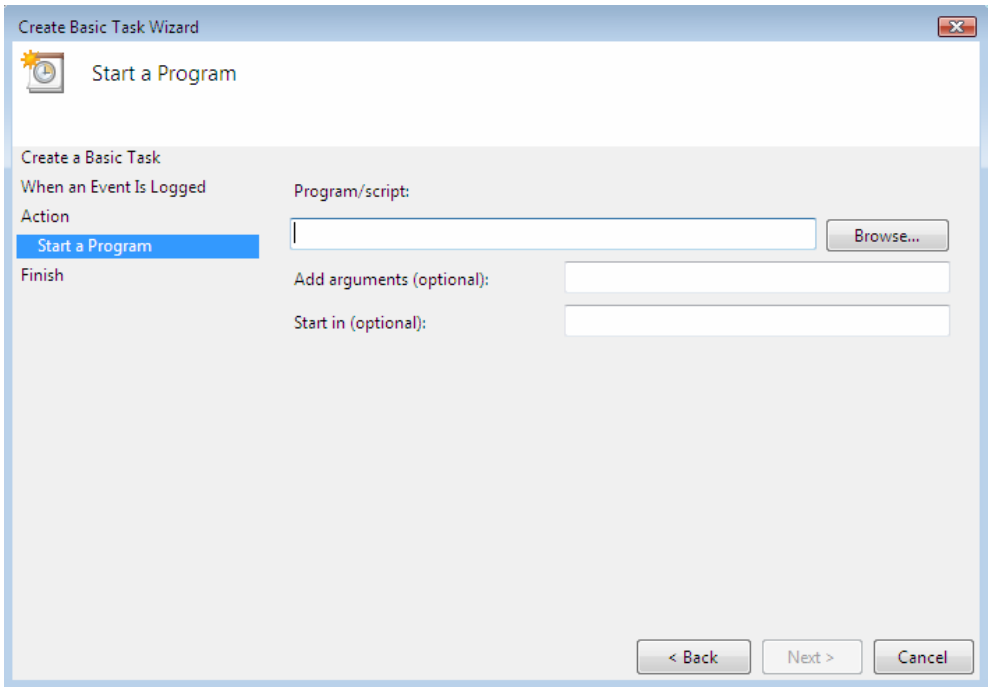
Le type de *log*, son groupe ainsi que son *EventID* sont affichés dans la fenêtre suivante, en sélectionnant un *log* de la manière précédente, ces informations ne peuvent être changées :



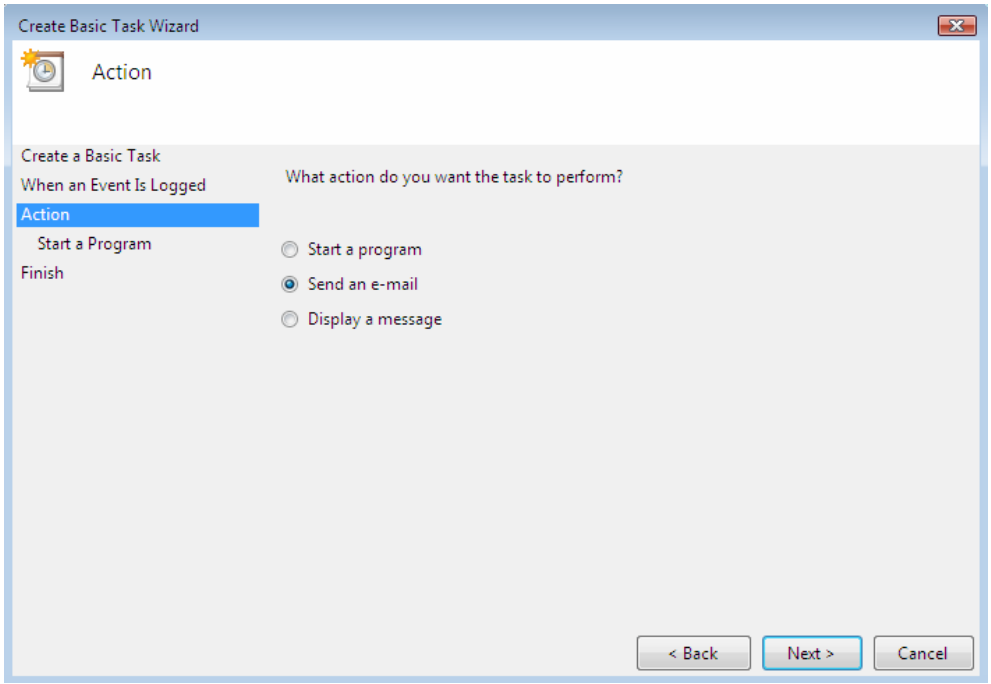
Choisir une des 3 options suivantes, démarrer un programme, envoyer un e-mail ou afficher un message :



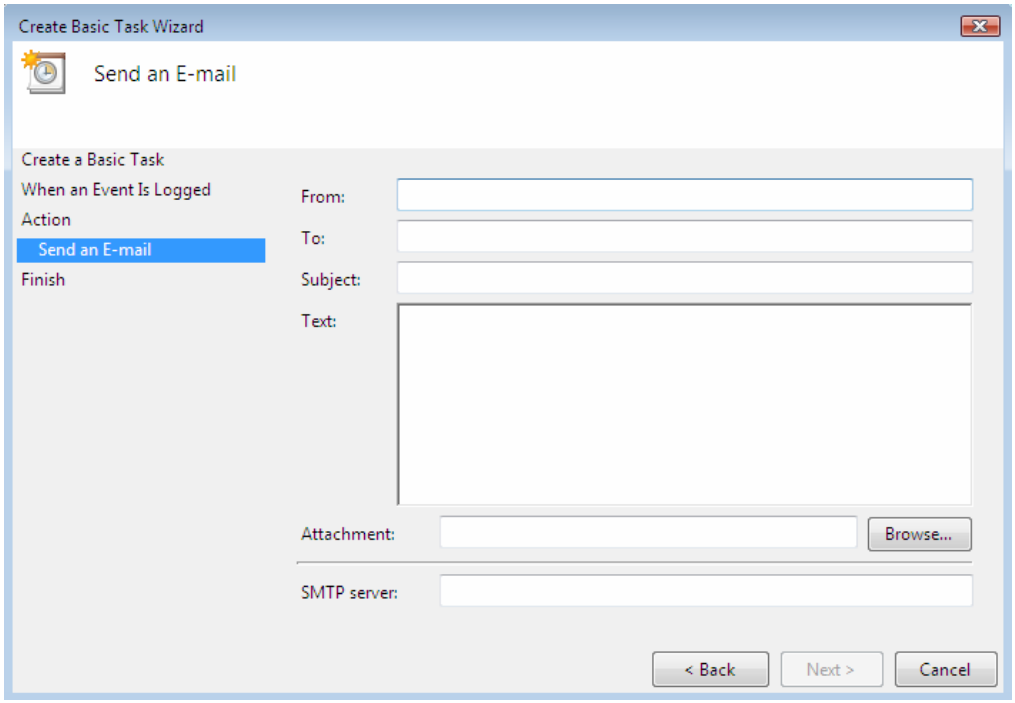
Si l'action *Start a program* est choisie, nous pouvons sélectionner un programme ou un *script* à exécuter (remarquer le bouton *Browse…* qui permet de chercher un programme/*script* dans le système de fichiers), on peut aussi spécifier des arguments pour le programme qui va être lancé :



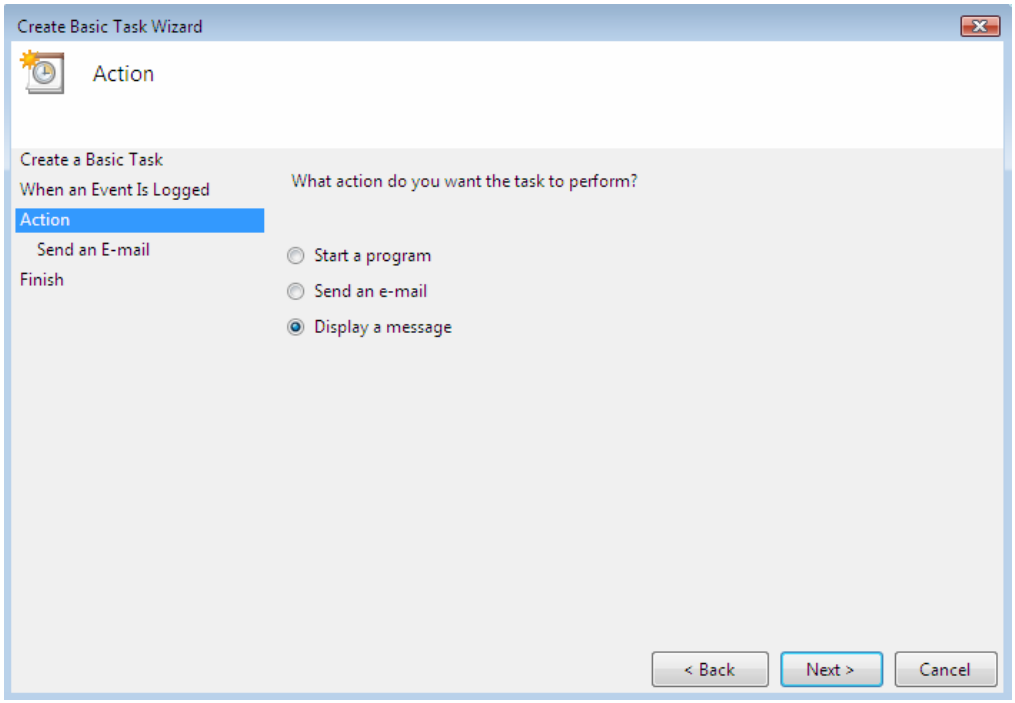
Si l'option *Send an e-mail* est choisie, un e-mail sera envoyé à chaque apparition de ce *log*. Cette option est très pratique pour envoyer un e-mail à l'administrateur en cas d'évènements importants !
Par exemple lorsqu'un ordinateur n'arrive pas à se connecter au contrôleur de domaine, ou lorsque un programme censé être toujours actif se plante.



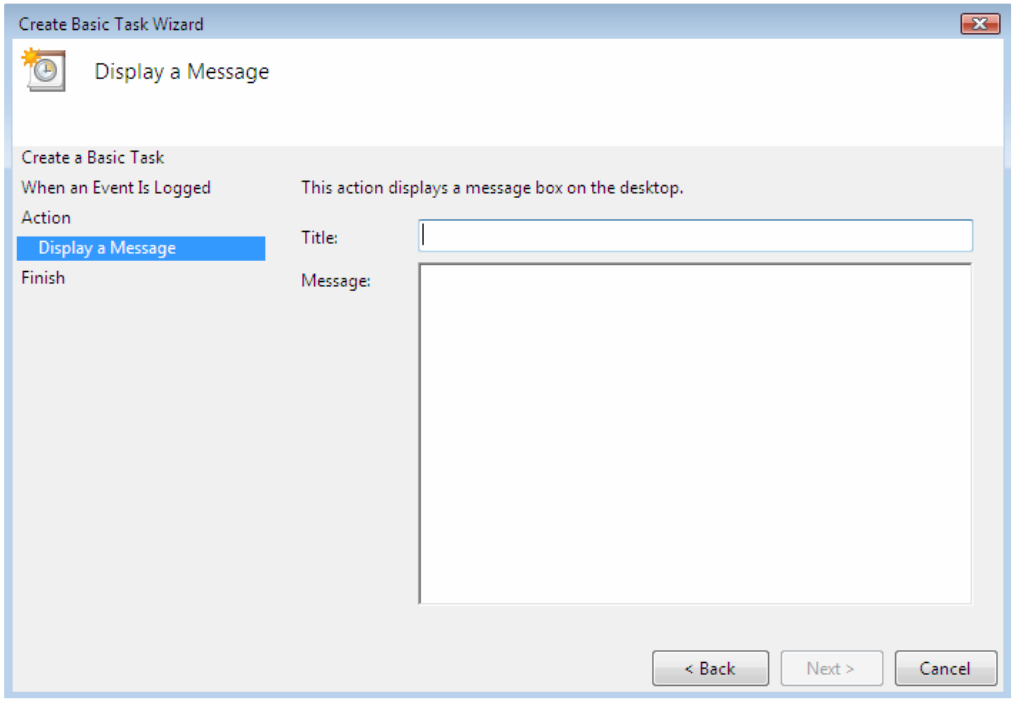
Entrer les informations demandées ci-dessous, il faut disposer d'un serveur SMTP afin que l'e-mail puisse être acheminé :



Si l'option *Display a message* est sélectionnée :



Un simple message peut être affiché à l'écran, pour cela entrer un titre et le message à afficher :



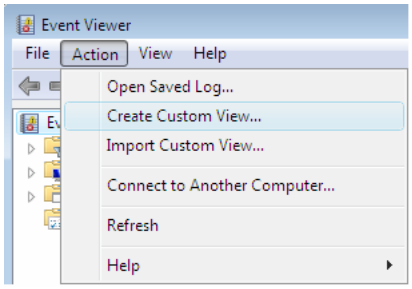
3.3.8 Filtrer les *logs*

Les *logs* affichés sont souvent nombreux, il est important de pouvoir les filtrer afin d'accéder plus rapidement aux *logs* recherchés.

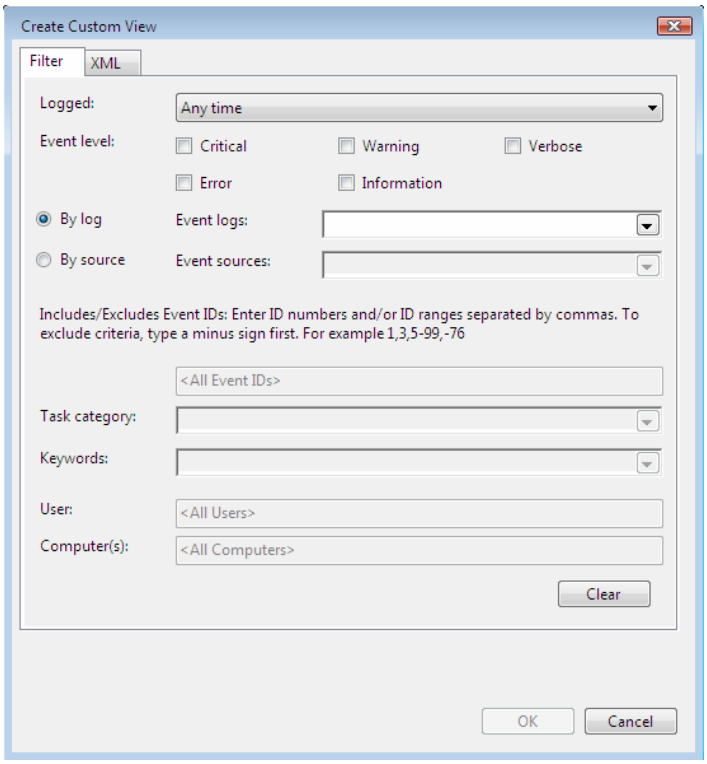
Nous allons voir dans ce paragraphe comment créer des filtres personnalisés, appelés *Custom Views* dans *Event Viewer*.
Nous verrons aussi que ces filtres peuvent être sauvés pour être réutilisés sur d'autres ordinateurs.

3.3.8.1 Créer des filtres personnalisés (*Custom View*)

Sélectionner le menu *Action*, puis *Create Custom View…*

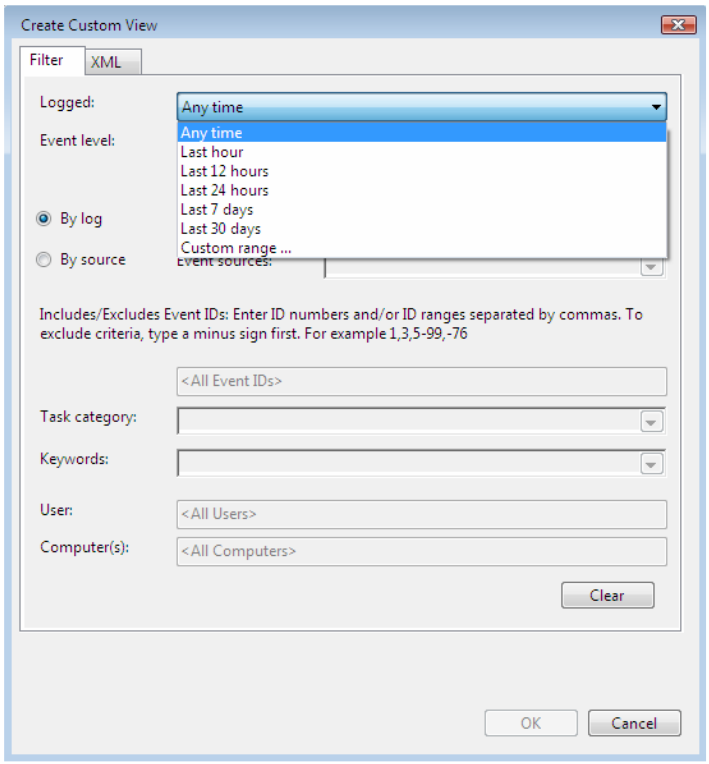


La fenêtre suivante s'affiche, nous pouvons y remarquer plusieurs options.

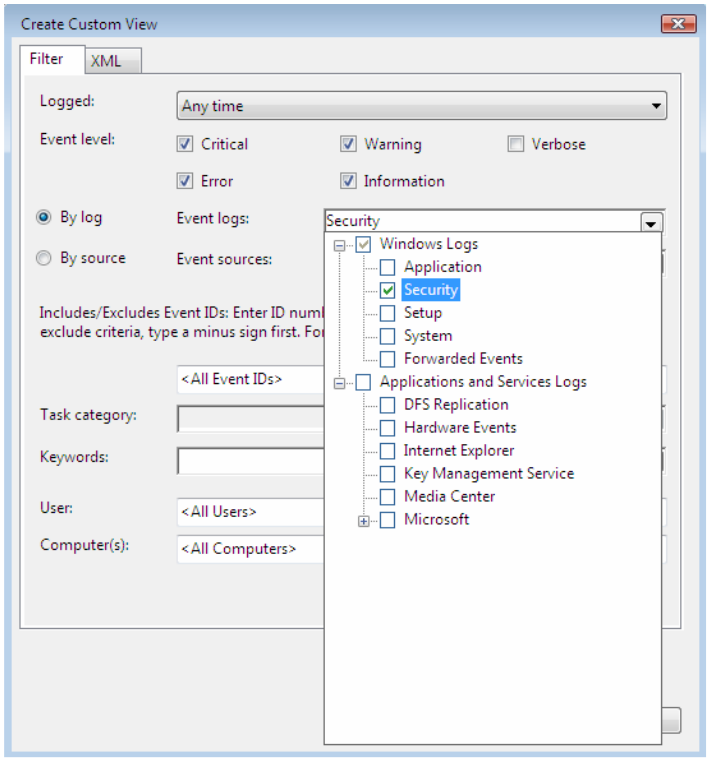


L'option ***Event level*** permet de sélectionner le degré des *logs* que l'on veut consulter (*logs* de niveau critique, *logs* d'erreur, etc.)

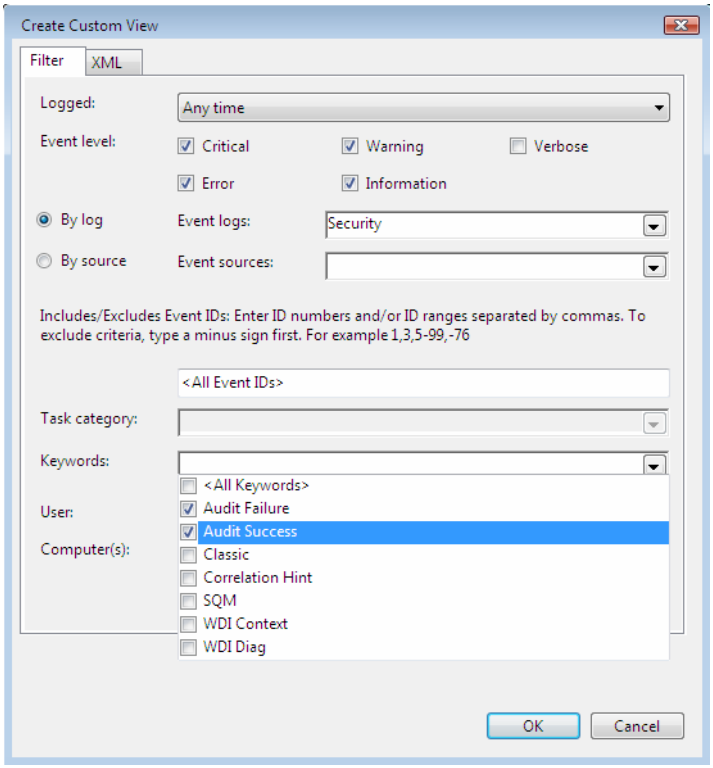
L'option ***Logged*** permet de définir un intervalle de temps pendant laquelle les *logs* seront filtrés. Par exemple en sélectionnant *Last Hour*, seuls les *logs* qui ont été créés durant la dernière heure seront pris en compte.
Il est aussi possible de définir sa propre intervalle de temps avec *Custom range…*



L'option ***By log*** permet de filtrer par rapport à une ou plusieurs catégories de *logs*. Dans l'exemple suivant nous souhaitons uniquement des *logs* de type *Security*.

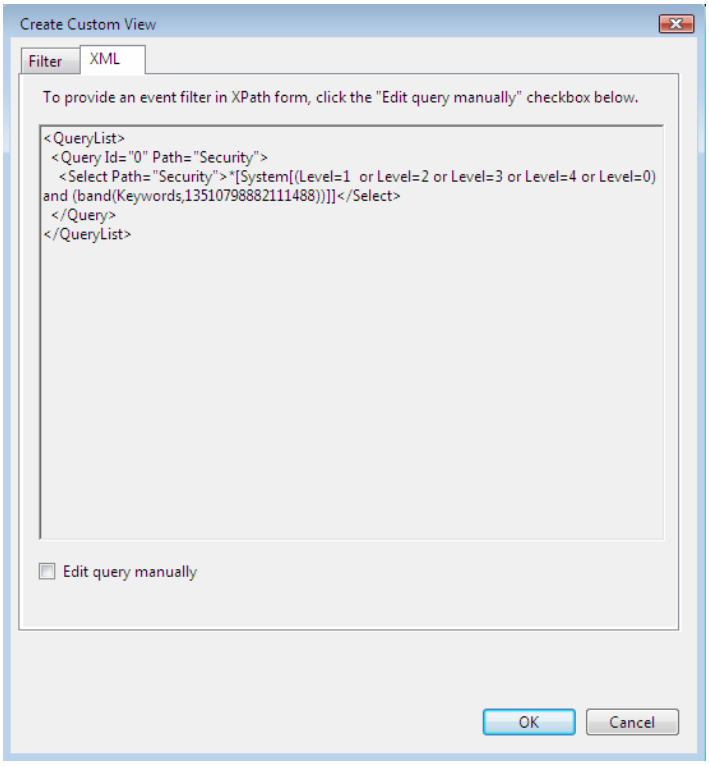


L'option ***Keywords*** permet de spécifier les divers types de *logs* (présents dans les catégories de *logs* choisies avec l'option précédente) que l'on veut filtrer.
Dans cet exemple nous souhaitons uniquement les *logs* de type *Audit Failure* et *Audit Success*.

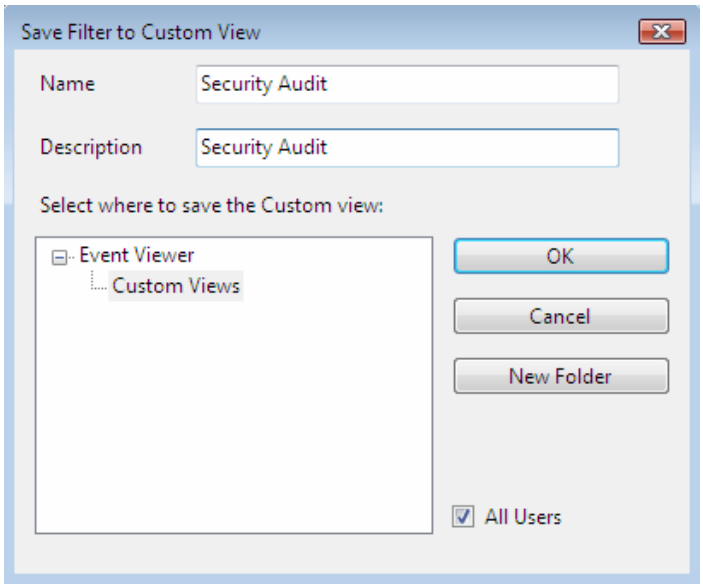


En sélectionnant l'onglet XML, nous pouvons voir la requête XML qui est générée avec les options définies au préalable.

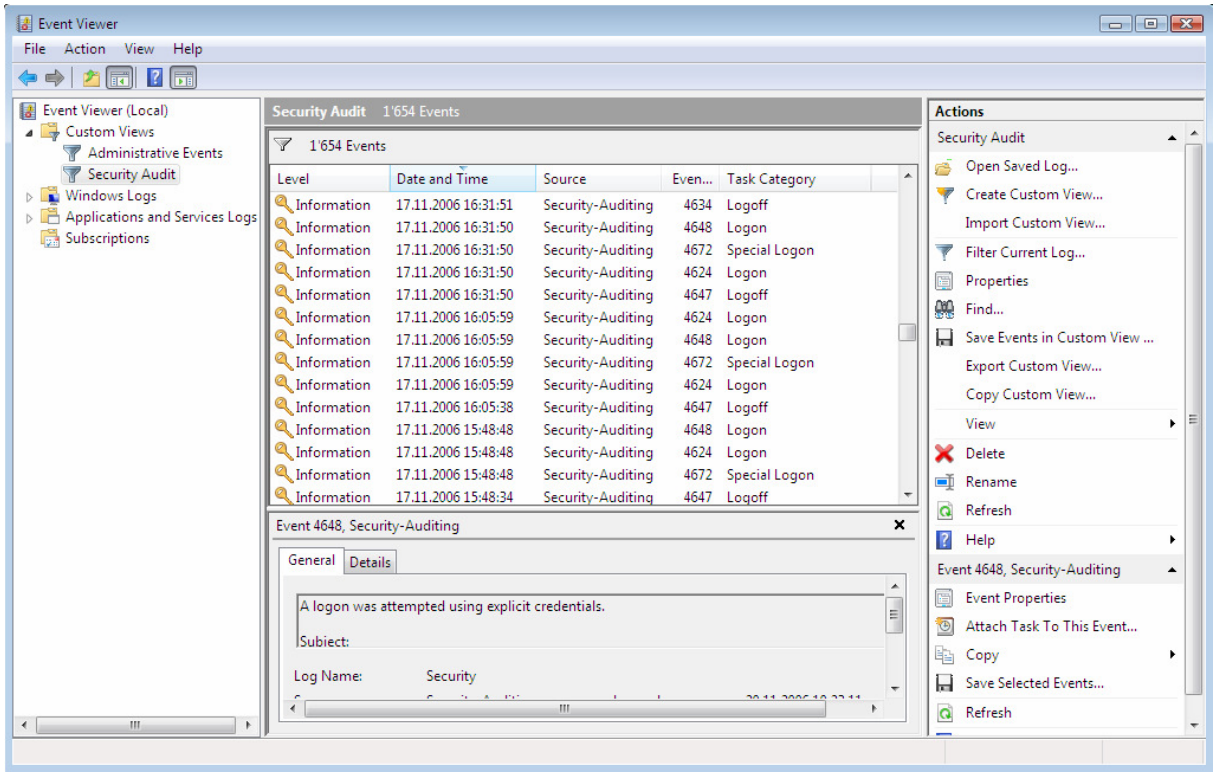
Il est également possible de créer une telle requête à l'aide d'un simple éditeur de texte.



En cliquant sur le bouton *OK*, *Event Viewer* nous demande un nom ainsi qu'une description de notre filtre, et sous quel dossier sauver notre filtre (il est possible de créer plusieurs dossiers contenant plusieurs filtres).



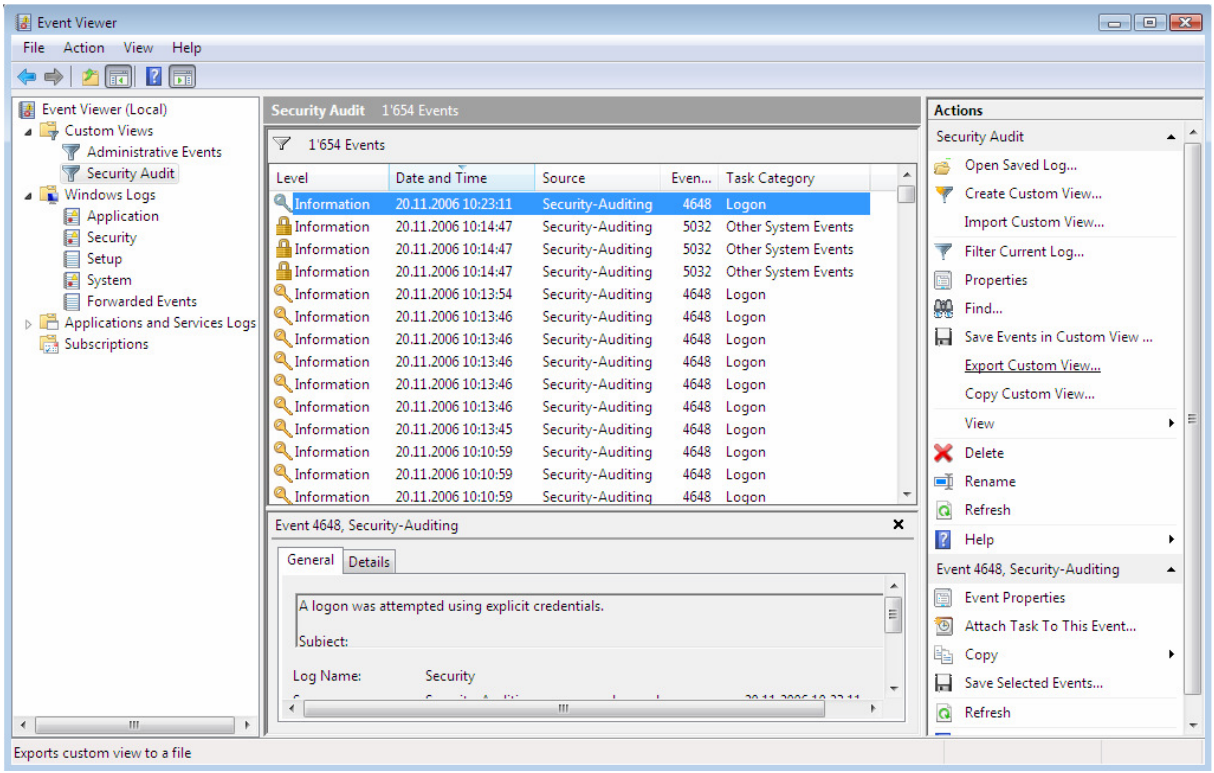
Notre filtre est alors créé et nous pouvons consulter les *logs* recherchés



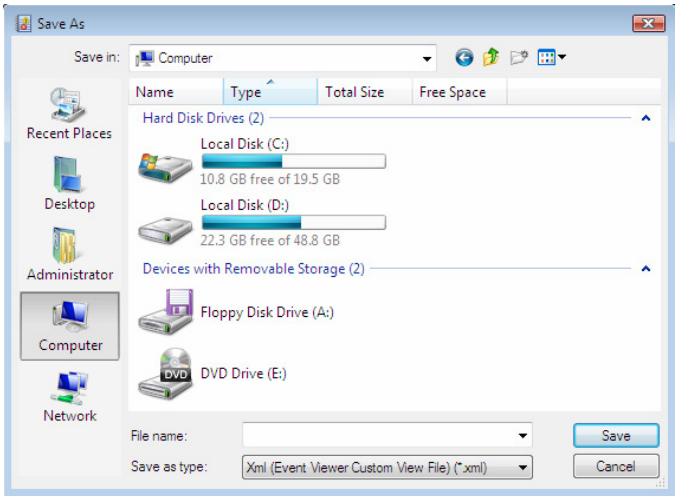
3.3.8.2 Exporter un filtre

Il est possible d'exporter un filtre afin de pouvoir le réutiliser par exemple sur un autre poste, ce qui nous évitera de devoir le recréer.

Sélectionner dans les *Custom Views* le filtre que l'on veut exporter, puis cliquer sur *Export Custom View…* (Fenêtre *Actions*) :



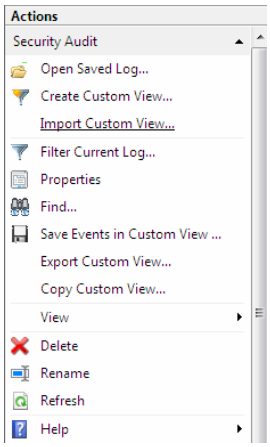
Il nous est alors possible de le sauver sur le disque dur, ou dans un dossier partagé.



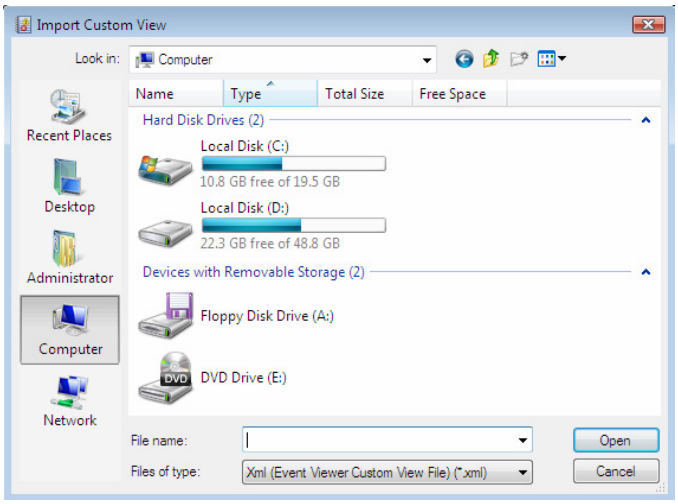
Comme on peut le constater sur cette figure, le filtre sauvé n'est rien d'autre qu'un fichier XML (qui peut aussi être créé à la main).

3.3.8.3 Importer un filtre

Dans la fenêtre *Actions*, sélectionner *Import Custom Views…*



Il nous est alors possible d'importer un filtre :



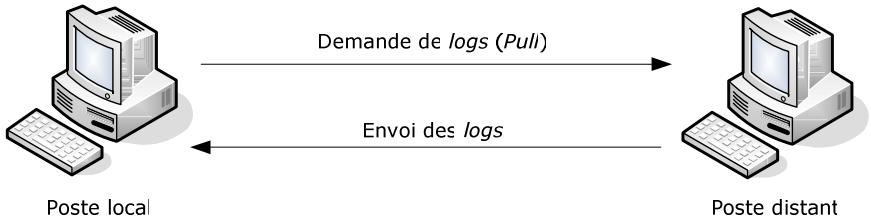
3.3.9 Consulter les *logs* d'un poste distant

Il est possible à l'aide d'*Event Viewer* de se connecter à un poste distant afin de consulter les *logs* qui y sont présents.

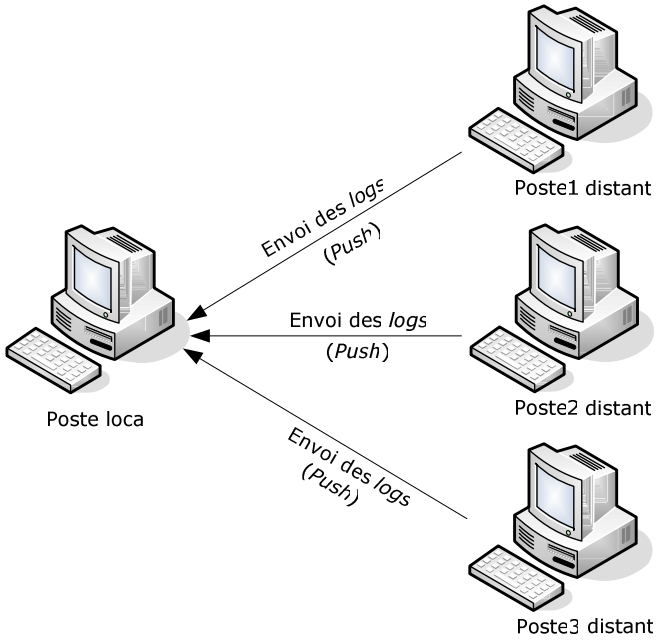
Ces *logs* peuvent aussi être sauvés localement.

3.3.9.1 Méthode de type *Push* ou *Pull* ?

Cette méthode est donc du type ***Pull*** (nous faisons du *pulling*) qui consiste à aller prendre les *logs* sur un ordinateur du réseau.



L'inverse de cette méthode est dit méthode ***Push***, qui consiste à envoyer les *logs* vers une destination. Cette méthode est meilleure que la méthode *Pull*, il est en effet préférable de recevoir automatiquement des *logs* plutôt que d'aller les chercher.



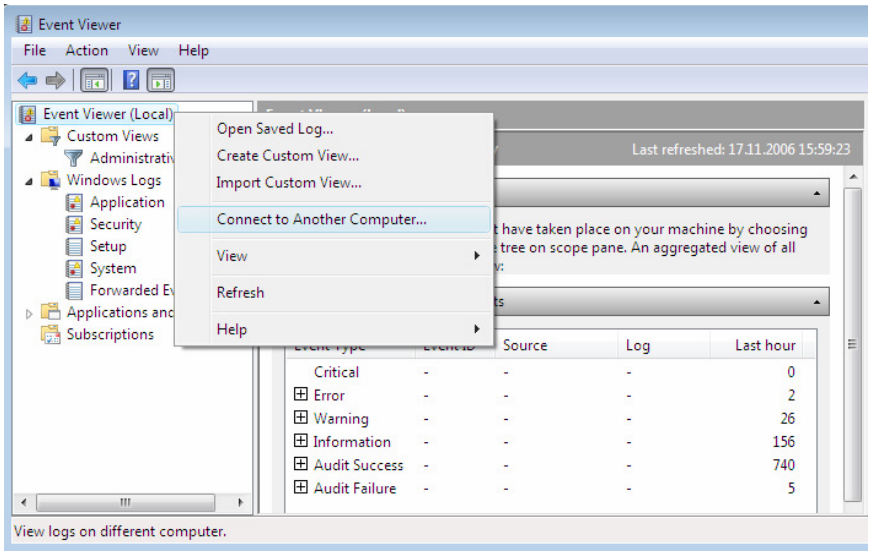
3.3.9.2 Manipulations nécessaires

Les manipulations suivantes ont été effectuées depuis un poste *Windows Vista*, vers un autre poste *Windows Vista*, et aussi vers un poste *Windows Server 2003*.

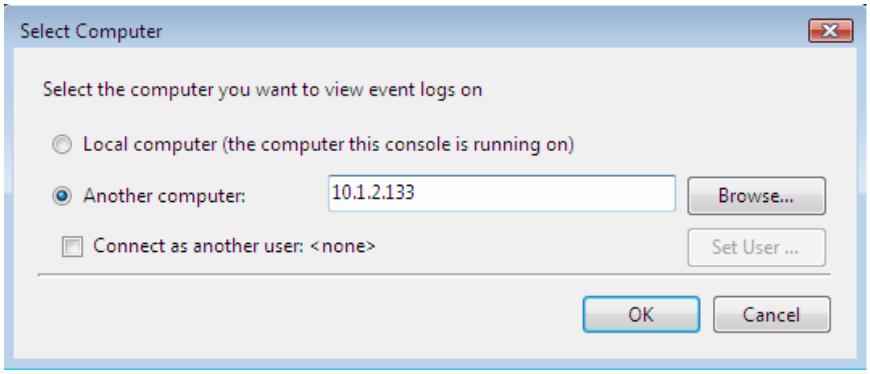
Tous ces différents ordinateurs étaient connectés à un domaine (nécessaire pour effectuer ces manipulations).

Pour simplifier, les *firewall* ont été désactivés.

Afin de se connecter à un poste distant à l'aide d'*Event Viewer*, effectuer un clic droit sur *Event Viewer* (*Local*) puis sélectionner *Connect to Another Computer…*



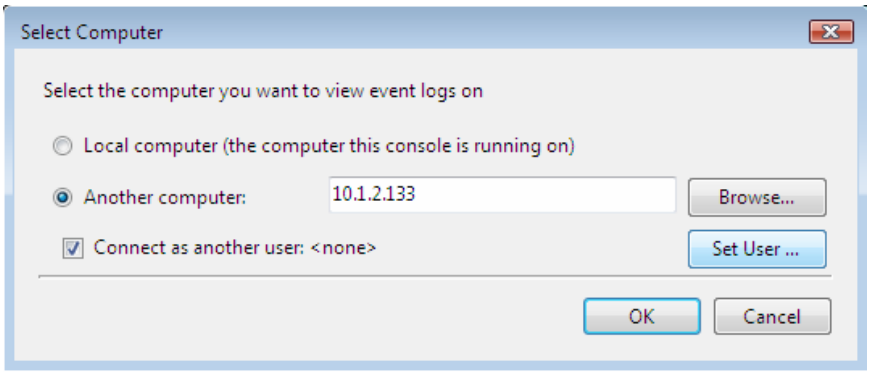
Dans la fenêtre suivante, entrer l'adresse IP (ou le nom d'ordinateur) du poste où l'on veut consulter les *logs*



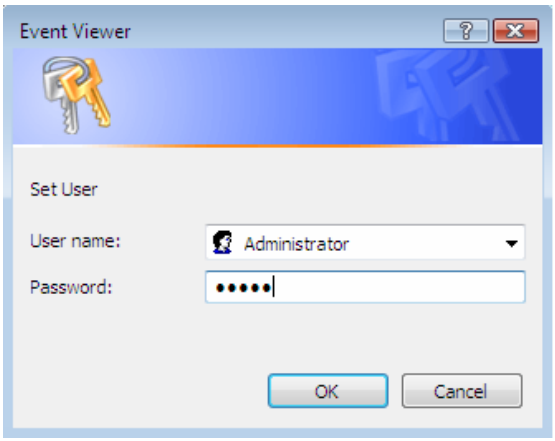
Nous remarquons l'option *Connect as another user*, qui permet de se connecter à un ordinateur distant avec un autre nom d'utilisateur (et mot de passe) que l'utilisateur courant.

Le compte effectuant la demande à distance de lecture des *logs*, doit être membre du groupe *Administrators*.

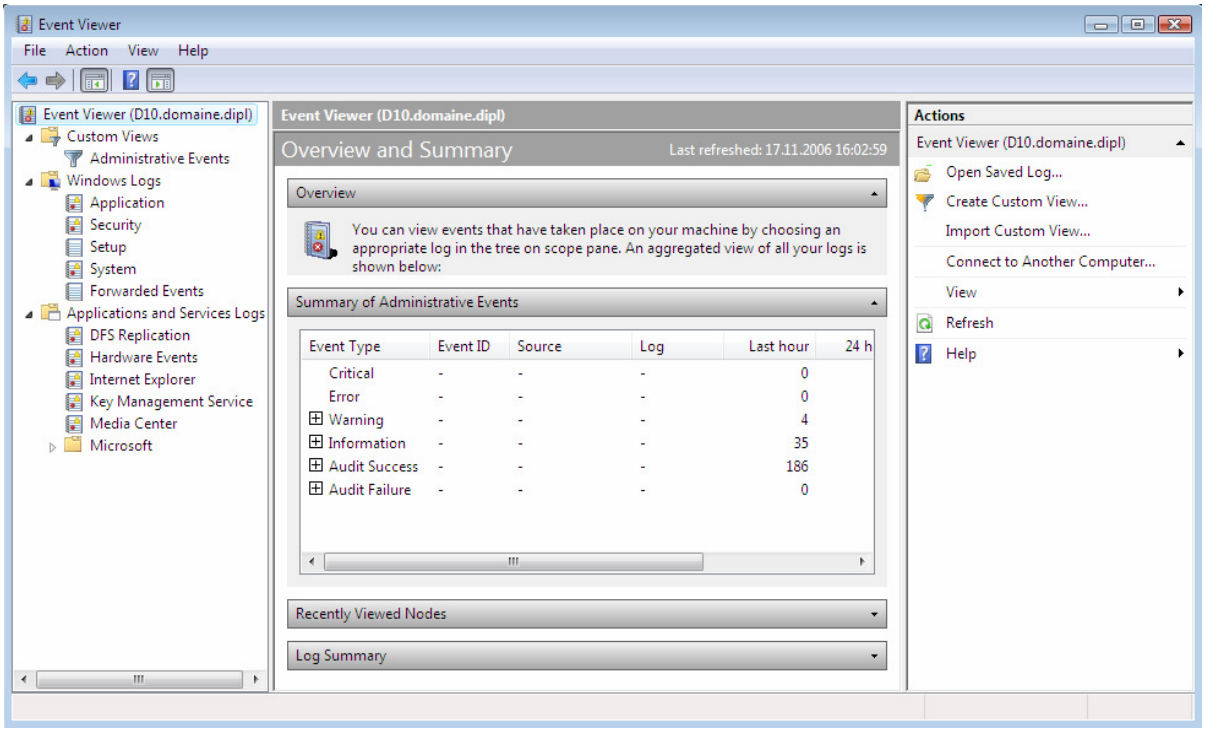
Sélectionner la case *Connect as another user*, puis cliquer sur le bouton *Set User…*



Entrer le nom d'utilisateur et le mot de passe

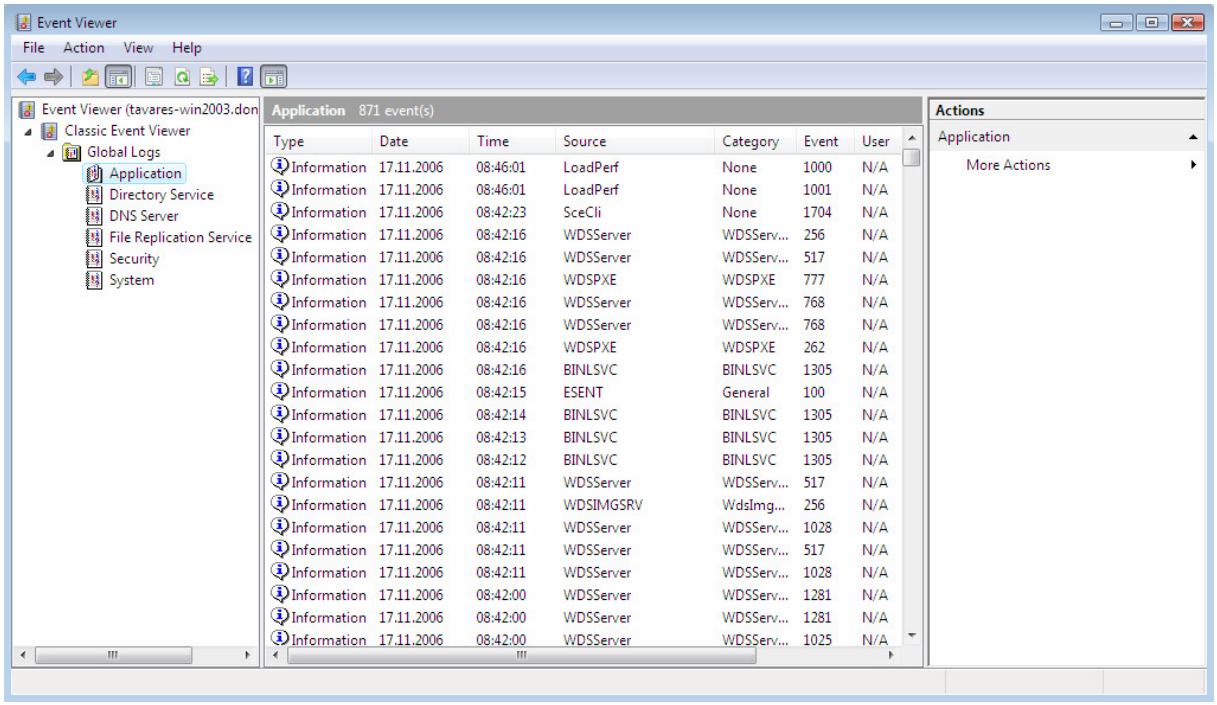


La connexion est alors effectuée, nous pouvons consulter les *logs* du poste distant, qui dans cet exemple est un poste *Windows Vista*.



La même opération peut être effectuée vers un poste de version antérieure à *Windows Vista*.

Dans l'exemple suivant, la demande de lecture des *logs* a été effectuée vers un poste *Windows Server 2003*.
Nous remarquons que la présentation des *logs* à changé. Pour les versions antérieures à *Windows Vista*, *Event Viewer* présente les *logs* comme dans les anciennes versions d'*Event Viewer*.



3.3.10 Centraliser les *logs*

Avec cette nouvelle version d'*Event Viewer*, il est possible de centraliser les *logs* sur une machine du réseau.

Pour effectuer ceci, la machine qui centralise les *logs* doit être un simple poste *Windows Vista* (car il est nécessaire de disposer de la nouvelle version d'*Event Viewer* pour effectuer ceci).

Et par conséquent, seuls les postes *Windows Vista* peuvent envoyer leurs *logs* vers un poste *Vista*, les versions antérieures de *Windows* ne peuvent effectuer ceci du fait qu'elles ne disposent pas de la dernière version d'*Event Viewer*.

J'ai demandé sur les *newsgroups* de *Microsoft*, si la dernière version d'*Event Viewer* à prévu d'être mise en téléchargement afin de l'installer sur les postes antérieurs à *Windows Vista*, mais jusqu'à ce jour je n'ai obtenu aucune réponse.

Les divers postes doivent être connectés à un domaine.

Pour simplifier les *firewall* ont été désactivés.

3.3.10.1 Configuration des postes source

Les postes source sont les postes qui vont envoyer leurs *logs* sur une machine centralisée.

Il faut configurer chaque poste source, pour ceci ouvrir une fenêtre d'invite de commandes avec les privilèges Administrateur, puis effectuer la commande suivante :

`winrm quickconfig`	Configure *WinRM* (*Windows Remote Management*) afin que les *logs* puissent être envoyés. Utilise le protocole *WS-Management.*

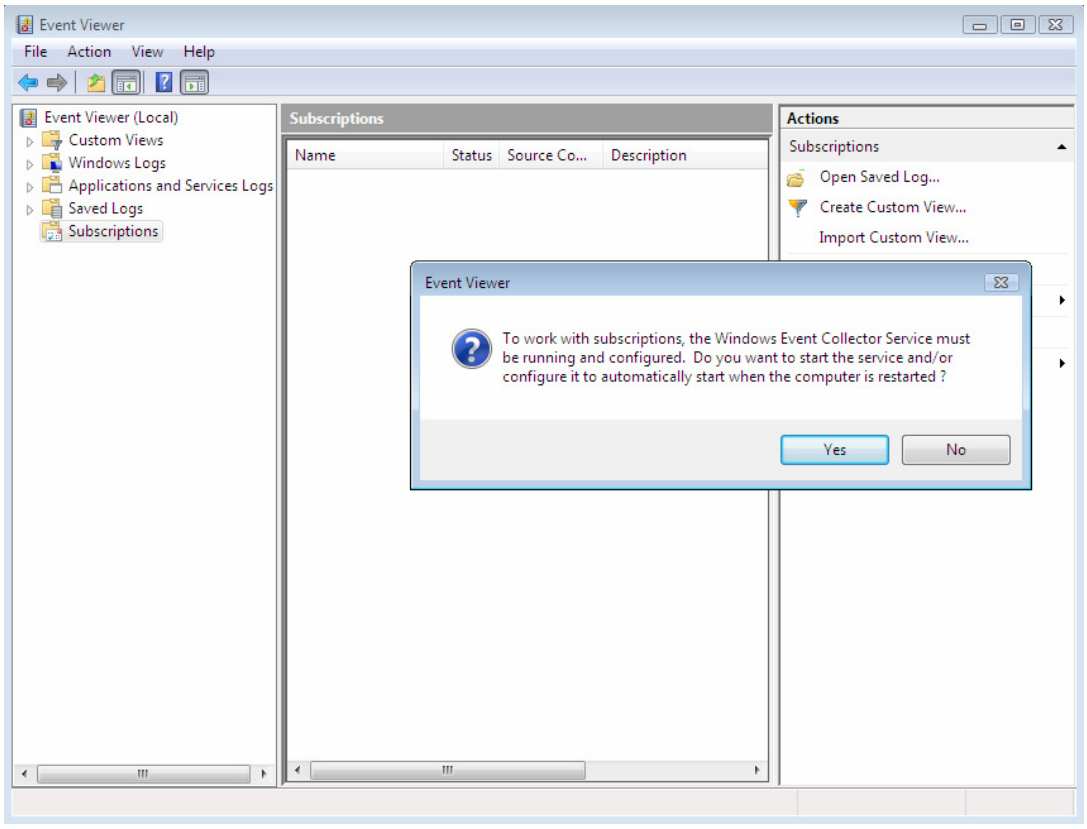
3.3.10.2 Configuration du poste de centralisation

Afin de configurer le poste qui va centraliser les *logs*, effectuer les opérations suivantes :

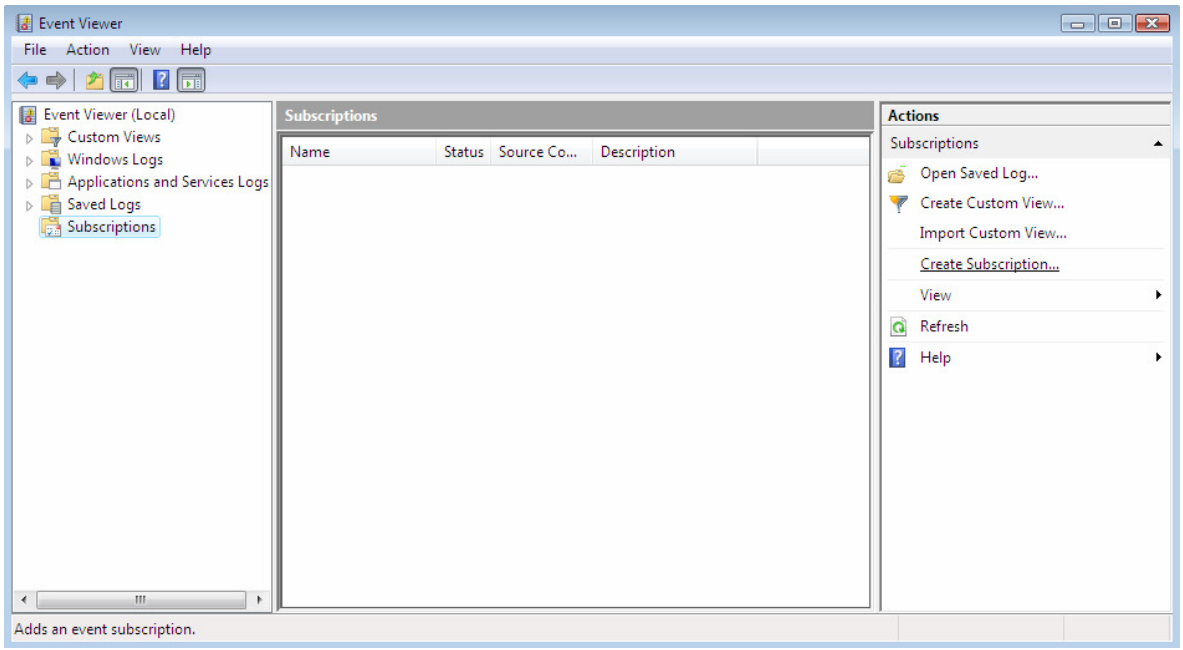
Ouvrir une fenêtre d'invite de commandes puis entrer la commande suivante :

`wecutil qc`	Autorise la création de "*Events Subscriptions*" pour les *logs* envoyés à partir de postes distants supportant le protocole *WS-Management.* Un "*Event Subscription*" spécifie quels sont les *logs* qui vont être collectés et sauvegardés.

Dans *Event Viewer*, sélectionner *Subscriptions* pour pouvoir créer un *Event Subscription*. Dans le cas où *Windows Event Collector Service* n'est pas activé, nous aurons le message suivant nous demandant la confirmation de son activation :

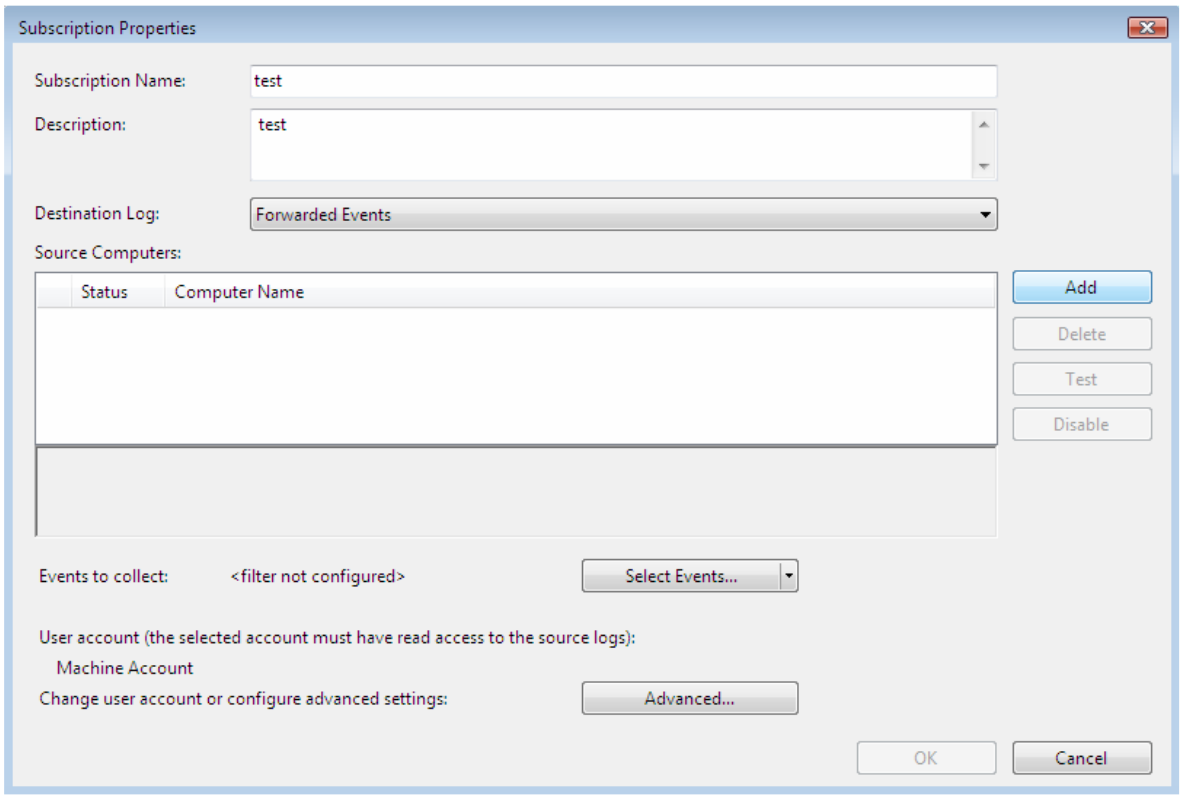


Accepter son activation, puis sélectionner *Create Subscription…* dans la fenêtre *Actions* :



Définir un nom pour la "*Subscription*", puis une description (optionnelle).
Choisir la destination pour l'enregistrement des *logs* reçus (*Destination Log*), par défaut les *logs* sont sauvegardés dans *Forwarded Events* (évènements transmis).

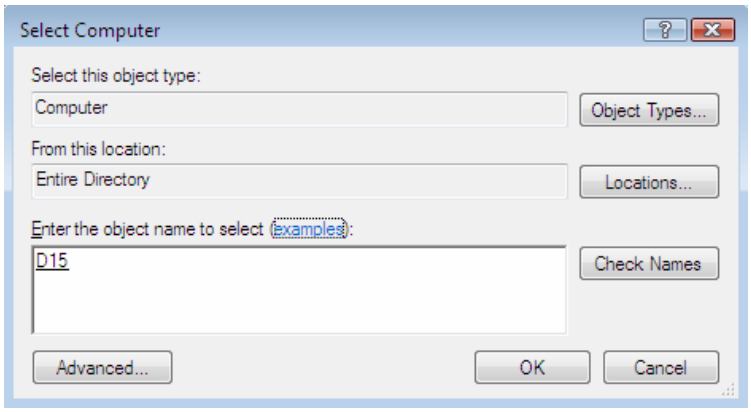
Pour ajouter un poste à partir duquel nous souhaitons l'envoi de ses *logs*, cliquer sur le bouton *Add* :



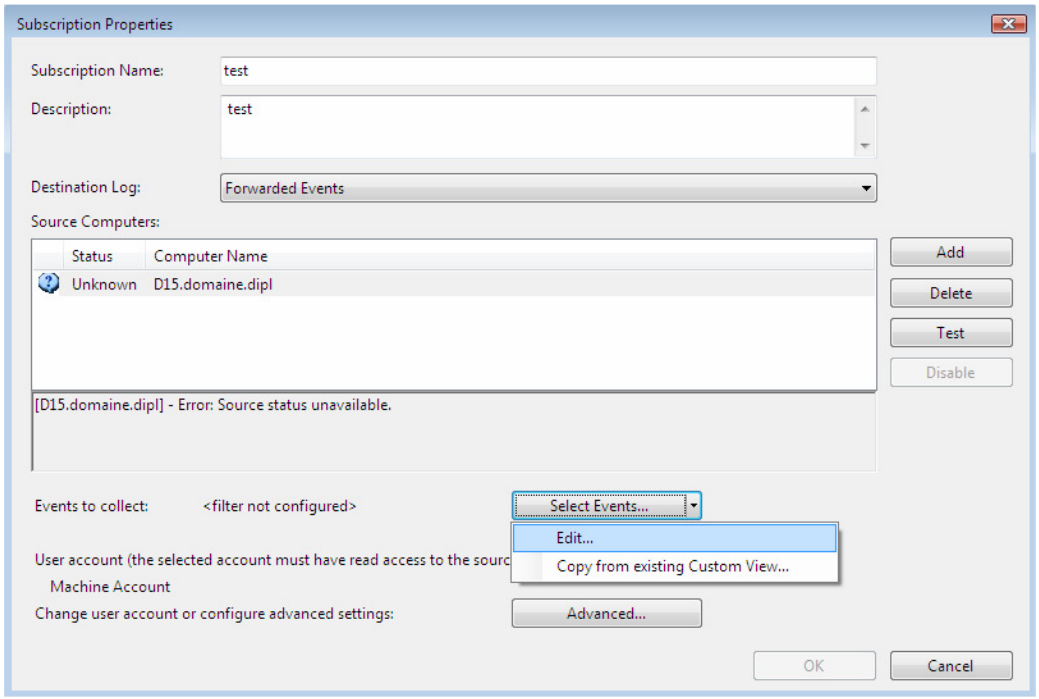
Remarque : Si les ordinateurs ne sont pas connectés à un domaine, le bouton *Add* renverra une erreur !

Nous remarquons que c'est sur le poste collecteur que l'on doit spécifier les ordinateurs qui doivent nous envoyer leurs *logs* et non sur les postes distants!
Il y aura donc une phase de dialogue entre le poste collecteur et les postes distants avant transmission des *logs*.

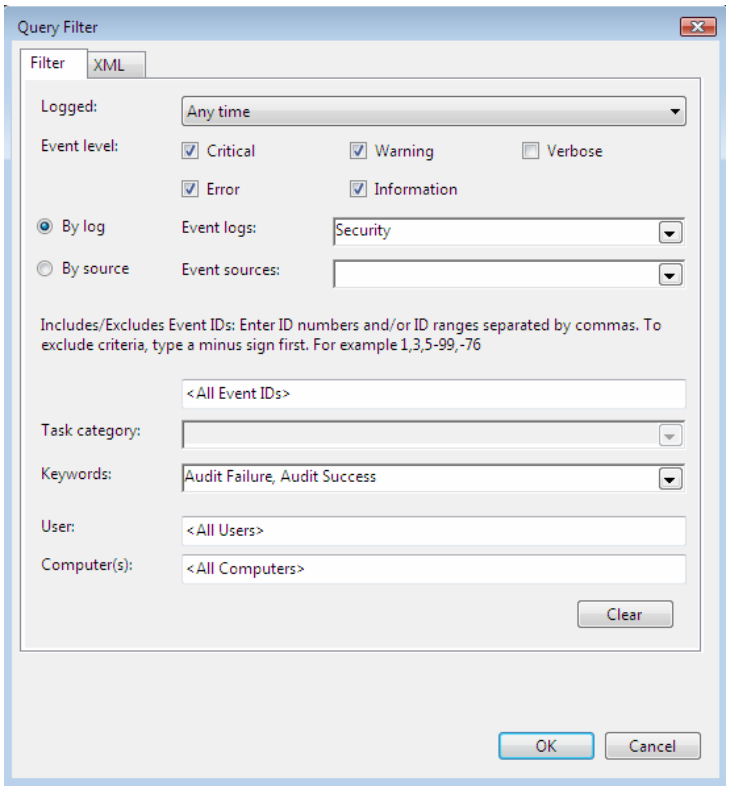
Dans la fenêtre suivante, sélectionner le poste qui devra nous envoyer des *logs* puis valider avec le bouton OK.
Lors de mes tests au laboratoire, le poste distant avait comme nom D15.



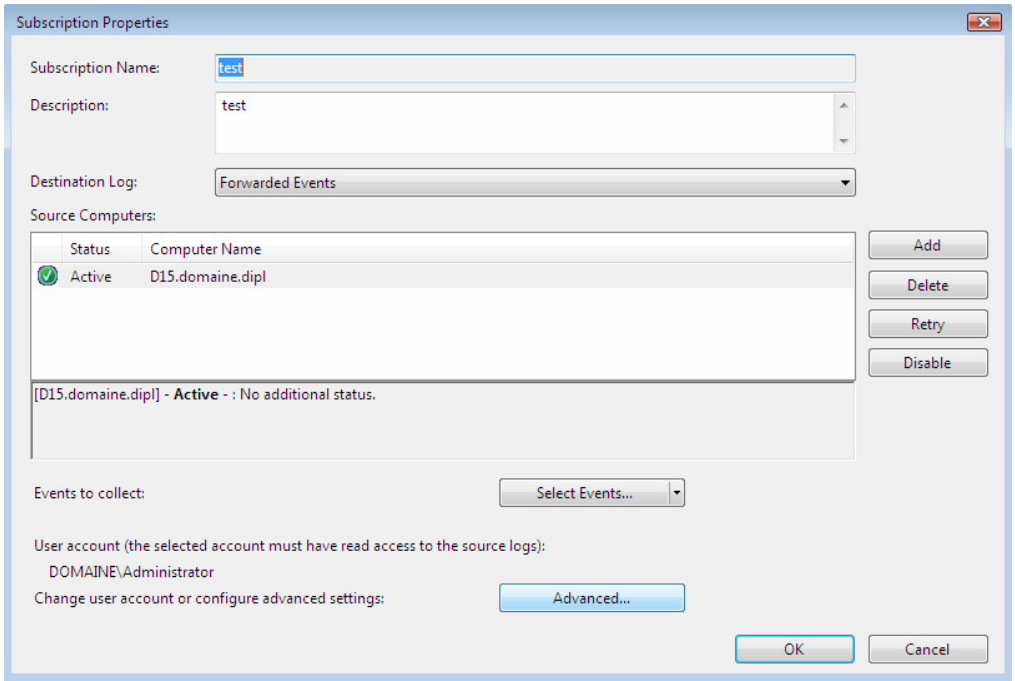
Il est possible de créer un filtre comme vu dans le paragraphe 3.3.8.1 page 139, pour cela sélectionner *Select Events…* puis *Edit…*



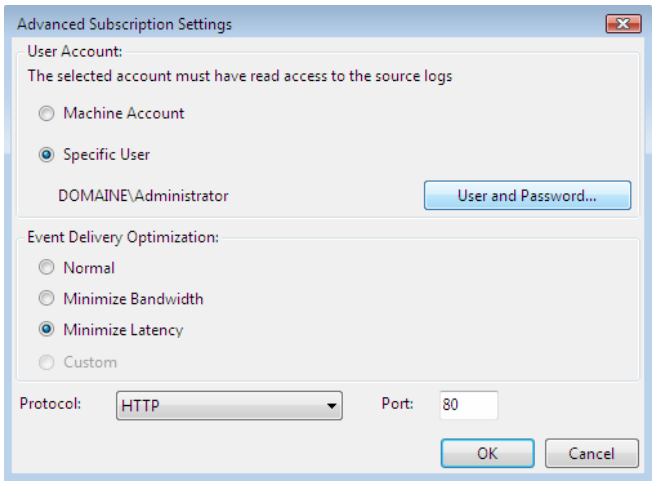
Le filtre a été configuré à titre d'exemple comme dans le paragraphe 3.3.7.1, afin de collecter uniquement les informations *Audit Success* et *Audit Failure*.

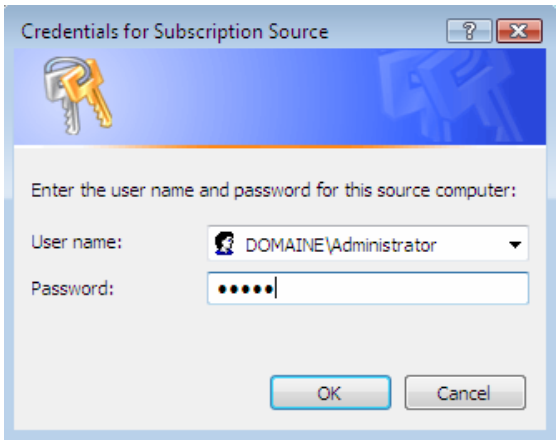


Sélectionner *Advanced…* afin de configurer les options avancées de notre "*Subscription*"



Nous pouvons alors définir le compte utilisateur qui sera utilisé pour se connecter aux machines distantes. Ce compte soit être membre du groupe *Administrators* sur le domaine.





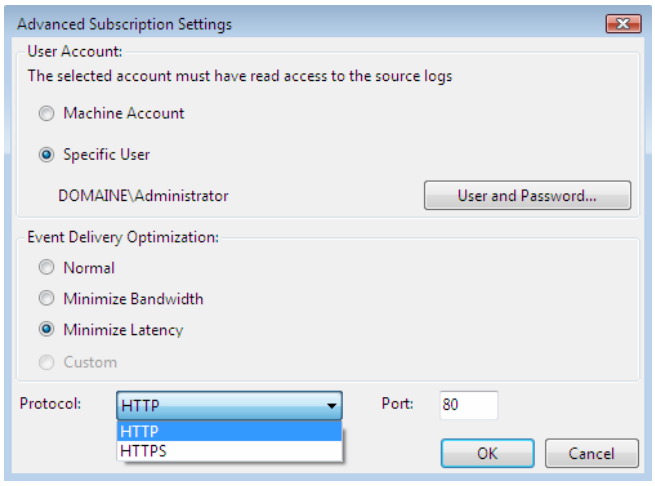
Nous pouvons constater sur l'image ci-dessus 3 modes pour l'envoi de *logs* :

- *Normal*, c'est le mode par défaut, qui emploie la méthode *pull*, ce n'est donc pas le meilleur mode pour l'envoi de *logs*, comme cité précédemment en page 145.

- *Minimize Bandwidth*, qui minimise la bande passante utilisée, ce mode emploie la méthode *push*, cependant les *logs* ne sont pas émis immédiatement. Selon l'aide disponible avec *Event Viewer*, les *logs* sont envoyés toutes les 6 heures.

- *Minimize Latency*, qui assure l'envoi de *logs* dans un délai minimum, ce mode emploie la méthode *push*.
  C'est le mode qui est préférable, surtout si l'on désire recevoir les *logs* de type critiques/erreur le plus rapidement possible.

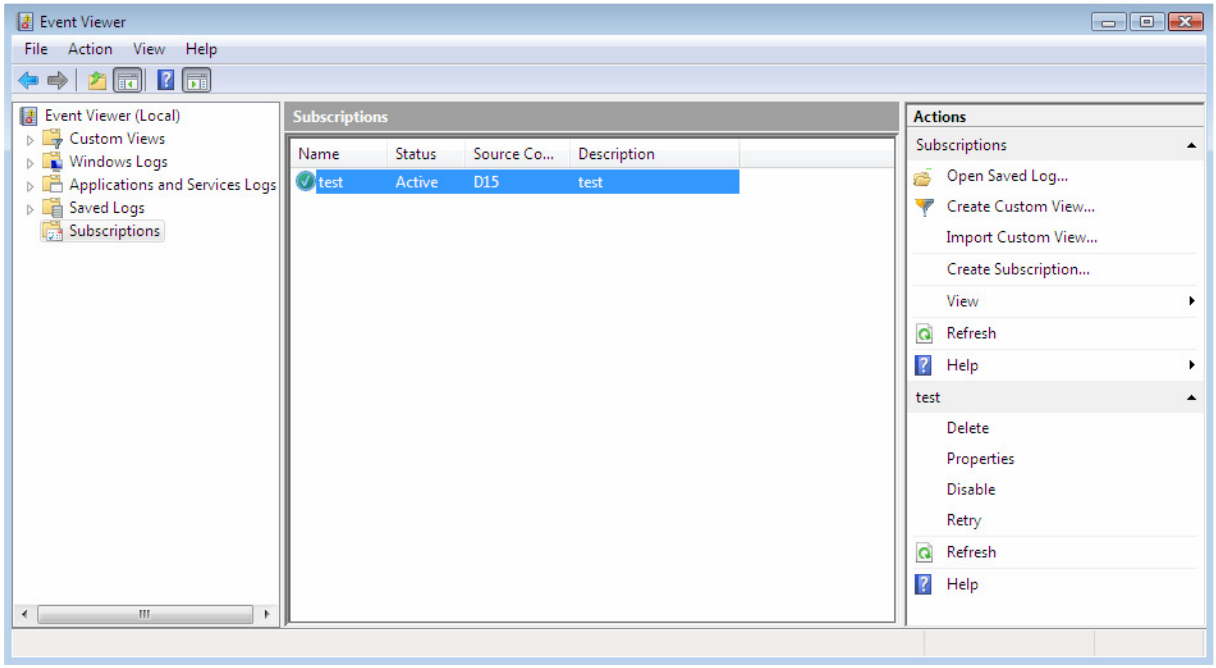
Nous pouvons aussi choisir le protocole utilisé (ainsi que le port) pour l'envoi de *logs* :



Nous pouvons choisir le protocole HTTP ou le protocole sécurisé HTTPS.

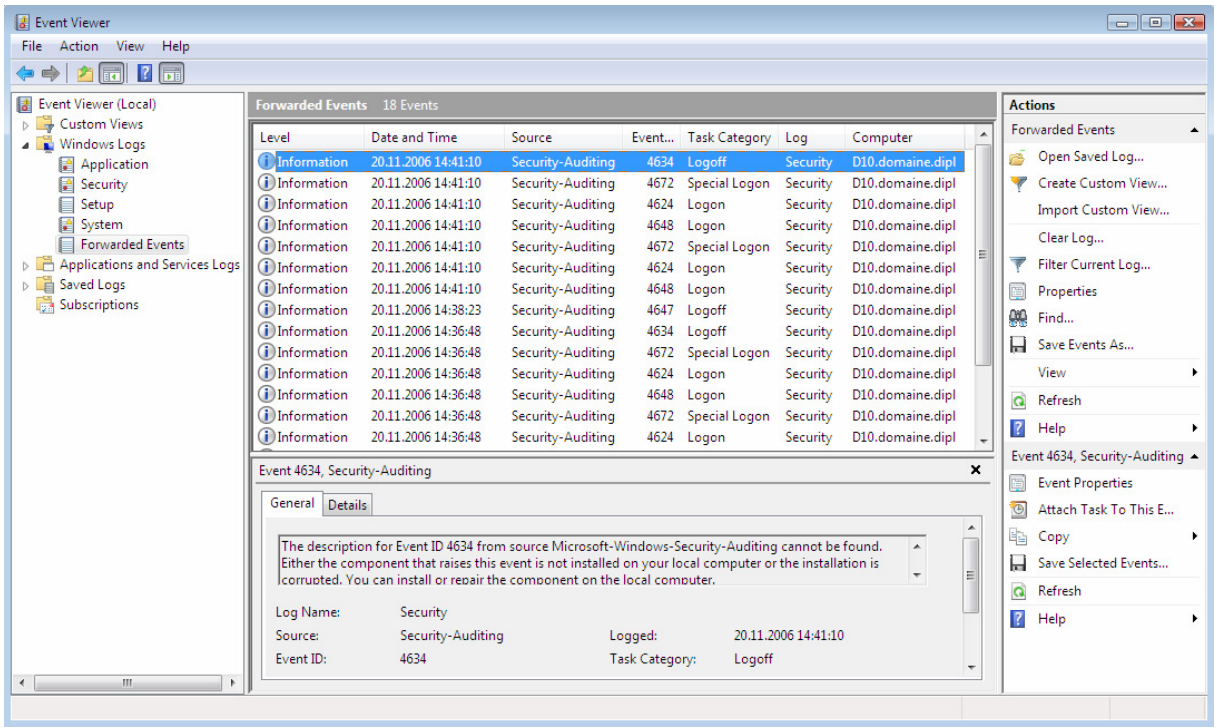
Nous pouvons donc en déduire que les *logs* sont envoyés en mode connecté TCP et donc avec contrôle d'erreur.

En cliquant sur OK, nous pouvons constater si tout se déroule correctement, que la connexion est active (ici connexion avec le poste D15) :



Remarque : L'authentification de l'utilisateur s'effectue avec le protocole *Kerberos* !

Les *logs* seront alors reçus de ce poste, et seront enregistrés dans *Forwarded Events*



Nous pouvons constater le champ *Computer*, qui permettra ensuite de pouvoir filtrer relativement simplement les *logs* reçus par les divers postes !

Remarque : Seuls les *logs* produits à partir du moment de création de la connexion seront envoyés !

3.3.11 Du point de vue sécurité

Afin de se connecter à un ordinateur du réseau et effectuer des opérations de lecture, il est nécessaire d'être à l'intérieur d'un domaine et avoir les droits suffisants (être membre du groupe *Administrators*).

En ce qui concerne la centralisation des *logs*, les ordinateurs doivent aussi être à l'intérieur d'un domaine, disposer aussi des droits suffisants afin de se connecter aux diverses machines.
De plus, l'authentification se fait à travers le protocole *Kerberos*, qui est un protocole réputé pour être très bon niveau sécurité.

D'après *Microsoft*, il serait possible de centraliser des *logs* de postes ne faisant pas partie d'un domaine, cependant j'ai essayé brièvement cette méthode avec *Windows Vista RC2* mais sans résultats.

De plus, en centralisant les *logs*, il nous est possible d'effectuer des statistiques sur l'ensemble des *logs* reçus et aussi d'effectuer un *backup* de la totalité des *logs* centralisés.

Il est très pratique aussi d'effectuer des analyses de *troobleshooting* (résolution de problèmes) en centralisant les *logs*. Nous pourrons par exemple constater certains problèmes ou erreurs récursives sur divers postes du réseau et ainsi identifier plus facilement le problème.

Conclusion

Dans la première partie de mon travail de diplôme, nous avons pu analyser les concepts, outils et diverses opérations nécessaires à la création et au déploiement d'images Windows Vista.

Lors de la deuxième partie, nous avons découvert le nouveau *shell* nommé *PowerShell*, et ainsi nous avons pu noter les similarités et les différences avec MS-DOS et les *shells* disponibles dans le monde *Unix*/*Linux*.

Enfin lors de la dernière partie de ce diplôme, nous avons mis en avant la nouvelle version de l'outil *Event Viewer* avec notamment la possibilité de centraliser les journaux d'évènements de plusieurs postes *Vista.*

Suite à ces trois études, nous pouvons maintenant répondre à la question suivante : est-il intéressant pour une entreprise d'utiliser ces outils notamment en prenant en compte les solutions existantes et les outils tiers déjà présents sur le marché ?

Un premier point à prendre en compte est le nouveau format d'image *WIM* qui offre de nombreuses possibilités, tels que l'installation non destructive, le fait de pouvoir incorporer plusieurs images différentes au sein du même fichier (sans pour autant que la taille de ce fichier *WIM* augmente considérablement) et la possibilité de gérer les images en mode Offline. Toutes ces possibilités sont aujourd'hui uniques et ne sont pas proposées par les solutions concurrentes.

De plus, *Microsoft* met à disposition un serveur d'images gratuit (serveur *WDS*), qui permet de déployer relativement facilement ces mêmes images sur les différents postes du réseau.

Enfin, tous les outils créés spécialement pour gérer les images *Windows Vista* sont fournis avec le système d'exploitation, ceci représente un point important lorsque nous considérons le prix de certaines solutions standards du marché comme *Ghost*.

Je conseille donc à toute entreprise désirant déployer *Windows Vista* (y compris les entreprises disposant déjà de solutions comme *Norton Ghost*), d'évaluer les divers outils proposés par *Microsoft*, en particulier si l'entreprise dispose d'au minimum une licence *Windows Server 2003*, nécessaire pour l'installation d'un serveur *WDS*.

Dans le cas contraire, si l'entreprise ne dispose pas de licence *Windows Server 2003* afin d'installer un serveur *WDS*, et qu'elle possède déjà des logiciels de création d'images et déploiement tels que *Norton Ghost*, il leur faudra faire un choix : soit continuer à utiliser *Norton Ghost* (impossible de gérer les images en mode *offline*, pas de possibilité d'incorporer plusieurs images au sein du même fichier, pas d'installations non destructives), soit se procurer une licence *Windows Server 2003* pour en bénéficier.

En ce qui concerne *PowerShell*, nous avons vu qu'il s'agit d'une réelle évolution par rapport à l'interpréteur de commande hérité de MS-DOS sans toutefois s'y substituer entièrement. Ce nouveau *shell* bénéficie de grande amélioration et peut se comparer aux *shells Unix* ou *Linux* qui restent tout de même des références en la matière.

Pour finir, la nouvelle version d'*Event Viewer* permet de gérer les *logs* efficacement. Il offre de bonnes capacités de filtrage, que ce soit simplement pour consulter les journaux d'évènements que pour les trier à des fins de centralisation.
Les échanges se font via les protocoles HTTP ou HTTPS, ce qui permet un contrôle d'erreur ainsi que la sécurisation des *logs* en utilisant le protocole HTTPS. Il est de plus possible de choisir le port utilisé pour les échanges. Malheureusement pour le moment, il n'est pas possible de centraliser les *logs* de postes antérieurs à *Windows Vista* (nous pouvons uniquement nous connecter à ces postes et consulter les *logs* qui y sont présents)

En conclusion, *Windows Vista* représente une évolution majeure dans le domaine de la gestion des systèmes *Windows*. L'ensemble des concepts et outils ont été revus pour aider l'administrateur tout au long du cycle de vie du poste : de sa conception à sa maintenance quotidienne en passant par son pilotage quotidien. Nul doute que ces nombreuses possibilités doivent être prises en compte par les administrateurs de parc *Windows* afin de simplifier, améliorer, voire standardiser leurs procédures d'exploitation et ceci pour un coût entièrement inclus dans l'achat de la licence de *Vista* elle-même.

Annexes

A.1 Créer un CD *bootable Windows PE* personnalisé

Requis préalables :

- Avoir une installation de ***Windows AIK***
- Avoir un graveur CD ou DVD à disposition

Outils utilisés :

- ***ImageX*** (créer et gérer les images WIM)
- ***Peimg*** (ajouter des packages à une image)
- ***Oscdimg*** (créer un fichier ISO de Windows PE afin de pouvoir être gravé sur CD)

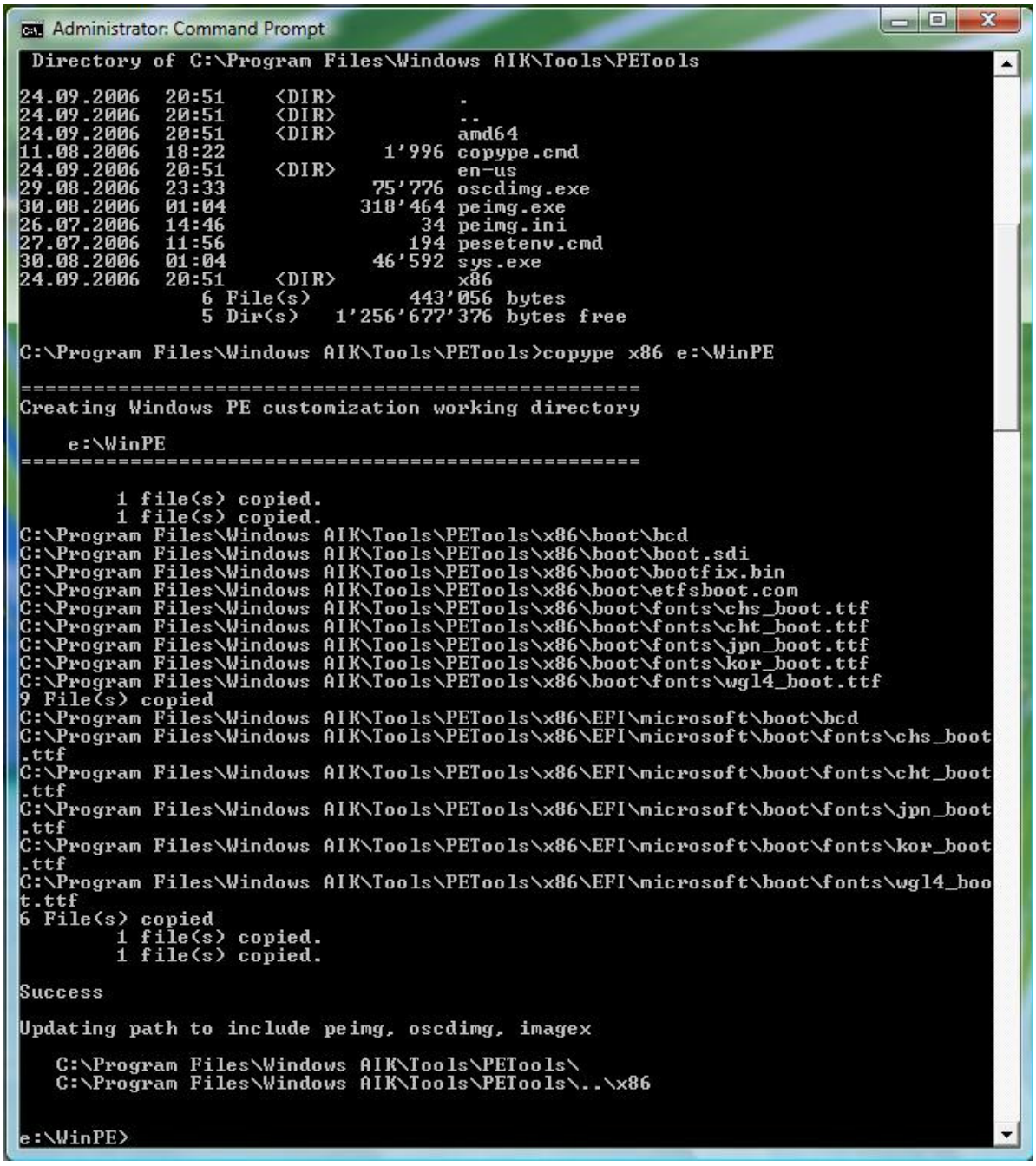
Scénario :

Nous désirons créer un CD *Windows PE* personnalisée, avec l'outil *ImageX* qui nous permettra de capturer et installer des images *WIM*.
De plus nous voulons effectuer une image de *Windows PE* la plus petite possible afin que son chargement soit le plus court possible.

A.1.1 Créer la structure de fichiers *Windows PE* :

Exécuter les différentes commandes dans l'ordre :

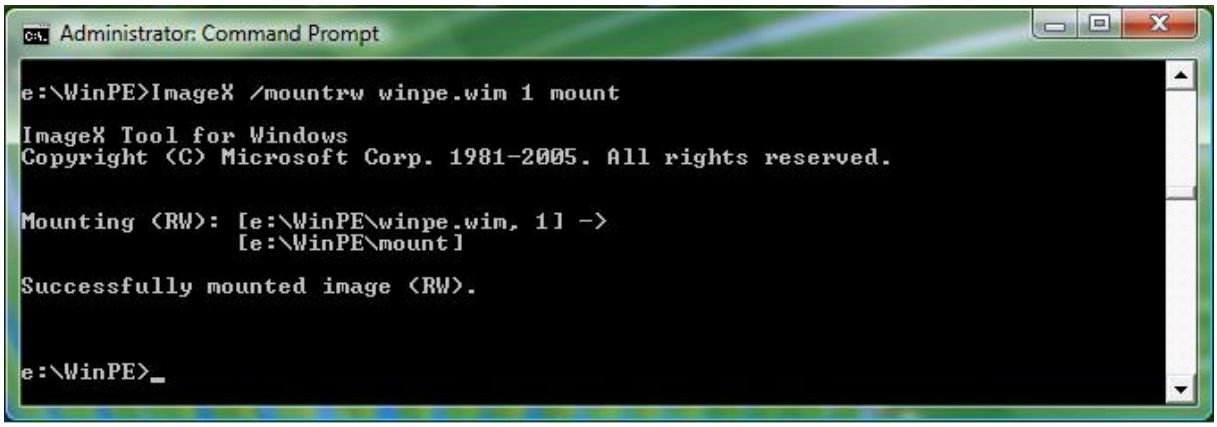
cd C:\Program Files\Windows AIK\Tools\PETools	Dans le répertoire contenant les programmes (*tools*) que nous devons utiliser
COPYPE X86 e:\WinPE	Prépare les fichiers de *Windows PE* pour traitement vers **e:\WinPE** Ici **e**:\ est une autre partition, il est possible de le faire sur la même partition que le système d'exploitation



A.1.2 Monter l'image pour la personnalisation :

Monter une image signifie rendre visible le contenu réel d'une image, vers un dossier par exemple. (Voir ce qu'il y a à l'intérieur de cette image, les différents fichiers, dossiers).

ImageX /mountrw winpe.wim 1 mount	"monte" l'image de *Windows PE* (**winpe.wim**) vers le répertoire **mount**. C'est dans ce répertoire que vont se faire les modifications. L'option **/mountrw** monte l'image en *read-write* (lecture-ecriture) Au lieu de **/mountrw**, il est possible d'utiliser l'option **/mount** dans le cas ou l'on ne souhaite pas modifier l'image.



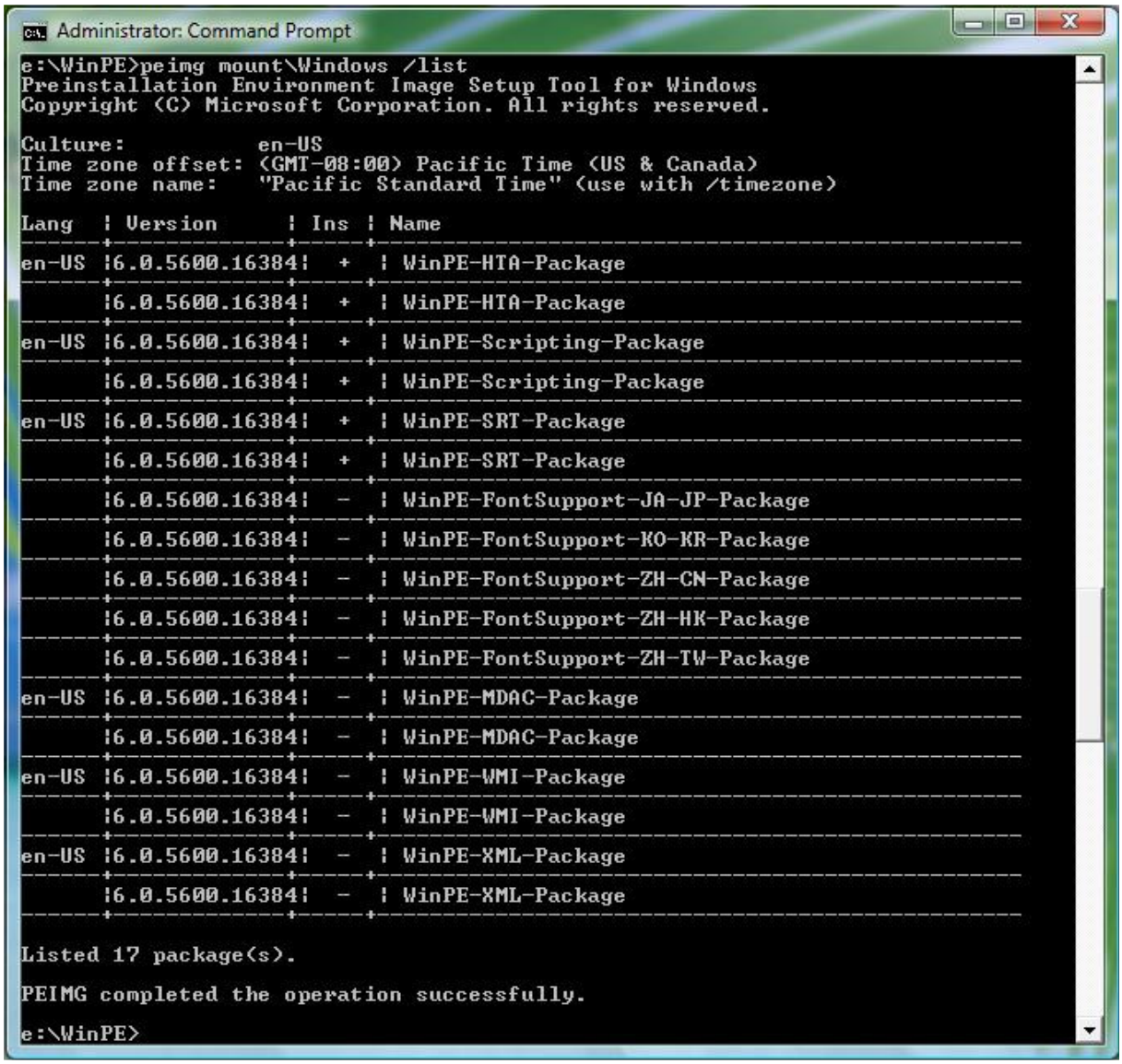
A.1.3 Insertion de *packages* dans l'image de *Windows PE* :

Ici un *package* est un ensemble de fichiers nous permettant d'utiliser des outils par exemple pour afficher du HTML, permettre l'utilisation de scripts, etc.

Afin de créer uniquement une image de notre système, il n'est pas vraiment nécessaire d'ajouter des *packages*, car ces *packages* ne nous offrent pas grand chose du point de vue création/installation d'images.
Le seul *package* qui pourrait être intéressant est le *package SRT* (*Server Recovery Tools*) qui peut permettre de réparer des installations qui ne se sont pas correctement déroulées.

Voici les commandes pour installer quelques *packages* si besoin :

peimg Mount\Windows /install= winpe-hta-package	Installe le *package* HTA, qui permet d'afficher du HTML. Celui-ci peut être utilisé pour informer les utilisateurs du nouvel environnement.
peimg Mount\Windows /install= *SRT*	Installe le *package* SRT (*Server Recovery Tools*) qui permet de réparer des installations qui ne se sont pas correctement déroulées.
peimg Mount\Windows /install= winpe-Scripting*	Installe le *package Scripting* qui permet l'utilisation de scripts afin d'automatiser des tâches dans l'environnement WinPE.
peimg Mount\Windows /list	Affiche la liste des *package* disponibles. Les *packages* qui sont installés comportent des signes + dans la colonne "Ins"



Les packages ajoutés correctement on des signes "+"

Ajouter *ImageX* à notre image de *Windows PE* :
Ceci peut se faire via l'explorer ou à travers la ligne de commande :

Copy "C:\Program files\Windows AIK\Tools\x86\imagex.exe" e:\WinPE\mount\Program Files	Copie imagex.exe vers le répertoire de notre *Windows PE* personnalisé.

Remarque : Il est conseillé de copier uniquement les programmes et packages réellement nécessaires, ceci car lors du boot de notre *Windows PE*, toute l'image sera copiée en mémoire RAM, et plus nous lui ajoutons de programmes, plus l'occupation en mémoire RAM sera grande.

En outre, plus l'image de *Windows PE* sera grande, plus son temps de chargement sera long !

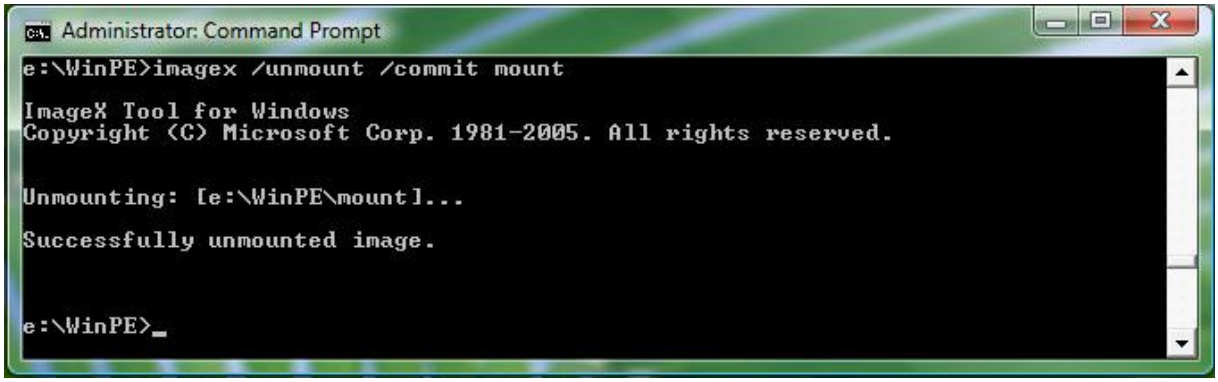
A.1.4 Créer l'image personnalisée de *Windows PE* :

Maintenant que nous avons effectué toutes les modifications voulues à *Windows PE*, il ne nous reste plus qu'à créer son image au format WIM à l'aide d'*ImageX*.

Il y a 2 manières différentes d'effectuer ceci.

Première méthode :

ImageX /Unmount /commit mount	Le paramètre **/commit** applique les changements effectués à l'image de base *Windows PE*. Le paramètre **/Unmount** "Démonte" l'image du répertoire **mount**, qui est à présent vide.



Cette méthode ne nécessite pas de devoir recréer une image complète, elle copie uniquement les changements à l'intérieur de l'image de base de *Windows PE*. Elle est par conséquent plus rapide que la méthode ci dessous.

Deuxième méthode :

ImageX /boot /capture mount WinPECustom.wim "Windows PE Custom Build"	Le paramètre **/boot** permet de rendre l'image de *Windows PE* bootable. Le paramètre **/capture** effectue une nouvelle capture de notre *Windows PE* (dossier **mount**), et donc une nouvelle image nommée **WinPECustom.wim** Cette commande nécessitera quelques minutes d'attente.
ImageX /Unmount mount	Le paramètre **/Unmount** "Démonte" l'image du répertoire **mount**, qui est à présent vide

Cette méthode est plus lente que la première méthode, du fait qu'elle effectue une toute nouvelle image de *Windows PE*.

Si nous allons maintenant regarder dans notre répertoire de travail, qui est **e:\WinPE\ ,** nous constatons que nous avons 2 images au format *WIM* sur notre disque dur, une ayant comme nom **WinPE.wim** qui est l'image *Windows PE* de base, et une autre ayant comme nom **WinPECustom.wim**, qui est notre image de *Windows PE* personnalisée.

A.1.5 Créer un CD *bootable* avec notre image de *Windows PE* :

Nous pouvons maintenant créer un CD *bootable* avec notre image *WIM*.

Ceci va se faire à travers l'outil OSCDIMG.

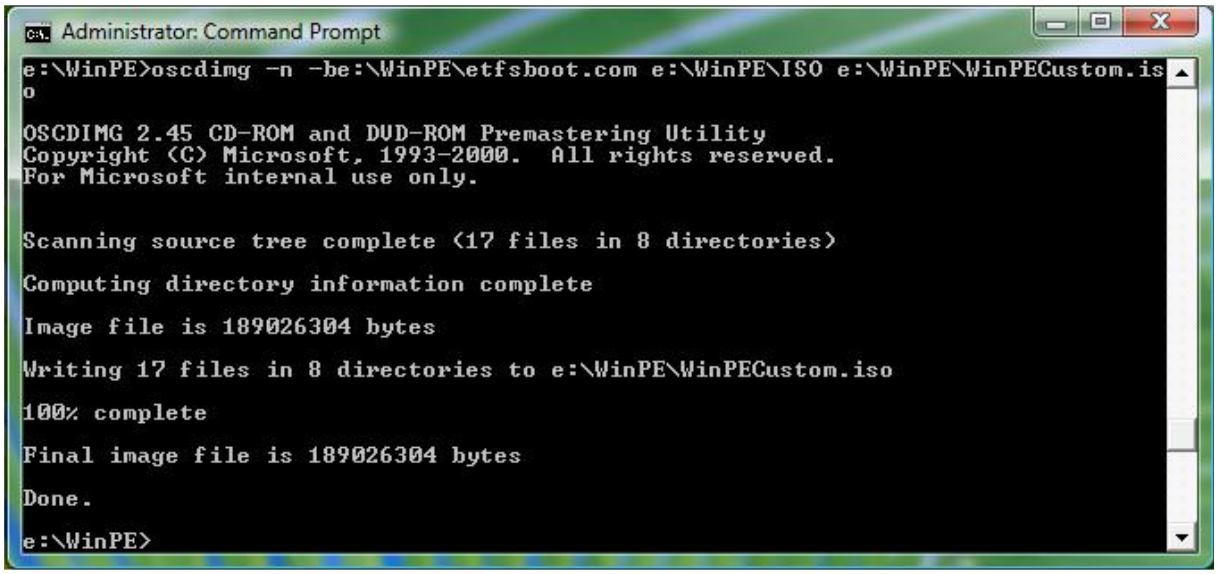
Il faut commencer par copier notre image *WIM* personnalisée de *Windows PE* vers le répertoire ISO. Ce répertoire sera utilisé pour créer une image ISO de *Windows PE*, afin de pouvoir la graver sur CD.

Si lors de la phase Créer l'image customisée de *Windows PE* la première méthode a été choisie :

Copy "C:\WinPE\winpe.wim" e:\WinPE\ISO\sources	Copie notre image de *Windows PE* personnalisée vers le répertoire **e:\WinPE\ISO\sources**
Cd e:\WinPE\ISO\sources	On se place dans le répertoire où sont à présent les images de *Windows PE*
Del boot.wim	Efface l'image de base de *Windows PE* (**boot.wim**)
Ren winpe.WIM boot.wim	Renomme notre image personnalisée de *Windows PE*
OSCDIMG –n –be:\WinPE\etfsboot.com e:\WinPE\ISO e:\WinPE\WinPECustom.iso	Crée l'image de *Windows PE* au format ISO, pour pouvoir être gravé sur CD. Cette opération durera quelques minutes. L'image ISO peut maintenant être gravée sur un CD.

Si la deuxième méthode a été choisie :

Copy "C:\WinPE\WinPECustom.wim" e:\WinPE\ISO\sources	Copie notre image de *Windows PE* customisée vers le répertoire **e:\WinPE\ISO\sources**
Cd e:\WinPE\ISO\sources	On se place dans le répertoire où sont à présent les images de *Windows PE*
Del boot.wim	Efface l'image de base de *Windows PE* (**boot.wim**)
Ren WinPECustom.WIM boot.wim	Renomme notre image personnalisée de *Windows PE*
OSCDIMG –n –be:\WinPE\etfsboot.com e:\WinPE\ISO e:\WinPE\WinPECustom.iso	Crée l'image de *Windows PE* au format ISO, pour pouvoir être gravé sur CD. Cette opération durera quelques minutes. L'image ISO peut ensuite être gravée sur un CD.



Option :

peimg mount\ /prep	Le paramètre **/prep** optimise la taille de *Windows PE*, lorsque l'image est "montée" dans un répertoire (ici **mount**). Cependant, en effectuant cette commande, il ne sera plus possible de remodifier notre image de *Windows PE* par la suite.

A.2 Windows System Image Manager (WSIM)

A.2.1 Introduction

Lorsqu'une entreprise souhaite installer un nouveau système d'exploitation sur tout (ou une partie) de ses ordinateurs, elle nécessite le besoin d'automatiser et personnaliser les installations.

Dans *Windows Vista*, cette automatisation et une partie de la personnalisation peut se faire à l'aide d'un fichier dit "fichier de réponse" ou "*answer file*" en anglais (le terme anglais sera repris dans la suite du document).

Une telle installation automatisée et personnalisée est appelée "*Unattended installation*" par *Microsoft*. Le fichier sera nommé *Unattend.xml* (format XML).

En effet il n'est pas possible de personnaliser entièrement une installation *Vista* avec uniquement une *answer file*.
Ce fichier permet de personnaliser uniquement les *components* et *packages* pouvant être paramétrés et qui sont contenus dans l'image *WIM* (voir ci dessous pour la définition d'un *component* et *package*).

Or, une entreprise nécessite le besoin d'installer divers programmes supplémentaires pour ses employés (un antivirus par exemple).
Ces programmes supplémentaires ne peuvent être configurés à l'aide d'une *answer file*, ils doivent être configurés au moment de leur installation sur un *Windows* de référence.

Il faudra par la suite utiliser *sysprep* afin de préparer notre *Windows* de référence pour le déploiement, capturer son image puis finalement créer une *answer file* associée à notre image.

L'outil qui permet de créer une *answer file* est *Windows System Image Manager* (abréviation "*Windows SIM*" ou "*WSIM*"). C'est un outil totalement graphique.

Une *answer file* peut contenir diverses options ou informations d'installations, telles que partitionner un disque dur, formater un disque, la partition sur laquelle installer *Windows Vista*, la clé de *Windows Vista*, etc.

Chaque *answer file* est créée et associée pour une image bien précise.

Il est cependant possible (mais déconseillé) d'utiliser une *answer file* (créée par exemple pour *image1.WIM*) afin d'automatiser l'installation d'une autre image (par exemple *image2.WIM*).
Si une telle manipulation est effectuée, l'*answer file* peut faire référence à des *components* ou *packages* qui n'existent pas dans image2.WIM et si tel est le cas ces configurations seront ignorées.
Nous pourrons alors croire que certains composants ont étés paramétrés alors qu'en réalité ils ne l'ont pas été, à utiliser avec grande précaution !

Les configurations d'un fichier *Unattend.xml* sont organisées en 2 sections :

- ***Components***, se sont des parties du système d'exploitation qui indique les dossiers, ressources et configurations pour un dispositif *Windows* ou une partie spécifique d'un dispositif *Windows*.
  Certains de ces composants font référence à l'installation de *Windows*, ceci nous permet d'automatiser une installation *Windows* (par exemple créer une partition).
  Ces *components* peuvent être configurés durant l'une des 7 phases disponibles pour installer *Vista*, qui sont : *Windows PE*, *OfflineServicing*, *generalize*, *specialize*, *auditSystem*, *auditUser* et *OobeSystem* (ces différentes phases seront détaillées plus loin).

- ***Packages***, qui est un ensemble de fichiers fourni par *Microsoft*, dans le but de modifier certains dispositifs *Windows*.
  Ces *packages* peuvent être des *service packs*, des mises à jour de la sécurité, des *packs* de langue pour le système d'exploitation (français, anglais, etc.)

Avec un fichier *Unattend.xml*, on peut ajouter ou supprimer des *packages* à une image WIM. Ceci se fait durant la phase *offlineServicing*.

Remarques :

- Les *packages* désactivés ne sont pas retirés de l'image WIM !

- *Windows System Image Manager* est installé lors de l'installation de *Windows AIK*, disponible dans *BDD 2007 Beta 2*.
  Cette version de *WSIM* ne fonctionne pas avec des images de *Windows Vista RC1 Build 5728* ni *RC2 Build 5744*, elle fonctionne uniquement avec les images de *Windows Vista RC1 Build 5600*. Ceci m'a fait perdre une journée de travail.

A.2.2 Utilisation de Windows System Image Manager

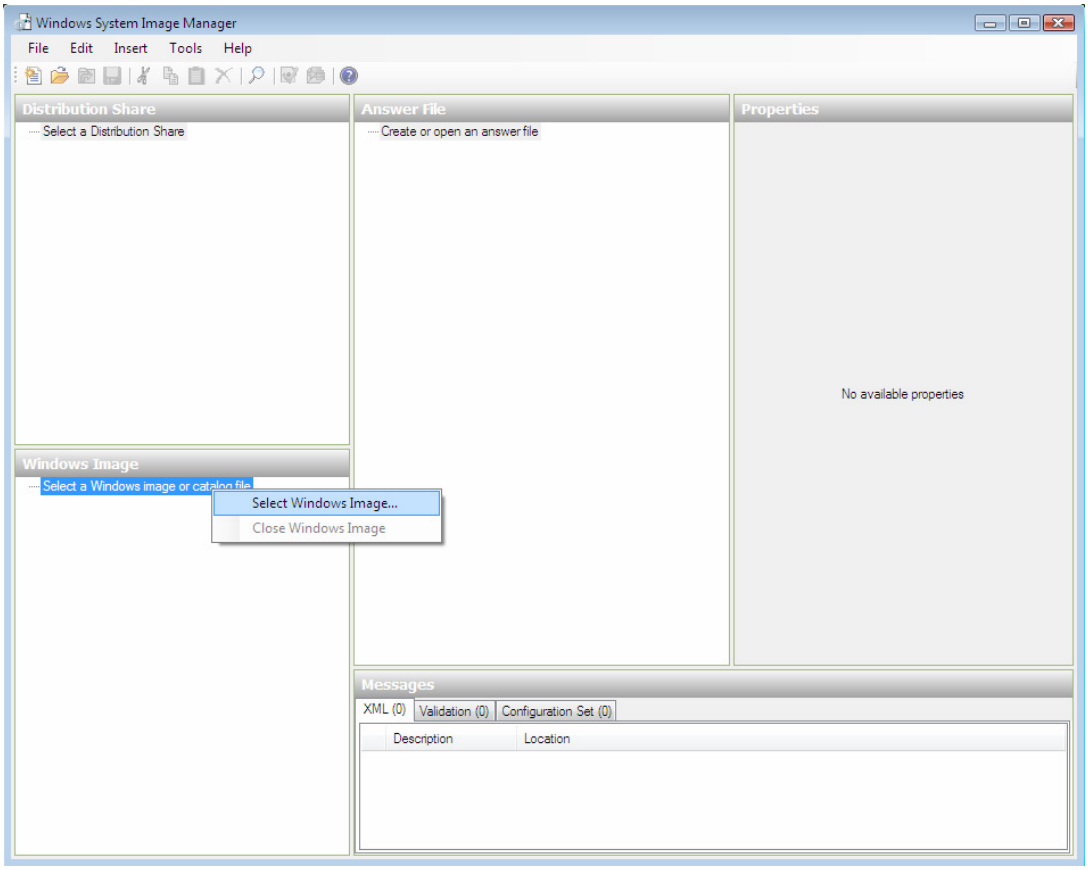
But : Explorer les composants et *packages* de l'image WIM de base de *Windows Vista RC1 Build 5600*, les modifier pour en faire une installation occupant le moins d'espace disque possible (enlever un maximum d'options et de programmes non nécessaires).
Paramétrer certains composants tels qu'*Internet Explorer* pour le laboratoire de l'école d'ingénieurs.
Automatiser l'installation de *Windows Vista*.

A.2.2.1 Ouvrir une image

Nous allons travailler avec l'image *WIM* de base de *Windows VistaRC1 Build 5600*, contenue dans le DVD d'installation de *Vista*.

Cette image se trouve à partir de la racine du DVD, dans le répertoire **`sources\install.wim`**

Copier cette image sur le disque dur, puis l'ouvrir avec *WSIM*.



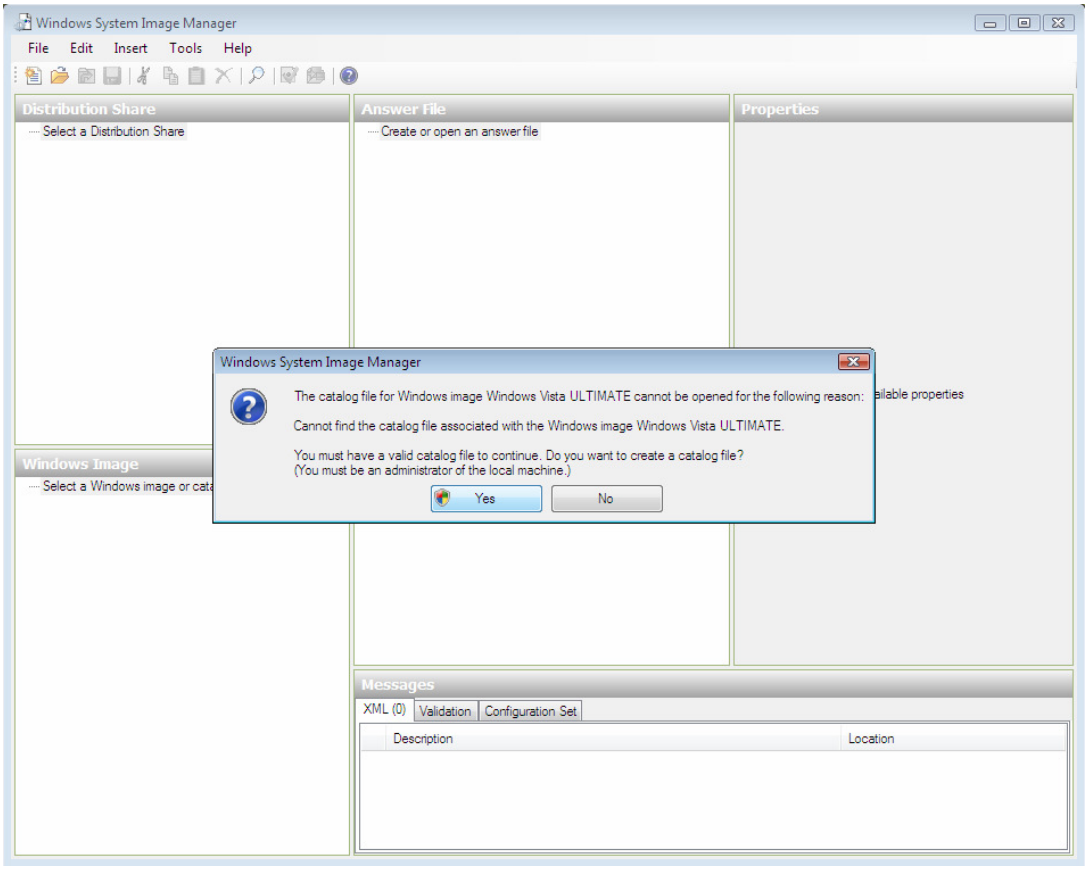
Remarques : Il y a plusieurs fenêtres différentes à l'intérieur de *WSIM*.

Parfois dans la suite de ce document, seulement certaines fenêtres seront affichées pour éviter de faire des captures d'écran trop grandes.

La fenêtre *Properties* change constamment de nom. Elle reprend le nom du *component* ou du *package* et rajoute ce nom devant *Properties* (par exemple son nom devient *Windows Foundation Properties* lorsque l'on souhaite voir les propriétés du *package Foundation*, voir page 6).

A.2.2.2 *Catalog File*

Autoriser *WSIM* à créer un *catalog file*, ce n'est en réalité qu'un parcourt de l'image afin de pouvoir répertorier les différents *components* et *packages* contenus dans cette image.



Cette opération nécessitera environ 1 minute, dépendant de la configuration *hardware* de la machine.

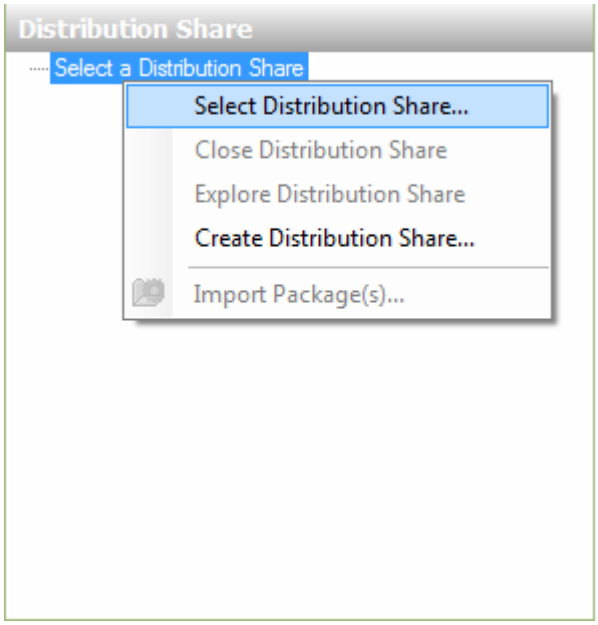
A.2.2.3 Ajouter un *package* à l'image

Il est relativement simple d'ajouter un *package* à l'image.

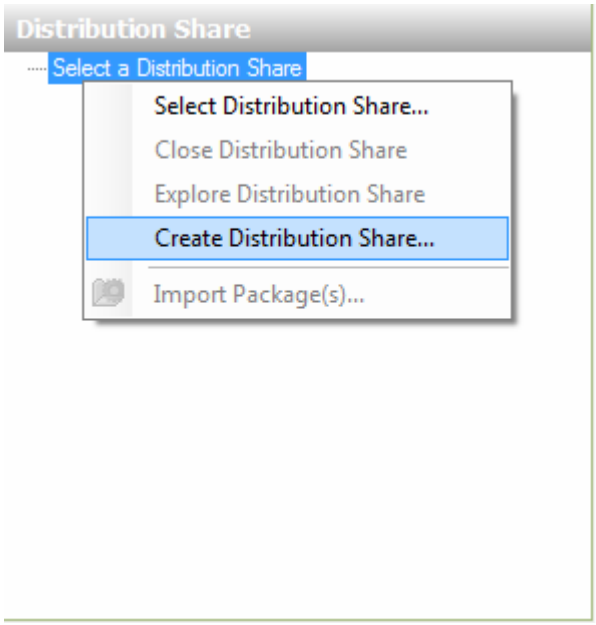
Selon *Microsoft* tous les prochains service packs de *Windows Vista* seront disponibles en tant que *packages*, ce qui permettra de les ajouter très facilement à une image à l'aide de *Windows System Image Manager*.

Pour cela, il faut créer ou sélectionner un dossier partagé de distribution (*Distribution Share*).

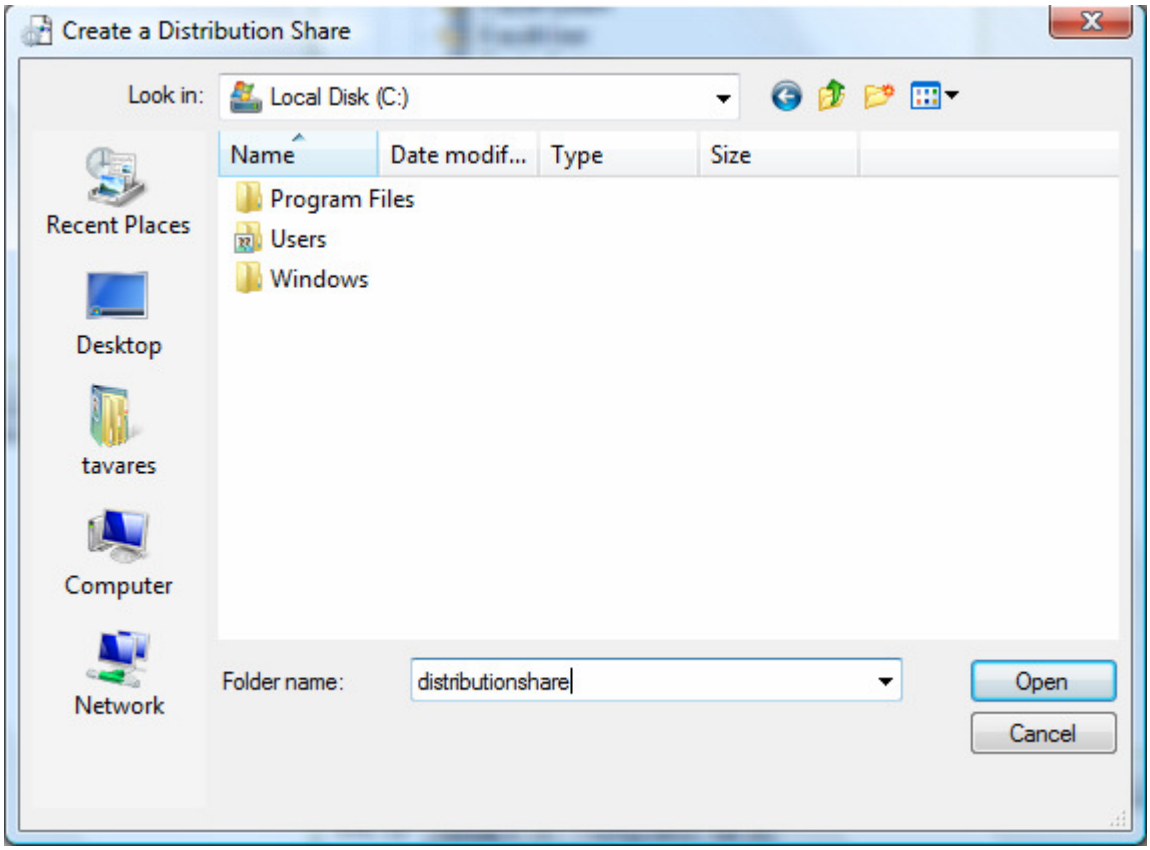
Pour en sélectionner un à l'aide de *Windows System Image Manager*, effectuer un clic droit sur *Select a Distribution Share*, puis sélectionner *Select a Distribution Share* :



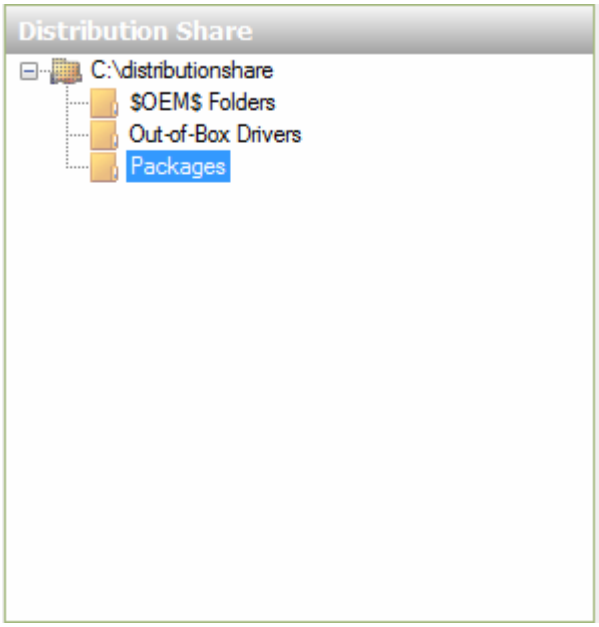
Dans le cas contraire en créer un en effectuant un clic droit sur *Select a Distribution Share*, puis *Create Distribution Share* :



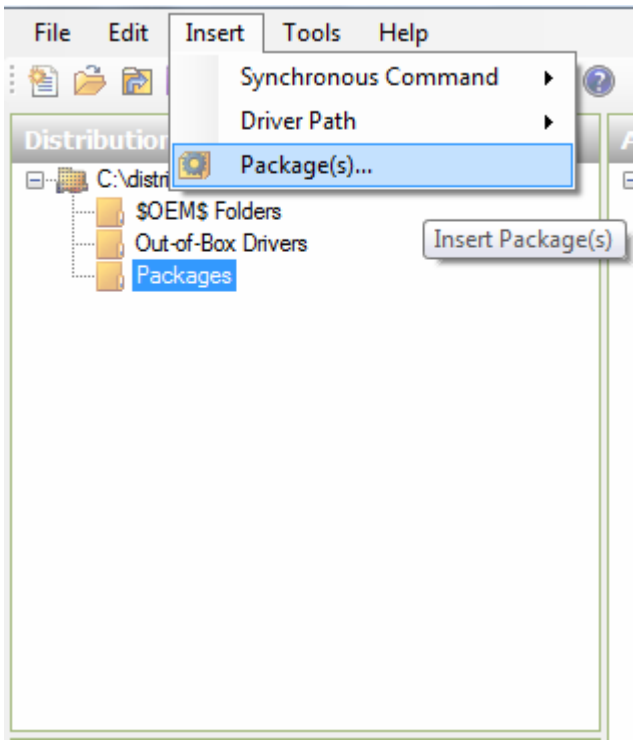
Choisir son emplacement dans la fenêtre suivante :



Windows System Image Manager créera alors la structure de fichiers dont il a besoin. Ensuite, il faudra tout simplement ajouter des *packages* dans le répertoire ***Packages***, puis sélectionner les *packages* voulus afin de les incorporer à l'image.



Pour ajouter un *package* à l'image, sélectionner le menu *Insert* puis *Package(s)…* :

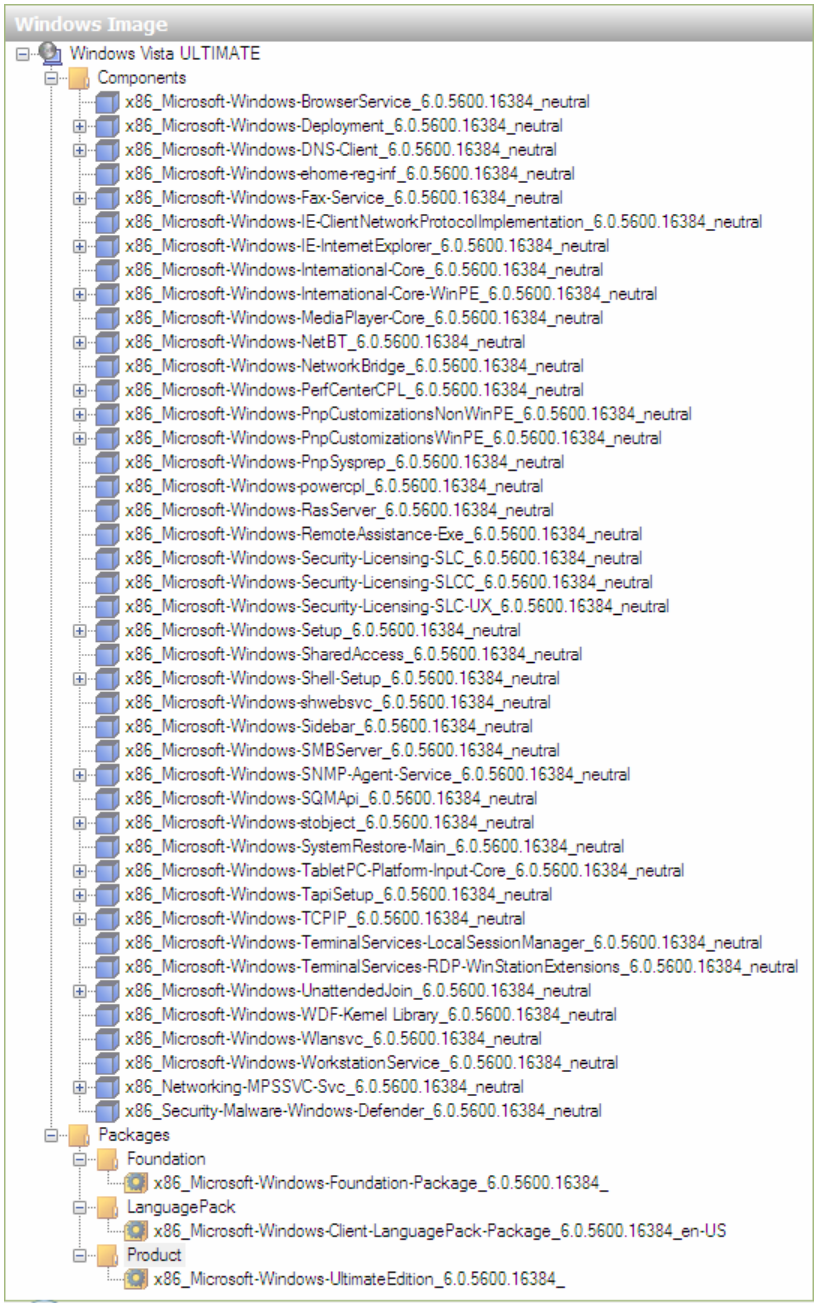


Choisir les *packages* à ajouter. Ceci est très pratique car il n'est pas nécessaire d'installer une image afin de lui appliquer un *service pack* puis re-capturer l'installation, il suffit simplement d'effectuer la procédure ci-dessus et le *package* sera installé en même temps que la future installation de l'image !

Remarque : Il est aussi possible d'ajouter des *hotfix* (mises à jour, un *service pack* est composé de plusieurs *hotfix*).

A.2.2.4 Parcourir les *components* et *packages* contenus au sein de l'image

Lorsque le *catalog file* est créé, il est possible d'explorer les *components* et *packages*.



On constate qu'il y a 43 *components* et seulement 3 *packages* dans l'image.

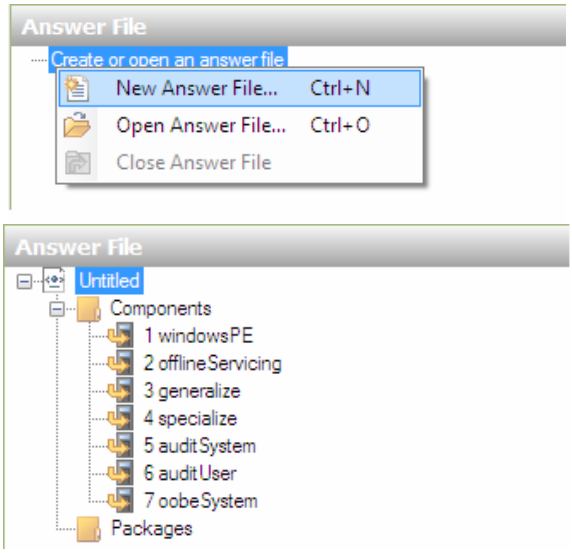
A.2.2.5 Analyse des divers *packages*

Dans *LanguagePack* (figure précédente) il peut y avoir plusieurs *packages* contenant différentes langues (français, anglais, etc.) de *Windows Vista*. Dans cette image uniquement l'anglais est présent. Il n'est pas possible de paramétrer ce *package*.

Dans *Product*, le *package* contient notre version de *Windows Vista*. Dans cette image il s'agit de la version *Ultimate*. Ce *package* ne peut être paramétré.

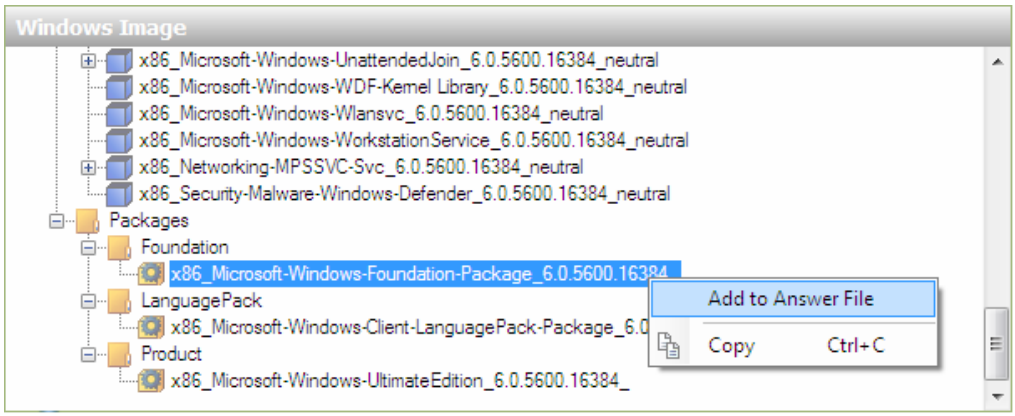
Le *package* dans *Foundation* contient divers outils qui peuvent être activés/désactivés. Ce package est le seul à pouvoir être paramétré.
Nous devons pour cela créer une *answer file*, qui sera sauvée sous le nom *Unattend.xml*. Ce fichier devra être utilisé plus tard pour effectuer le déploiement de *Windows Vista*.

Création d'une nouvelle *Answer File* :

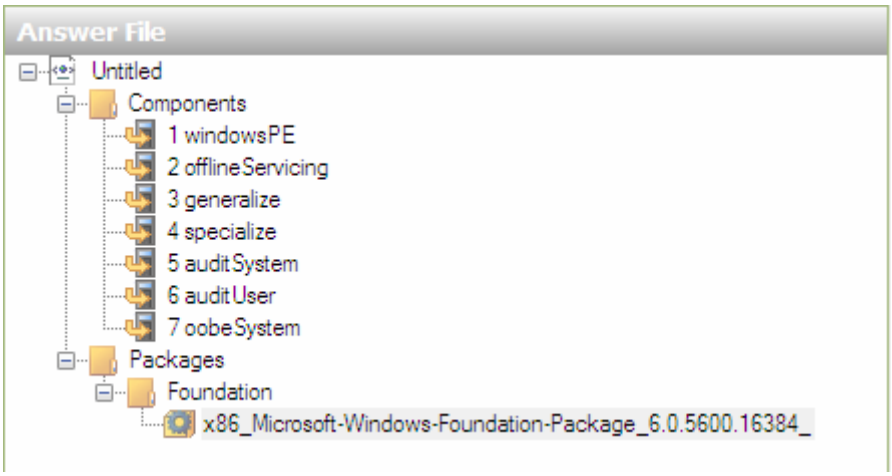


Paramétrer le *package* dans *Foundation* :

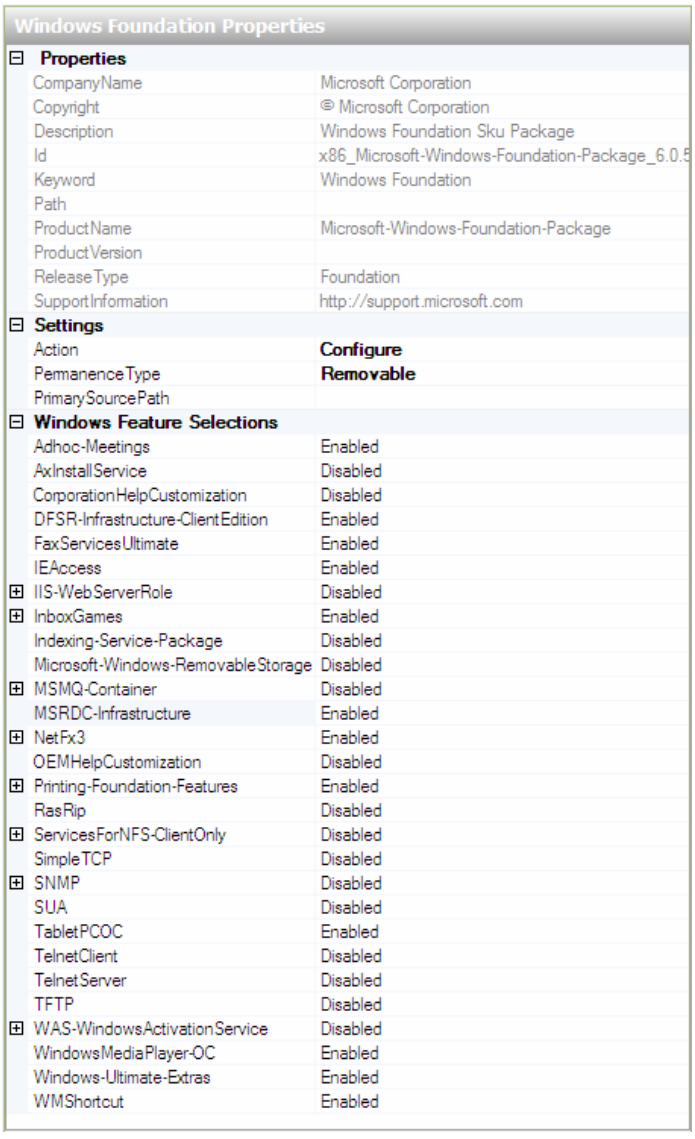
Ajouter ce *package* à notre *Answer File*, afin de pouvoir le paramétrer.



Dans la fenêtre *Answer File* on peut vérifier que le package a bien été ajouté.



Dans la fenêtre *Windows Foundation Properties*, on peut voir les différents programmes (ou options) qui sont activés/désactivés par défaut dans notre image *Windows Vista*.

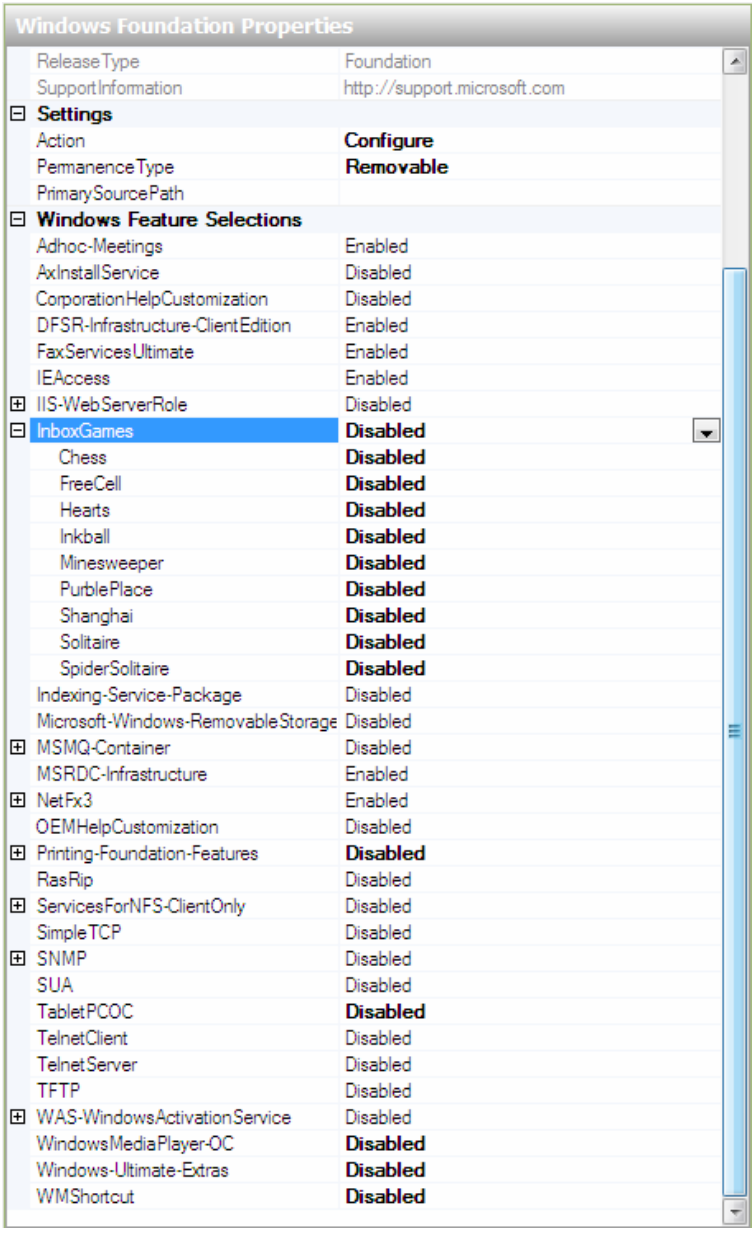


Lorsque les valeurs par défaut son modifiées, les nouvelles valeurs seront marquées en gras.

Paramétrer ce *package* :

Désactiver les programmes (ou options) non nécessaires, tels que les jeux (solitaire et autres), *tablet PC*, *Windows Media Player*, *Windows Ultimate Extras*, *Windows Media Shortcut* (raccourci *Windows Media*).

Remarque : Ces services ne sont pas retirés de l'image WIM, ils sont uniquement désactivés. En d'autres termes ils ne seront pas installés lors du déploiement de l'image, ceci à condition de faire le déploiement avec le fichier XML que *WSIM* nous crée.



A.2.2.6 Analyse des divers *components*

Les *components* les plus intéressants sont (voir figure en page 16) :

- **x86_Microsoft-Windows-Setup_6.0.5600.16384_neutral**, permet de gérer les disques durs (tels que formater, partitionner), donner l'emplacement du disque dur (partition) sur laquelle l'image doit être installée, etc.

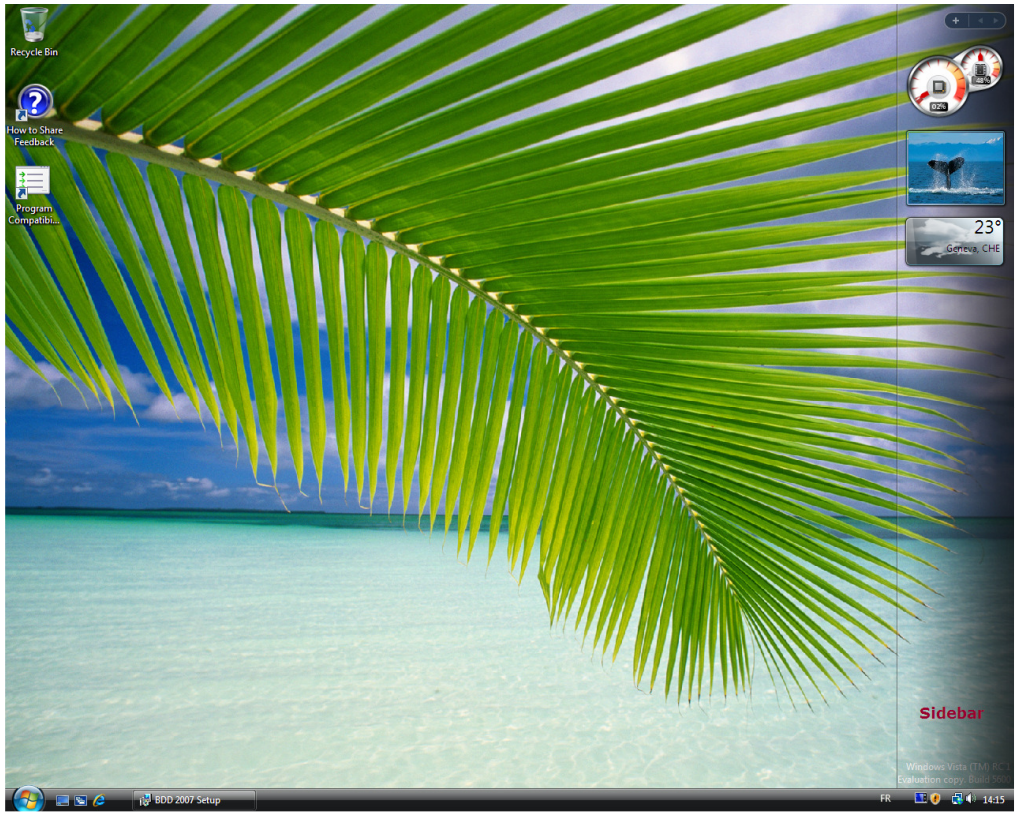
- **x86_Microsoft-Windows-Shell-Setup_6.0.5600.16384_neutral**, permet d'effectuer un *AutoLogon* (se connecter à un compte automatiquement), gérer les comptes utilisateurs, définir un mot de passe administrateur, etc.

- **x86_Microsoft-Windows-IE-InternetExplorer_neutral**, permet de configurer *Internet Explorer* en lui définissant une *Home Page* (page de démarrage), en activant le blocage de *popups* (fenêtres publicitaires), désactiver les sons, etc.

- **x86_Microsoft-Windows-TCPIP_6.0.5600.16384_neutral**, permet de configurer les paramètres TCP/IP

- **x86_Microsoft-Windows-DNS-Client_6.0.5600.16384_neutral**, permet de configurer les paramètres DNS

- **x86_Microsoft-Windows-Sidebar_6.0.5600.16384_neutral**, permet de désactiver la *Sidebar* (barre par défaut où l'on peut afficher des gadgets tels que la consommation instantanée du CPU, occupation de la RAM… Cette barre se lance à droite de l'écran).



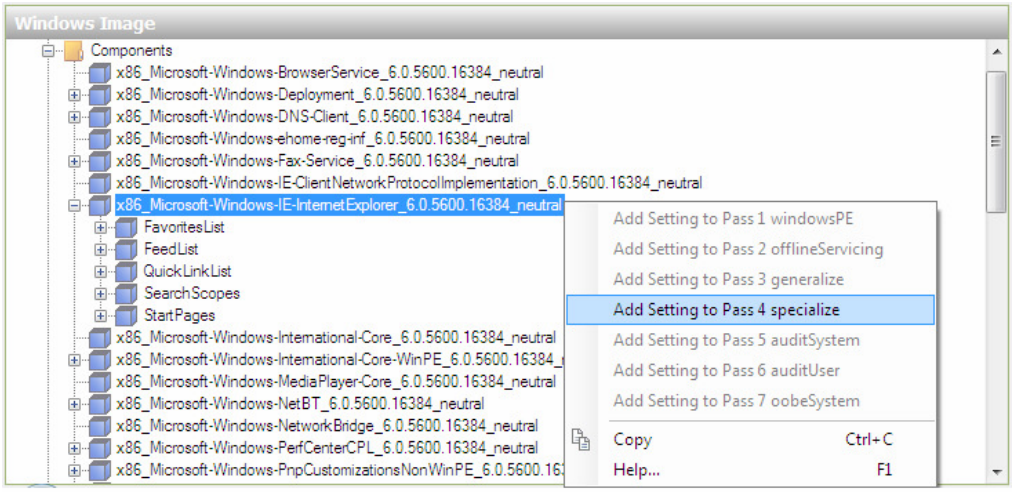
- **x86_Microsoft-Windows-RemoteAssistance-Exe_neutral**, permet de paramétrer l'assistance à distance. Ce service peut aussi être désactivé comme nous le verrons par la suite.

- **`x86_Networking-MPSSVC-Svc_6.0.5600.16384_neutral`**, permet d'activer/désactiver le *firewall* pour certains profils (profils appartenant à un domaine, profils privés ou profils publiques), activer/désactiver la création de *logs* (lorsque un paquet est perdu, lors d'une connexion réussie), activer/désactiver les services *Statefull FTP* (*File Transfert Protocol*) et *Statefull PPTP* (*Point To Point Tunneling Protocol*)

A.2.2.7 Paramétrer les divers *components*

A.2.2.7.1 Configurer *Internet Explorer* (*IE*)

Ajouter le *component* **`x86_Microsoft-Windows-IE-InternetExplorer_neutral`** à notre *Answer File*.



Lors de l'ajout d'un *component* à une *Answer File*, chaque *component* doit être ajouté lors d'une des 7 phases d'installation de *Windows*.

Une seule phase est disponible pour ajouter le *component Internet Explorer* (*Pass 4 specialize*).

Parfois plusieurs phases sont possibles, il faudra choisir celle qui convient le mieux à nos besoins.

Dans la fenêtre *Microsoft-Windows-IE-InternetExplorer Properties*, effectuer les modifications suivantes :

1) ***BlockPopups*** => ***yes***
Active le blocage de *Popups*

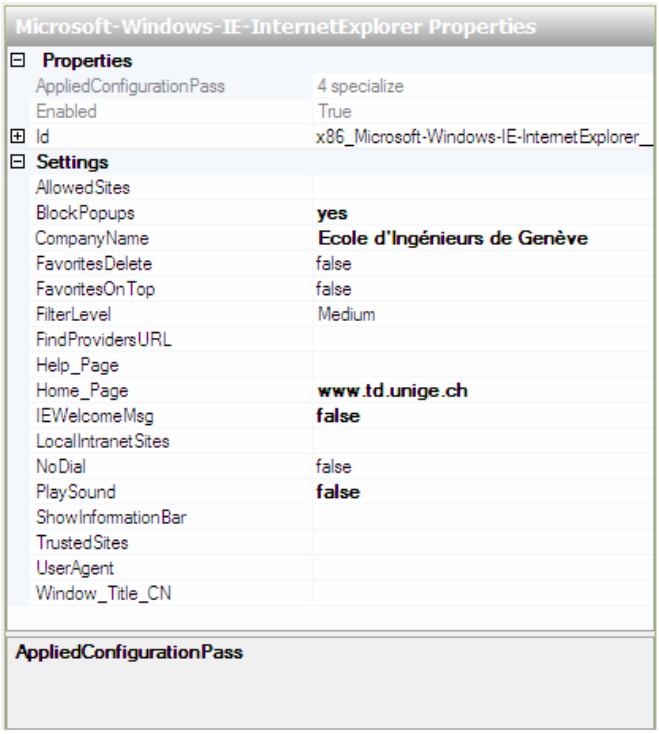
2) ***CompanyName*** => **Ecole d'Ingénieurs de Genève**
Nom d'entreprise pour *Internet Explorer*.

3) ***Home_Page*** => **www.td.unige.ch**
Page d'accueil d'*Internet Explorer*.

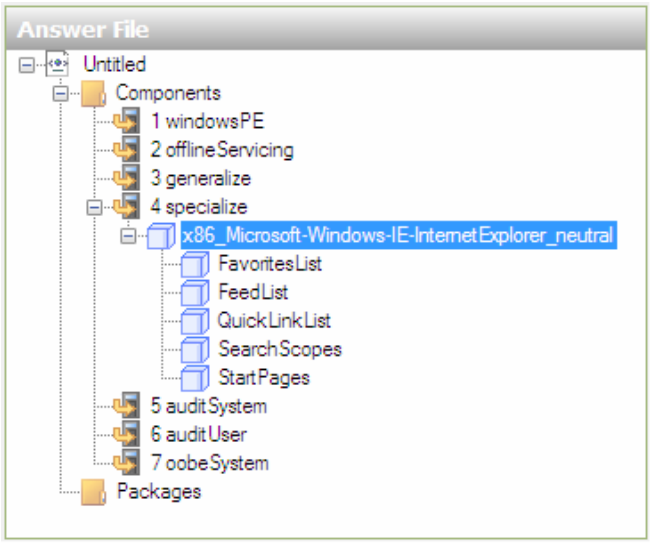
4) ***IEWelcomeMsg*** => ***false***
Désactive le message d'accueil qui est affiché lorsque l'utilisateur démarre pour la première fois *Internet Explorer*.

5) ***PlaySound*** => ***false***

Désactive les sons lorsque *Internet Explorer* bloque un *pop-up*

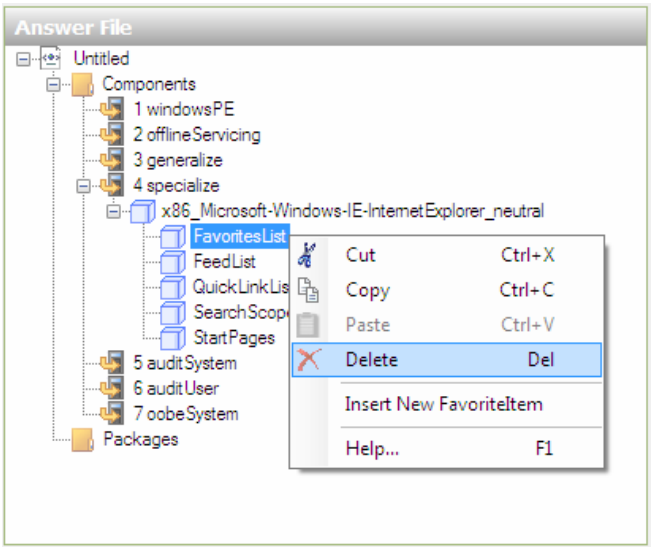


Remarque : Lors de l'ajout de ce *component*, divers "sous *components*" qui lui sont associés sont aussi ajoutés.



Ces "sous *components*" sont dans notre cas : *FavoritesList*, *FeedList*, *QuickLinkList*, *SearchScopes*, *StartPages*.

S'ils ne sont pas configurés par la suite, il faut les retirer de notre *Answer File*, pour cela il suffit de les sélectionner puis de faire un *Delete*.

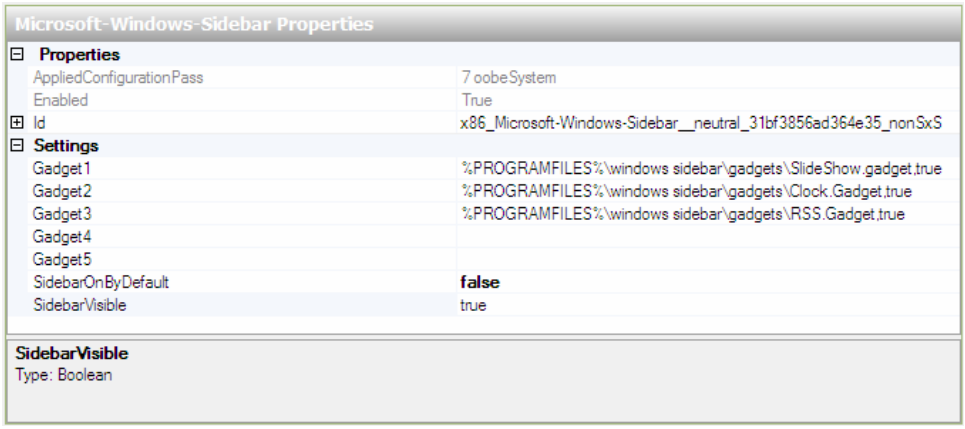


A.2.2.7.2 Désactiver la *Sidebar*

Ajouter le *component* **`x86_Microsoft-Windows-Sidebar_6.0.5600.16384_neutral`** à notre *Answer File*.

Dans la fenêtre *Microsoft-Windows-Sidebar Properties*, effectuer la modification suivante :

1) ***SidebarOnByDefault*** => ***false***
Désactive la *Sidebar*



A.2.2.7.3 Configurer l'assistance à distance (*RemoteAssistance*)

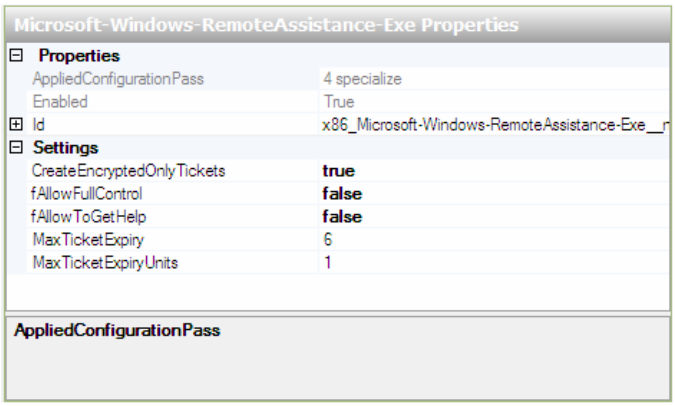
Ajouter le *component* **`x86_Microsoft-Windows-RemoteAssistance-Exe_neutral`** à notre *Answer File*.

Dans la fenêtre *Microsoft-Windows-RemoteAssistance Properties*, effectuer les modifications suivantes :

1) ***CreateEncryptedOnlyTickets*** => ***true***
Permet uniquement la création de tickets chiffrés.

2) ***fAllowFullControl*** => ***false***
Désactive le contrôle total de la machine lors d'une assistance à distance.

3) ***fAllowToGetHelp*** => ***false***
Un utilisateur ne pourra pas demander d'aide à distance.



Remarque : Cette configuration améliore la sécurité.

A.2.2.7.4 Configurer le *component* du *firewall* MPSSVC

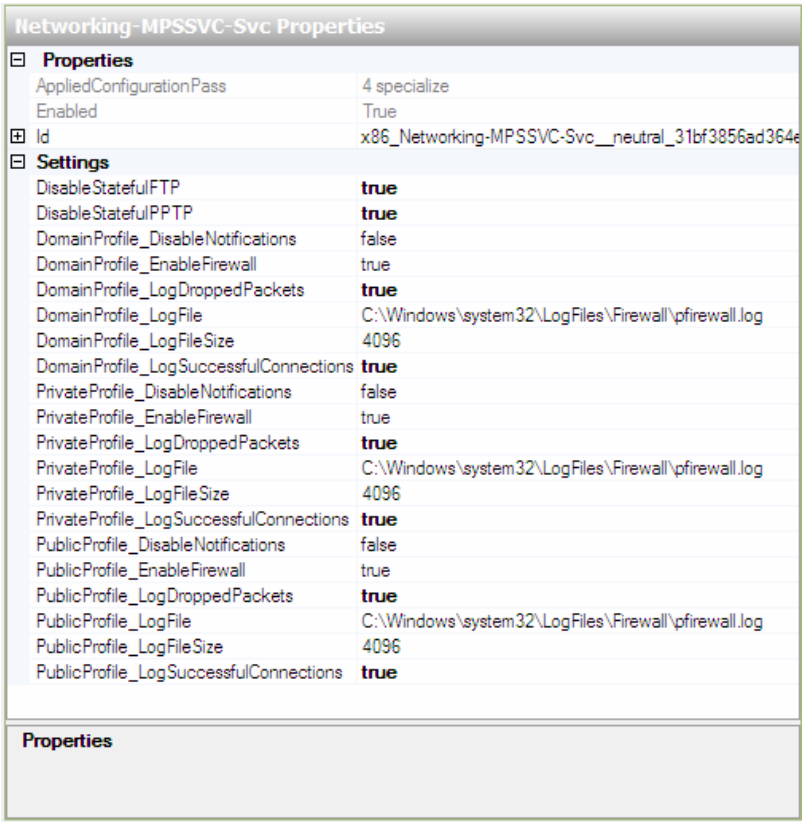
Ajouter le *component* **`x86_Networking-MPSSVC-Svc_6.0.5600.16384_neutral`** à notre *Answer File*.

Dans la fenêtre *Networking-MPSSVC-Svc Properties*, effectuer les modifications suivantes :

1) ***DisableStatefulFTP*** => ***true***
Désactive les services de type *Statefull FTP*

2) ***DisableStatefulPPTP*** => ***true***
Désactive les services de type *Statefull PPTP*

3) ***DomainProfile_LogDroppedPackets*** => ***true***
DomainProfile_LogSuccessfulConnections => ***true***
PrivateProfile_LogDroppedPackets => ***true***
PrivateProfile_LogSuccessfulConnections => ***true***
PublicProfile_LogDroppedPackets => ***true***
PublicProfile_LogSuccessfulConnections => ***true***
Active tous les *logs*



Remarque : Le *firewall* est activé par défaut, il n'est donc pas nécessaire d'effectuer des modifications à ce niveau, sauf si l'on désire désactiver le *firewall*.

A.2.2.8 Automatiser l'installation de *Windows Vista*

Il est possible d'automatiser l'installation de *Windows Vista* à l'aide des components **`x86_Microsoft-Windows-Setup_6.0.5600.16384_neutral`** et **`x86_Microsoft-Windows-Shell-Setup_6.0.5600.16384_neutral`**, ceci afin d'installer *Windows* le plus rapidement et avec le moins d'interactions humaines possible

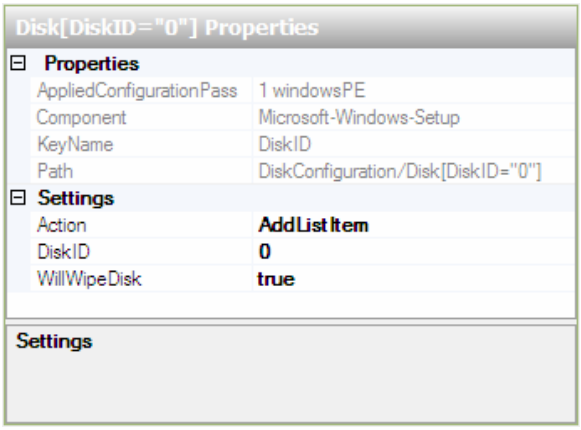
A.2.2.8.1 Créer une partition sur le disque dur

Ajouter le *component* **`x86_Microsoft-Windows-Setup_6.0.5600.16384_neutral\DiskConfiguration\Disk`** à notre *Answer File*.

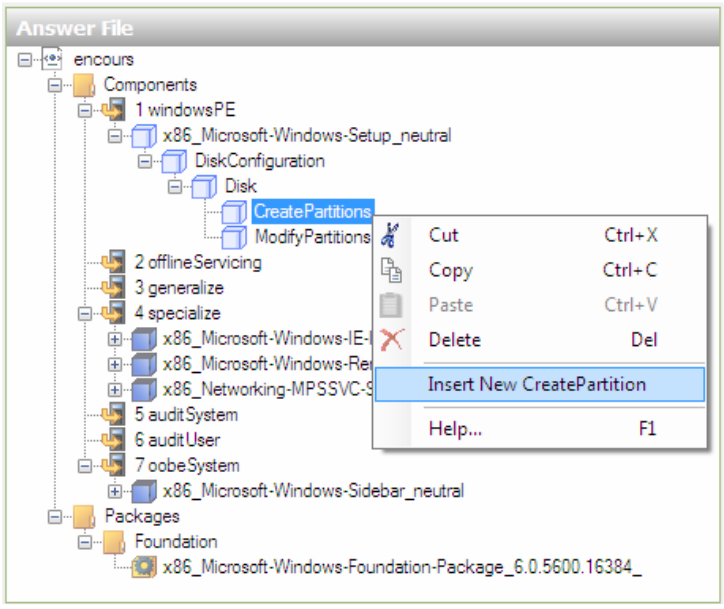
Dans la fenêtre *Disk Properties*, effectuer les modifications suivantes :

1) ***DiskID*** => **0**
Défini sur quel disque dur la partition sera créée.
Lorsqu'il n'y a qu'un disque dur, sont *DiskID* vaut 0, s'il y en a deux l'un autre le *DiskID* 0 et l'autre *DiskID* 1, et ainsi de suite.
Ces *DiskID* sont bien évidemment uniques (un pour chaque disque).

2) ***WillWipeDisk*** => ***true***
Toutes les partitions du disque seront effacées avant d'ajouter de nouvelles configurations au disque dur.



Dans la fenêtre *Answer File*, ouvrir l'onglet *Disk*, sélectionner *CreatePartition* puis avec clic droit, sélectionner *Insert New CreatePartition*.

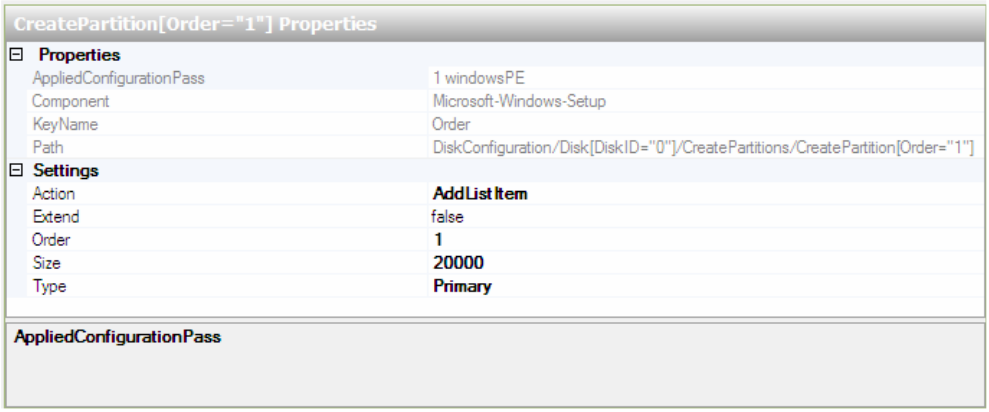


Dans la fenêtre *CreatePartition Properties*, affecter les valeurs suivantes :

1) ***Order*** => **1**
Spécifie l'ordre de création des partitions. Ici nous n'avons qu'une seule partition, cependant il faut tout de même affecter une valeur à cette option.

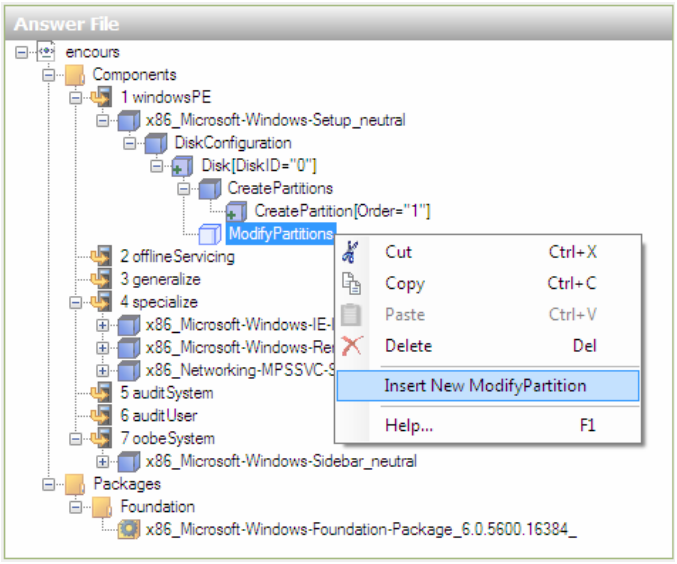
2) ***Size*** => **20000**
Définit la taille de la partition à créer, elle sera de 20gigas dans notre cas.

3) ***Type*** => ***Primary***
La partition sera de type *Primary* (primaire).
En règle générale les systèmes d'exploitation doivent être installés sur des partitions de type *Primary*, sauf si plusieurs systèmes d'exploitation sont installés sur le même ordinateur. Si tel est le cas, le premier système d'exploitation à installer nécessitera une partition de type *Primary*, les autres systèmes d'exploitation pourront être installés sur des partitions étendues ou secondaires.



A.2.2.8.2 Formater la partition créée

Dans la fenêtre *Answer File*, effectuer un clic droit sur *ModifyPartitions* puis sélectionner *Insert New ModifyPartition*.



Dans la fenêtre *ModifyPartiton Properties*, affecter les valeurs suivantes :

1) ***Active*** => ***true***
La partition sera une partition de type active après formatage.

2) ***Extend*** => ***false***
La partition ne sera pas de type *Extend* (étendue).

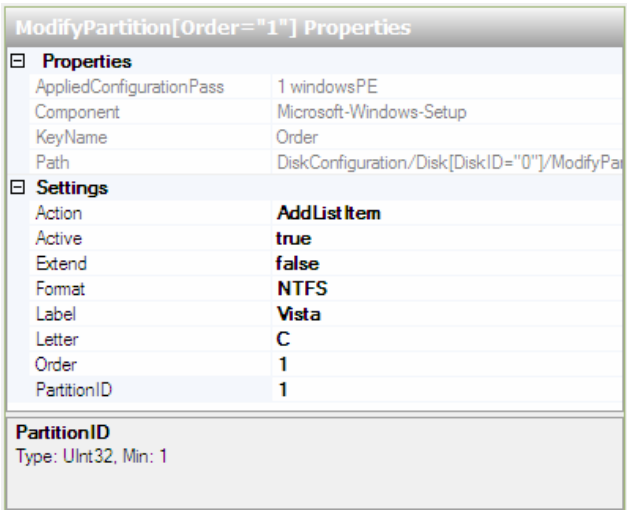
3) ***Format*** => **NTFS**
Formatera la partition en NTFS.

4) ***Label*** => ***Vista***
Le nom de la partition sera : Vista.

5) ***Letter*** => **C**
Affecte une lettre de disque à la partition créée.

6) ***Order*** => **1**
Défini l'ordre dans lequel les partitions seront modifiées

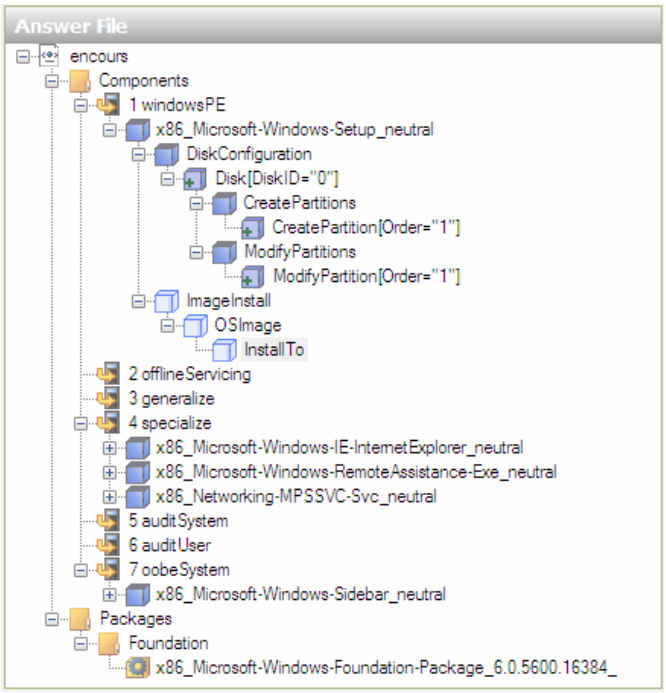
7) ***PartitionID*** => **1**
Spécifie le numéro d'identification de la partition qui doit être modifiée.
La première partition sur le disque possède la valeur 1, la deuxième possède la valeur 2, et ainsi de suite.



A.2.2.8.3 Spécifier une partition pour l'installation de *Windows Vista*

Nous alors ici définir la partition qui vient d'être créée comme destination pour l'installation de notre *Windows*.

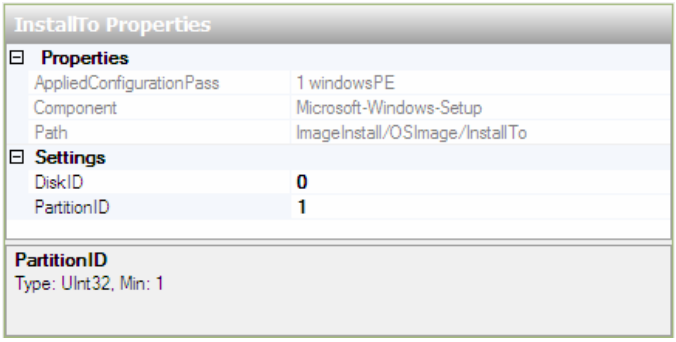
Ajouter le *component* **`x86_Microsoft-Windows-Setup_6.0.5600.16384_neutral\ImageInstall\OSImage\InstallTo`** à notre *Answer File*.



Dans la fenêtre *InstallTo Properties*, affecter les valeurs suivantes :

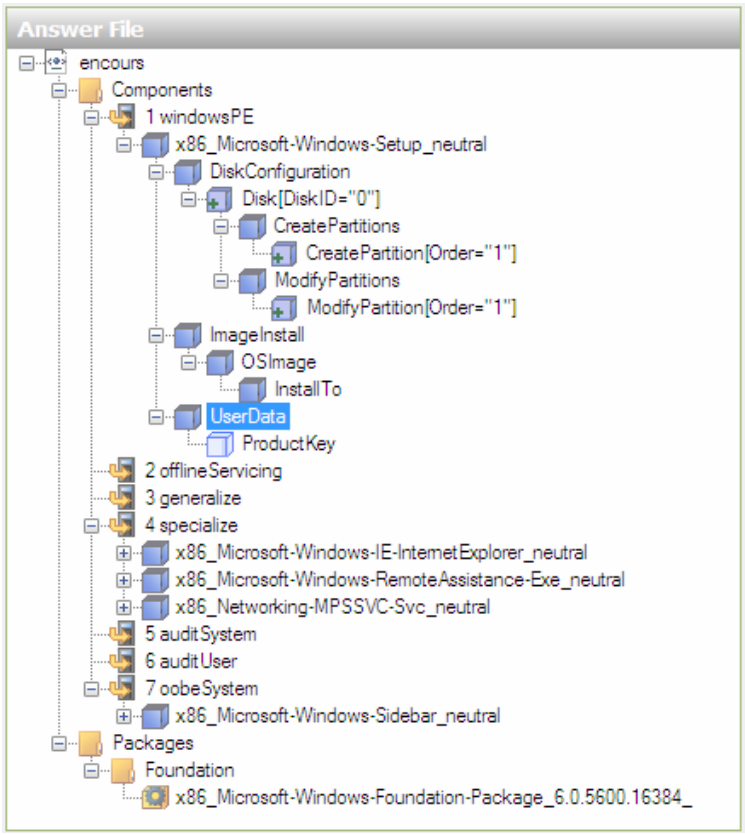
1) ***DiskID*** => **0**
L'installation va s'effectuer sur le disque avec l'identificateur 0.

2) ***PartitionID*** => **1**
La partition qui accueillera l'installation de Windows aura l'identificateur 1.



A.2.2.8.4 Entrer la clef de *Windows Vista* ainsi que les données d'enregistrement

Ajouter le *component* **`x86_Microsoft-Windows-Setup_6.0.5600.16384_neutral\UserData`** à notre *Answer File*.

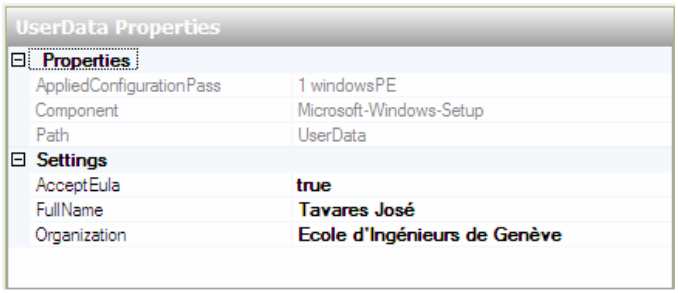


Dans la fenêtre *UserData Properties*, entrer les valeurs suivantes :

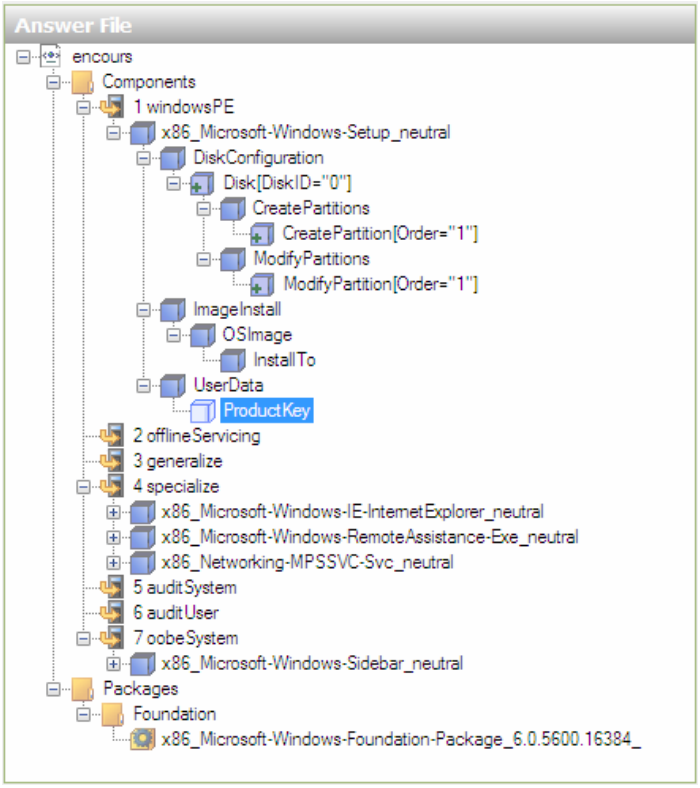
1) ***AcceptEula*** => ***true***
Lors d'une installation *Windows*, *Microsoft* demande à un certain moment d'accepter les termes de la licence produit. En affectant *AcceptEula* à *true*, ces termes licence sont automatiquement acceptés lors de l'installation, ce qui permet donc d'automatiser l'installation.

2) ***FullName*** => **Tavares José**
Entrer le nom complet de la personne ayant acheté *Windows*.

3) ***Organization*** => **Ecole d'ingénieurs de Genève**
Entrer le nom complet de l'entreprise.



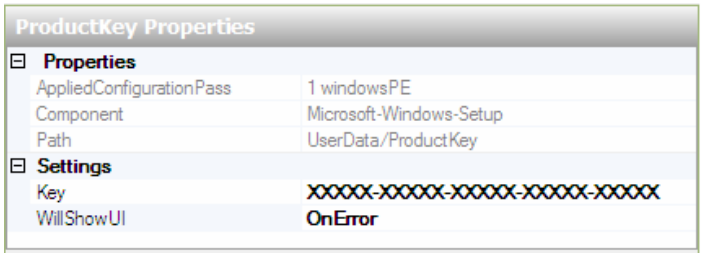
Dans la fenêtre *Answer File*, ouvrir *UserData*, sélectionner *Product Key*



Dans la fenêtre *ProductKey Properties*, affecter les valeurs suivantes :

1) ***Key*** => **XXXXX-XXXXX-XXXXX-XXXXX-XXXXX**
Entrer votre clef de *Windows Vista*.

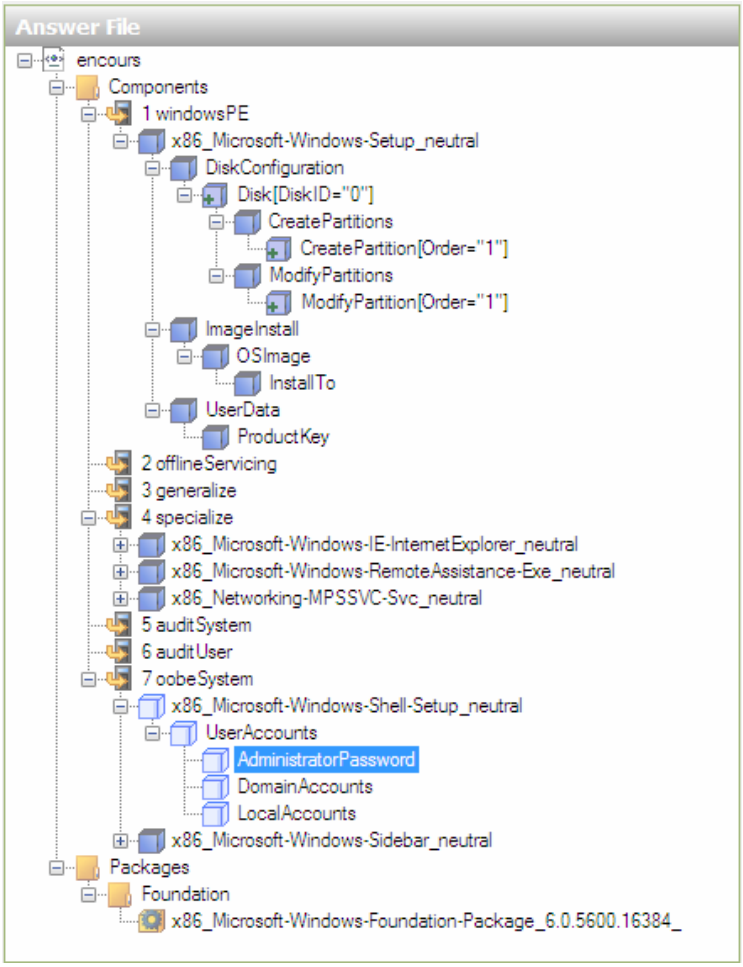
2) ***WillShowUI*** => ***OnError***
Spécifie dans quelles circonstances sera affichée l'interface graphique demandant la clef. Ici elle sera uniquement affichée lors d'une erreur.



A.2.2.8.5 Définir le mot de passe administrateur

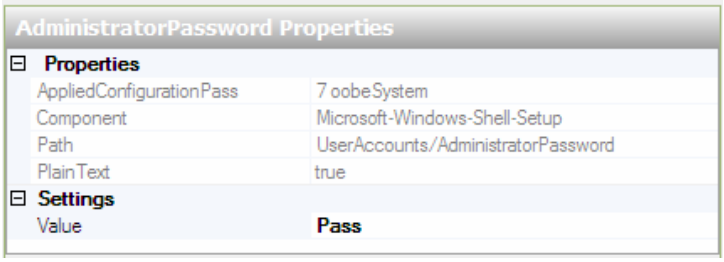
Ajouter le *component* **`x86_Microsoft-Windows-Shell-Setup_6.0.5600.16384_neutral\UserAccounts`** à notre *Answer File*.

Dans notre *Answer File*, ouvrir *UserAccounts* puis sélectionner *AdministratorPassword*.



Dans la fenêtre *AdministratorPassword Properties*, entrer les valeurs suivantes :

1) ***Value*** => **Pass**
Entrer le mot de passe Administrateur de votre choix.
Veuillez à entrer un mot de passe compliqué, avec des majuscules, minuscules, chiffres et caractères spéciaux afin de rendre ce mot de passe plus difficile à trouver pour un hacker.

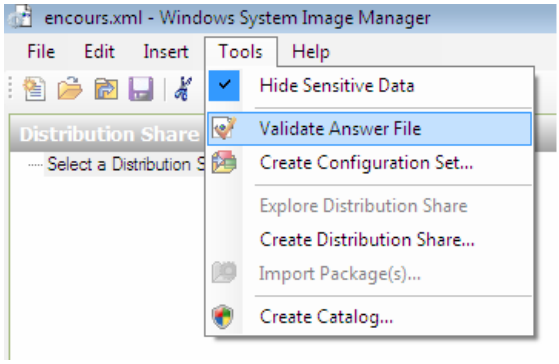


Remarque : Le mot de passe apparaît en clair dans *Windows System Image Manager* cependant, à l'intérieur du fichier xml que l'on va créer à l'aide de *WSIM*, le mot de passe sera chiffré !

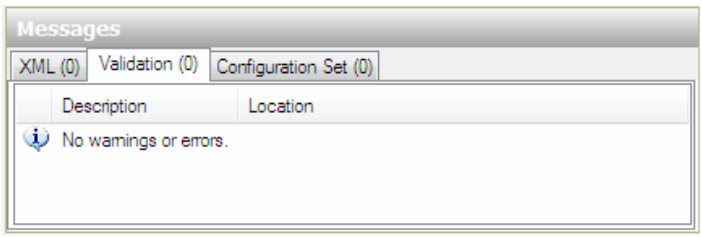
Il est aussi possible de définir des comptes utilisateurs locaux ou encore des comptes utilisateurs faisant partie d'un domaine.

Lorsque nous avons terminé d'utiliser *WSIM* pour personnaliser notre installation, il reste à **vérifier** que le **fichier xml** créé soit valide.

Pour ce faire, sélectionner ***Tools*** dans le menu de WSIM, puis ***Validate Answer File***.



Le message suivant est affiché par WSIM si le fichier xml est correct :



Sauver le fichier dans un emplacement du disque dur.

A.2.2.8.6 Paramétrer les options réseau

Pour paramétrer les options réseau, il faut ajouter le component **`x86_Microsoft-Windows-TCPIP_6.0.5600.16384_neutral`** à notre *anser file*.

Cependant, la configuration de ce *component* m'a posé de gros problèmes, je n'ai pas réussi à la faire fonctionner, c'est peut-être d'ailleurs pour cela que je n'ai trouvé aucune documentation sur ce *component* venant de *Microsoft*…

La seule documentation que j'ai pu utiliser est celle disponible dans l'aide de *Windows AIK* et qui n'est pas du tout claire.

Dans ce *component*, *Microsoft* ne parle pas de *gateway* (route par défaut), ni de *subnet mask* (masque de sous-réseau) mais de *NextHopAddress* (à priori le *gateway* en comparant avec les systèmes *Linux*…), de *Prefix* (*subnet mask* ?), de *Key* (identificateur de configuration), aussi de la *MAC address* de la carte réseau, ou encore nom de la connexion réseau), de *Value* (*Adresse IP*)…

Tous ces termes ne sont pas très clairs et mal expliqués dans l'aide de *Windows AIK*.

J'ai effectué plusieurs configurations différentes, j'ai même essayé un exemple de configuration disponible dans l'aide *Windows AIK*, aucune de ces méthodes n'a fonctionné (de plus il y a des erreurs dans l'exemple disponible dans l'aide *Windows AIK*…)

Espérons que tout ceci soit corrigé dans le futur.

Pour l'instant, si l'on souhaite configurer les options réseau de plusieurs postes dans une entreprise, il vaut mieux utiliser un serveur *DHCP* plutôt que ce *component*.

Voici le fichier XML créé avec les différentes opérations décrites ci-dessus :

```
<?xml version="1.0" encoding="utf-8" ?>
- <unattend xmlns="urn:schemas-microsoft-com:unattend">
- <servicing>
- <package action="configure">
  <assemblyIdentity name="Microsoft-Windows-Foundation-Package"
    version="6.0.5600.16384" processorArchitecture="x86"
    publicKeyToken="31bf3856ad364e35" language="" />
  <selection name="InboxGames" state="false" />
    </package>
    </servicing>
- <settings pass="specialize">
- <component name="Microsoft-Windows-IE-InternetExplorer"
    processorArchitecture="x86" publicKeyToken="31bf3856ad364e35"
    language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <BlockPopups>yes</BlockPopups>
  <CompanyName>Ecole d'Ingenieurs de Geneve</CompanyName>
  <Home_Page>www.td.unige.ch</Home_Page>
  <IEWelcomeMsg>false</IEWelcomeMsg>
  <PlaySound>false</PlaySound>
    </component>
- <component name="Microsoft-Windows-RemoteAssistance-Exe"
    processorArchitecture="x86" publicKeyToken="31bf3856ad364e35"
    language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <fAllowFullControl>false</fAllowFullControl>
  <fAllowToGetHelp>false</fAllowToGetHelp>
  <CreateEncryptedOnlyTickets>true</CreateEncryptedOnlyTickets>
    </component>
- <component name="Networking-MPSSVC-Svc" processorArchitecture="x86"
    publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DisableStatefulFTP>true</DisableStatefulFTP>
  <DisableStatefulPPTP>true</DisableStatefulPPTP>
  <DomainProfile_LogDroppedPackets>true</DomainProfile_LogDroppedPackets>

    <DomainProfile_LogSuccessfulConnections>true</DomainProfile_LogSuccessfulConnecti
    ons>
  <PrivateProfile_LogDroppedPackets>true</PrivateProfile_LogDroppedPackets>

    <PrivateProfile_LogSuccessfulConnections>true</PrivateProfile_LogSuccessfulConnectio
    ns>
  <PublicProfile_LogDroppedPackets>true</PublicProfile_LogDroppedPackets>
  <PublicProfile_LogSuccessfulConnections>true</PublicProfile_LogSuccessfulConnections>
    </component>
    </settings>
- <settings pass="oobeSystem">
- <component name="Microsoft-Windows-Sidebar" processorArchitecture="x86"
    publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
  <SidebarOnByDefault>false</SidebarOnByDefault>
    </component>
- <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="x86"
    publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <UserAccounts>
- <AdministratorPassword>

    <Value>UABhAHMAcwBBAGQAbQBpAG4AaQBzAHQAcgBhAHQAbwByAFAAYQBzA
    HMAdwBvAHIAZAA=</Value>
  <PlainText>false</PlainText>
    </AdministratorPassword>
    </UserAccounts>
    </component>
    </settings>
- <settings pass="windowsPE">
- <component name="Microsoft-Windows-Setup" processorArchitecture="x86"
    publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
    xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <DiskConfiguration>
- <Disk wcm:action="add">
- <CreatePartitions>
- <CreatePartition wcm:action="add">
  <Order>1</Order>
  <Size>20000</Size>
  <Type>Primary</Type>
    </CreatePartition>
    </CreatePartitions>
- <ModifyPartitions>
- <ModifyPartition wcm:action="add">
  <Active>true</Active>
  <Extend>false</Extend>
  <Format>NTFS</Format>
  <Label>Vista</Label>
  <Letter>C</Letter>
  <Order>1</Order>
  <PartitionID>1</PartitionID>
    </ModifyPartition>
    </ModifyPartitions>
  <DiskID>0</DiskID>
  <WillWipeDisk>true</WillWipeDisk>
    </Disk>
    </DiskConfiguration>
- <ImageInstall>
- <OSImage>
- <InstallTo>
  <DiskID>0</DiskID>
  <PartitionID>1</PartitionID>
    </InstallTo>
    </OSImage>
    </ImageInstall>
- <UserData>
- <ProductKey>
```

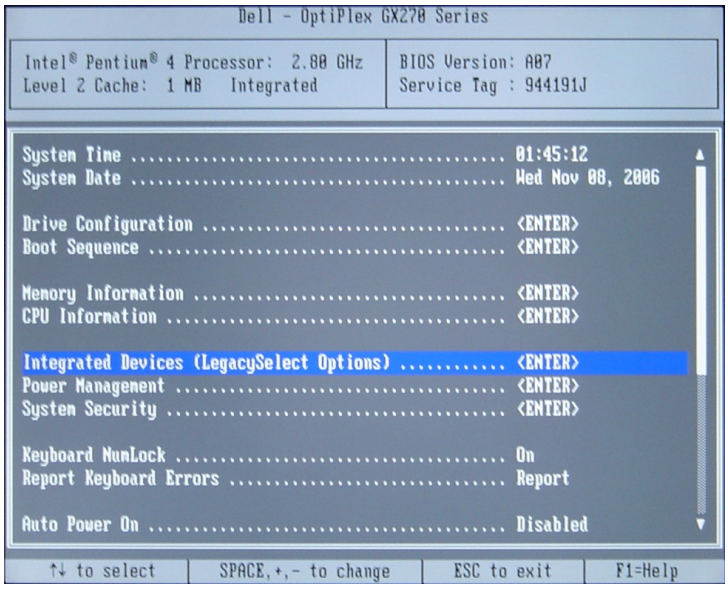
```
<Key>XXXXX-XXXXX-XXXXX-XXXXX-XXXXX</Key>
<WillShowUI>OnError</WillShowUI>
  </ProductKey>
<AcceptEula>true</AcceptEula>
<FullName>Tavares José</FullName>
<Organization>Ecole d'Ingénieurs de Genève</Organization>
  </UserData>
  </component>
  </settings>
<cpi:offlineImage cpi:source="wim:d:/5600bddwaik.wim#Vista 5600 avec BDD2007
   et WAIK" xmlns:cpi="urn:schemas-microsoft-com:cpi" />
  </unattend>
```

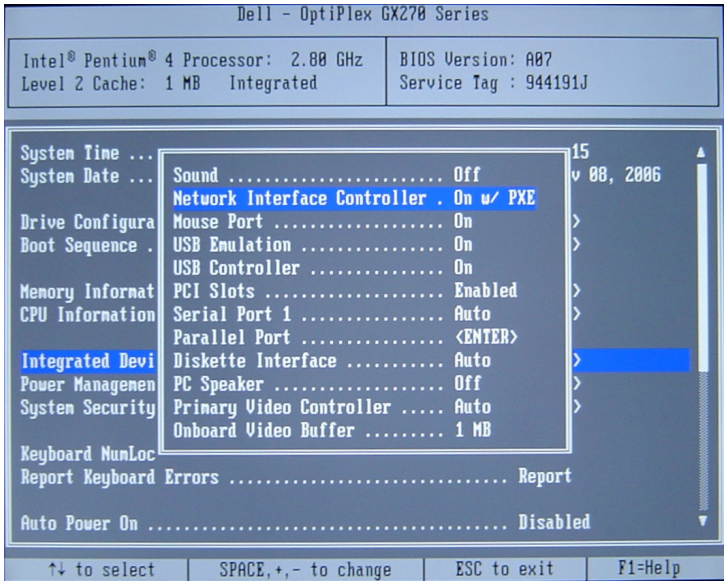
A.3 Configurer un poste afin de *booter* en mode *PXE*

Au laboratoire, le poste utilisé afin de *booter* en mode *PXE* est un Dell Optiplex GX260.

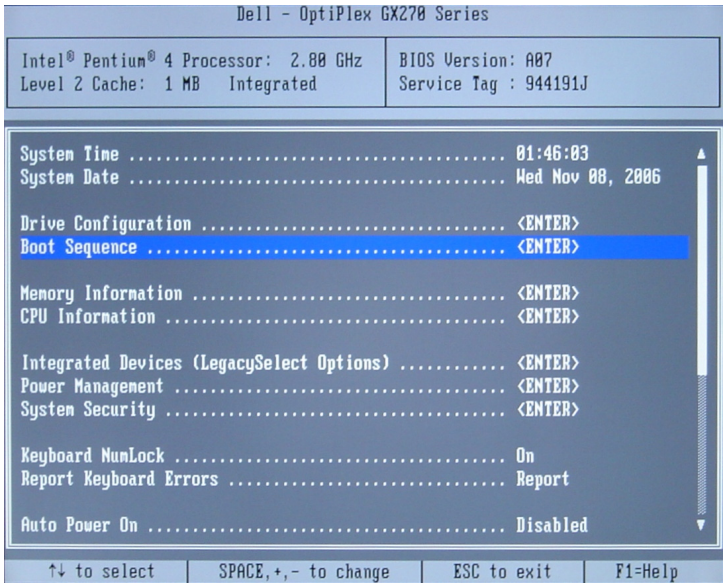
Pour le configurer voici les diverses manipulations à effectuer :

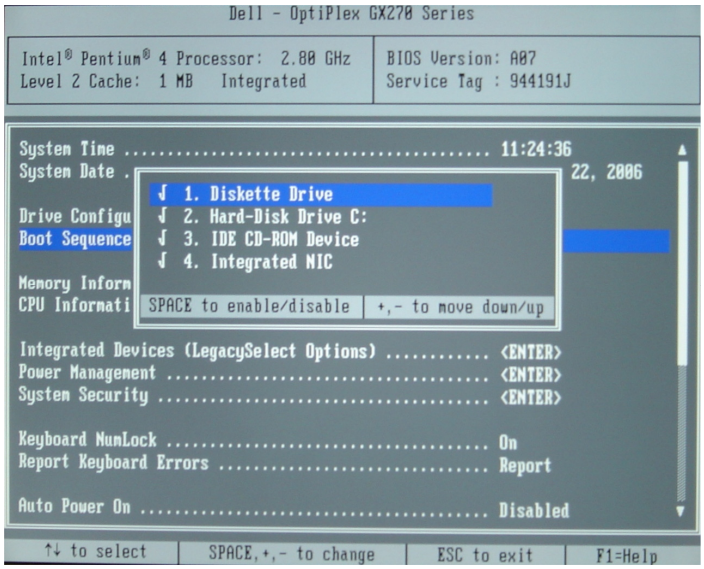
Dans le BIOS, activer le mode *PXE* :





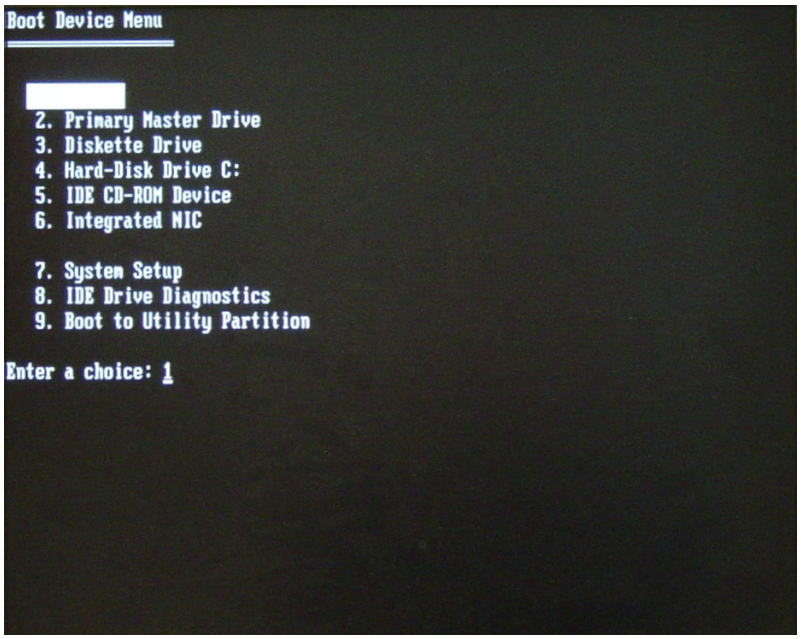
Puis activer le mode *PXE* en tant que mode de *boot* disponible :





Le poste est maintenant correctement configuré afin de *booter* en mode *PXE*.

Lors de son démarrage, il faut appuyer sur F12 pour voir le menu de *boot*, puis sélectionner ***Integrated NIC*** (*Network Interface Card*) afin de *booter* sur la carte réseau :



Le mode de *boot PXE* sera alors lancé.

A.4 Résumé de la présentation *Technet* à Beaulieu

Je suis allé à une présentation *Technet* sur *Windows Vista*, à Beaulieu (Lausanne).

Voici un bref résumé de ce qui a été dit lors de cette conférence :

Lors de cet événement il y a eu 3 présentations, une introduction sur *Vista*, une présentation sur la **migration** et une autre sur le **déploiement**.

Résumé de la partie d'introduction :

- *Ready Boot* => possibilité d'utiliser une clef USB comme cache disque. Cela augmenterait les performances de l'ordinateur.
  Les clefs USB sont performantes lors d'accès aléatoires pour de petits paquets, les disques durs sont performants pour de gros fichiers contigus.

- Protection des données : *Volume Shadow Copy* => nous permet de retrouver une version précédente d'un fichier Word par exemple, en faisant un clic droit sur le fichier.

Résumé de la partie **Migration** :

- *Windows Vista Upgrade Advisor RC* => *software* qui nous permet de voir si un système *Xp* peut être migré vers *Vista*.

  Il nous permet de choisir qu'elle version de *Vista* est la plus appropriée pour migrer notre système (quelle version est la plus compatible)

  Vérifie que notre *hardware* soit compatible Vista.

  Vérifie que nos *softwares* soient compatibles Vista.

  Crée un rapport décrivant tous les tests effectués.

  Lien pour télécharger *Windows Vista Upgrade Advisor RC* :
  www.microsoft.com/windowsvista/getready/upgradeadvisor

- *Application Compatibility Toolkit 5.0 (ACT)* => *software* qui parcourt tout un réseau et collecte les différents *softwares* installés dans les divers postes du réseau.

  Ce programme envoie/reçoit des rapports contenant la liste de tous les programmes installés. Il sera ensuite possible de vérifier si tous ces programmes sont compatibles *Vista*.

  Les utilisateurs de ce programme peuvent mettre des commentaires en ligne sur certaines applications, afin d'informer les autres utilisateurs sur le fonctionnement de ces applications, écrire des solutions pour aider les autres utilisateurs.

Résumé sur la partie **Déploiement** :

- 3 sortes de déploiement possible :

  a) Une toute nouvelle installation

  b) Récupérer les informations utilisateur du pc_1 puis les copier vers pc_2, installer Vista sur pc_1 puis récupérer les données utilisateur

  c) *Side by Side*

- Les versions 32 bits et 64 bits de *Windows Vista* sont sur la même image WIM

- Pour certaines entreprises il peut être utile de sauvegarder l'ancien OS avant d'installer *Windows Vista*, ceci dans le cas ou *Vista* ne fonctionnerait pas sur le pc à installer.

- *SMS (System Management Server)*, *software* payant, facilite le déploiement.

  Ce programme utilise les divers *tools*, tells qu'imagex pour le déploiement.

  Utilise divers scripts pour automatiser l'installation.

  BDD Workbench => possible de choisir de faire un déploiement pour SMS