ICACLS

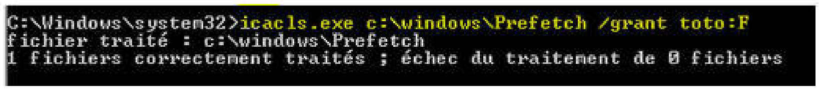
After* cacls, xcacls.vbs, *now we have* icacls *to set file and folder permissions.

Exemple1

Pour ajouter l'utilisateur standard « toto » aux utilisateurs qui ont le droit de modifier le dossier:

```
icacls dossier /grant toto:F
```

Si tout se passe bien voici l'écran que vous devriez obtenir



```
C:\Windows\system32>icacls.exe c:\windows\Prefetch /grant toto:F
fichier traité : c:\windows\Prefetch
1 fichiers correctement traités ; échec du traitement de 0 fichiers
```

On peut à présent tester en double cliquant sur le dossier: »*c:\windows\Prefetch*« .Et la plus aucun message bloquant nous avertissant que l'on est pas habilité à le parcourir.

J'ai volontairement donné un exemple simple mais sachez qu'il est possible de faire beaucoup de paramétrages fins avec **icacls.exe**
Pour plus d'infos sur les possibilités que nous laisse faire « icacls », taper:

```
icacls.exe /?
```

Ça ne vous dit pas quelque chose ?

Exemple2

```
icacls "%folder%" /grant:r "%username%:(OI)(CI)F"

echo "%folder%" est avec un control total !
```

exemple3

```
icacls "%folder%" /deny "%username%:(OI)(CI)F"

echo "%folder%" est avec un acces interdit !
```

Exemples :

Interdit à l'utilisateur "toto" l'accès au fichier "C:\titi.txt" (il n'y avait pas d'utilisateur "toto" dans les droits de ce dossier) :

```
CACLS C:\titi.txt /D toto
```

Interdit à l'utilisateur "toto" l'accès au fichier "C:\titi.txt" (il y avait déjà l'utilisateur "toto" dans les droits de ce dossier) :

```
CACLS C:\titi.txt /E /D toto
```

Interdit à l'utilisateur "toto" l'accès au dossier "C:\titi" et à tous les objets enfants qui s'y trouvent :

```
CACLS C:\titi /T /D toto
```

Autorise "toto" à accéder en lecture au dossier "C:\titi" et à tous les objets enfants qui s'y trouvent :

```
CACLS C:\titi /T /G toto:R
```

Autorise "toto" et "tata" à accéder en écriture au dossier "C:\titi" et à tous les objets enfants qui s'y trouvent :

```
CACLS C:\titi /T /G toto:C tata:C
```

Compléments

Now let's script the ntfs permissions for the apps share:
- "(OI)(CI):F" means Full Control "This Folder, Subfolders and files"
- "(OI)(CI):M" means Modify "This Folder, Subfolders and files"
- "/inheritance:r" means remove all inherited ACL's from parent

(OI) This folder and files

(CI) This folder and subfolders.

(OI)(CI) This folder, subfolders, and files.

(OI)(CI)(IO) Subfolders and files only.

(CI)(IO) Subfolders only.

(OI)(IO) Files only.

and the permission possibilities

perm is a permission mask and can be specified in one of two forms:

a sequence of simple rights:

N - no access

F - full access

M - modify access

RX - read and execute access

R - read-only access

W - write-only access

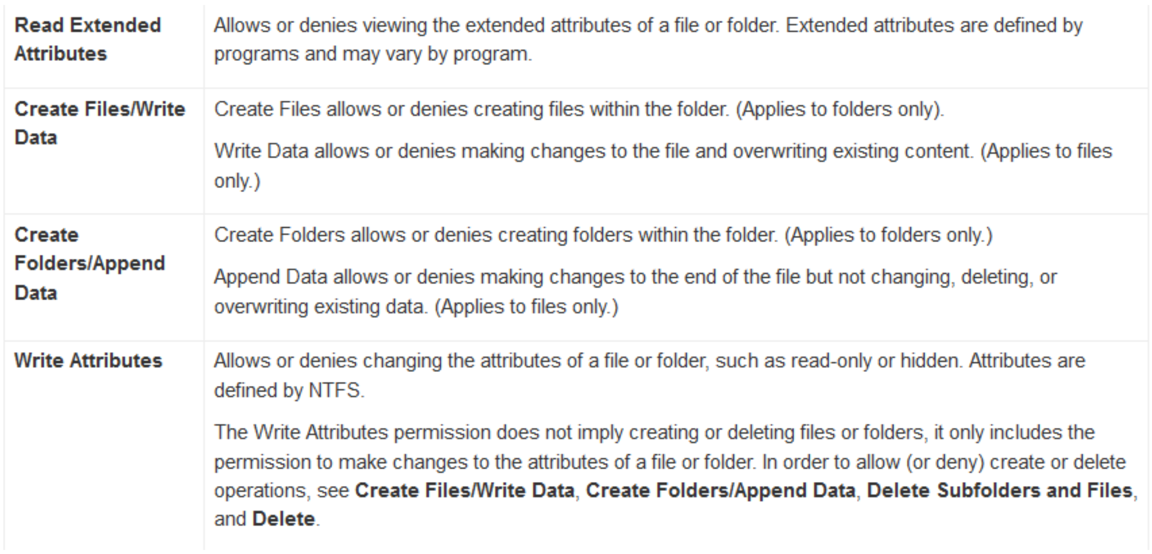
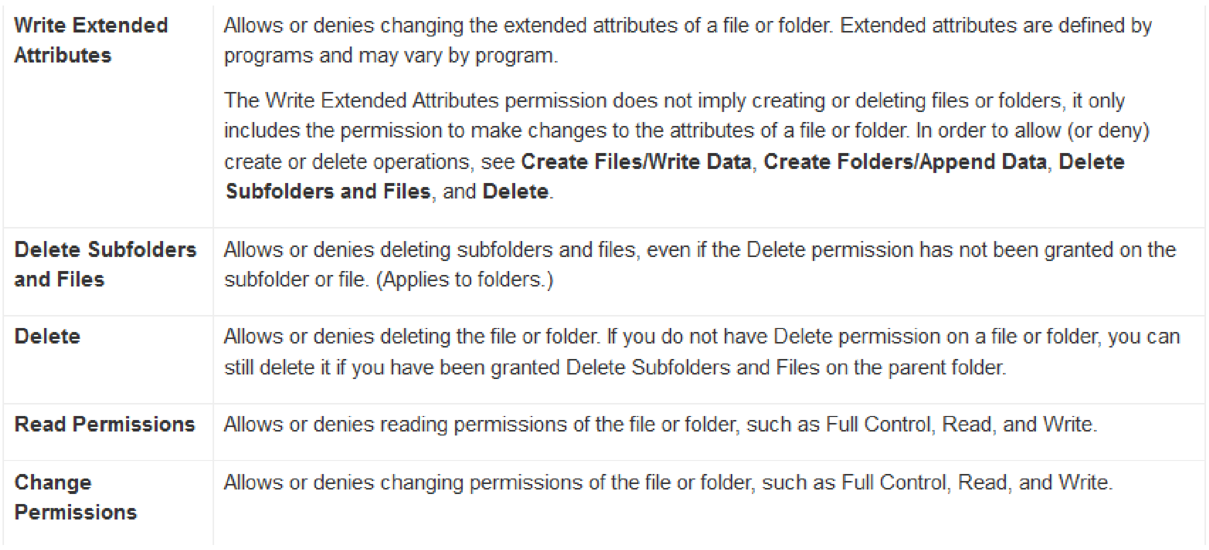
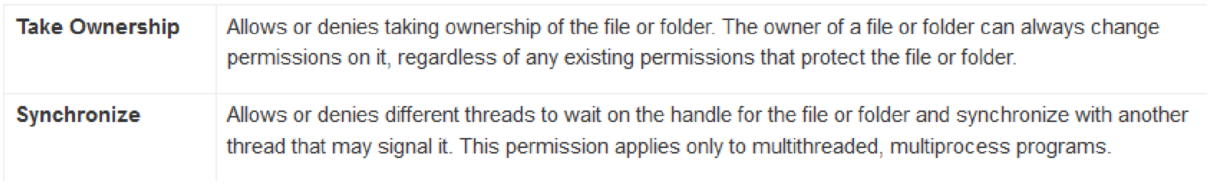
D - delete access

a comma-separated list in parentheses of specific rights:

- DE - delete
- RC - read control
- WDAC - write DAC
- WO - write owner
- S - synchronize
- AS - access system security
- MA - maximum allowed
- GR - generic read
- GW - generic write
- GE - generic execute
- GA - generic all
- RD - read data/list directory
- WD - write data/add file
- AD - append data/add subdirectory
- REA - read extended attributes
- WEA - write extended attributes
- X - execute/traverse
- DC - delete child
- RA - read attributes
- WA - write attributes

Here the discription of all the possible NTFS permissions

Permission	Description
Traverse Folder/Execute File	**For folders**: Traverse Folder allows or denies moving through folders to reach other files or folders, even if the user has no permissions for the traversed folders. (Applies to folders only.) Traverse folder takes effect only when the group or user is not granted the **Bypass traverse checking** user right in the Group Policy snap-in. (By default, the Everyone group is given the **Bypass traverse checking** user right.) **For files:** Execute File allows or denies running program files. (Applies to files only). Setting the Traverse Folder permission on a folder does not automatically set the Execute File permission on all files within that folder.
List Folder/Read Data	List Folder allows or denies viewing file names and subfolder names within the folder. List Folder only affects the contents of that folder and does not affect whether the folder you are setting the permission on will be listed. (Applies to folders only.) Read Data allows or denies viewing data in files. (Applies to files only.)
Read Attributes	Allows or denies viewing the attributes of a file or folder, such as read-only and hidden. Attributes are defined by NTFS.

Read Extended Attributes	Allows or denies viewing the extended attributes of a file or folder. Extended attributes are defined by programs and may vary by program.
Create Files/Write Data	Create Files allows or denies creating files within the folder. (Applies to folders only). Write Data allows or denies making changes to the file and overwriting existing content. (Applies to files only.)
Create Folders/Append Data	Create Folders allows or denies creating folders within the folder. (Applies to folders only.) Append Data allows or denies making changes to the end of the file but not changing, deleting, or overwriting existing data. (Applies to files only.)
Write Attributes	Allows or denies changing the attributes of a file or folder, such as read-only or hidden. Attributes are defined by NTFS. The Write Attributes permission does not imply creating or deleting files or folders, it only includes the permission to make changes to the attributes of a file or folder. In order to allow (or deny) create or delete operations, see **Create Files/Write Data**, **Create Folders/Append Data**, **Delete Subfolders and Files**, and **Delete**.
Write Extended Attributes	Allows or denies changing the extended attributes of a file or folder. Extended attributes are defined by programs and may vary by program. The Write Extended Attributes permission does not imply creating or deleting files or folders, it only includes the permission to make changes to the attributes of a file or folder. In order to allow (or deny) create or delete operations, see **Create Files/Write Data**, **Create Folders/Append Data**, **Delete Subfolders and Files**, and **Delete**.
Delete Subfolders and Files	Allows or denies deleting subfolders and files, even if the Delete permission has not been granted on the subfolder or file. (Applies to folders.)
Delete	Allows or denies deleting the file or folder. If you do not have Delete permission on a file or folder, you can still delete it if you have been granted Delete Subfolders and Files on the parent folder.
Read Permissions	Allows or denies reading permissions of the file or folder, such as Full Control, Read, and Write.
Change Permissions	Allows or denies changing permissions of the file or folder, such as Full Control, Read, and Write.
Take Ownership	Allows or denies taking ownership of the file or folder. The owner of a file or folder can always change permissions on it, regardless of any existing permissions that protect the file or folder.
Synchronize	Allows or denies different threads to wait on the handle for the file or folder and synchronize with another thread that may signal it. This permission applies only to multithreaded, multiprocess programs.

On the profiles share, only the “domain admins” should be allowed to enter all “Folders, Subfolders and files” (hence the (OI)(CI):F) , everyone else should be able to to ready “this folder only”.
So without an combination of (CI) and/or (OI) it means “this folder only”

```
icacls "d:\profiles" /grant "domain admins":(OI)(CI)F /inheritance:r
icacls "d:\profiles" /grant "everyone":R /inheritance:r
```

Upon creating a new user, the Domain Admin should manually create a profile folder for the user and add the user with appropriate rights.

The same goes for the users share containing the homedirectories of all users

```
icacls "d:\users" /grant "domain admins":(OI)(CI)F /inheritance:r
icacls "d:\users" /grant "everyone":R /inheritance:r
```