# Command Line-Version (SetACL.exe) – Syntax and Description

# Syntax

```
SetACL -on ObjectName -ot ObjectType
-actn Action1 ParametersForAction1
[-actn Action2 ParametersForAction2]
[Options]
```

It basically works like this: you specify one or more actions. Although actions have default parameters where that makes sense, in many cases you need to give SetACL some data to work with. That happens via additional parameters. Example: if the action is "set permissions", then you need one additional parameter for each permission (ACE) to set, like "full control for everybody".

# Return codes

SetACL returns 0 upon successful execution and a higher number if errors occurred. A complete list of all return codes can be found <u>here</u>.

# Parameter List

Lists all command-line parameters SetACL can digest. Some are mandatory, others are required if a certain other parameter has been specified, some are optional.

Click on the links to jump to sections with more detailed explanations.

### **Required parameters**

### **Object Name**

-on <u>ObjectName</u> Sets path and/or name of the object to process. Required

### **Object Type**

-ot <u>ObjectType</u> Sets the type of object. Required

### Action parameters

### ACE

-actn ace -ace "n:<u>Trustee;</u>p:<u>Permission[;i:Inheritance;</u>m:<u>Mode;</u>w:<u>Where</u>]"

Adds or modifies access control entries (ACEs). Multiple ACEs may be specified like this: -actn ace -ace "n:UserX;p:full" -ace "n:GroupY;p:read"

### Trustee

-actn trustee -trst "n1:<u>Trustee1[;n2:Trustee2];ta:TrusteeAction[;</u>w:<u>What]</u>" Removes Trustee1's ACEs, or replaces Trustee1 within ACEs with Trustee2, or copies ACEs from Trustee1 to Trustee2. Multiple trustees may be specified like this: -actn trustee -trst "n1:UserA;n2:UserB;ta:cpytrst" -trst "n1:UserX;n2:UserY;ta:cpytrst"

or (new in SetACL 3.0)

-actn trustee -trst "csv:<u>TrusteeInputFile</u>;ta:<u>TrusteeAction[;w:What]</u>" Removes ACEs with specified trustees, replaces trustees within ACEs or copies ACEs to different trustees. Input is read from *TrusteeInputFile*.

### Domain

-actn domain -dom "n1:Domain1[;n2:Domain2];da:DomainAction[;w:What]"

Removes all ACEs with trustees from *Domain1*, or replaces *Domain1* trustees in ACEs with corresponding *Domain2* trustees, or copies *Domain1* trustee ACEs to corresponding *Domain2* trustee ACEs. Multiple domains may be specified like this: -actn domain -dom "n1:DomainA;n2:DomainB;da:cpydom" -dom "n1:DomainX;n2:DomainY;da:cpydom"

### Set Owner

-actn setowner -ownr "n:<u>Trustee</u>" Sets the owner.

### Set Primary Group

-actn setgroup -grp "n:<u>Trustee</u>" Sets the primary group.

### Set Protection Flags

-actn setprot -op "dacl: Protection; sacl: Protection"

Configures protection from inheritance (whether to 'allow inheritable permissions from the parent object to propagate to this object').

### **Reset Children**

#### -actn rstchldrn -rst Where

Resets permissions on all sub-objects and enables propagation of inherited permissions.

### Clear

-actn clear -clr Where

Clears the ACL of any non-inherited ACEs. You can configure whether to remove non-inherited ACEs from the DACL, the SACL or both.

### List

-actn list [-lst "f:<u>Format</u>;w:<u>What</u>;i:<u>ListInherited</u>;s:<u>DisplaySID</u>;oo:<u>OrphanedOnly</u>"] [-bckp <u>Filename</u>] Lists permissions. If <u>-lst</u> is omitted, a listing of the non-inherited permissions is created in tabular format. The result can optionally be written to a backup file.

### Restore

#### -actn restore -bckp Filename

Restores entire or parts of security descriptors from a file created with the list function. Please note that the listing needs to be in SDDL format.

### **Delete Orphaned SIDs**

#### -actn delorphanedsids [-os Where]

Deletes ACEs with orphaned SIDs (SIDs from deleted users/groups). You can configure whether to do this for the DACL, the SACL or both (default: DACL).

### **Multiple Actions**

If multiple actions are specified, they are processed in this order:

- 1. restore
- 2. clear
- 3. trustee
- 4. domain
- 5. ace, setowner, setgroup, setprot
- 6. rstchldrn
- 7. list

### **Optional parameters**

### Recursion

-rec <u>Recursion</u> Configures recursion.

### Log File

*-log <u>Filename</u>* Sets the name of a log file.

### Filter Keyword

#### -fltr Keyword

Adds a filter keyword. May be specified more than once.

### Silent

-silent

Enables silent mode (no output written to screen).

### Ignore Errors

-ignoreerr

Enables the error ignore mode, where SetACL does not stop upon an error.

### Raw Mode

-raw

In version 2.1.1 filtering of pseudo-inherited ACEs was introduced. This changes pseudo-inherited ACEs into truly inherited ACEs right before they are set. This behavior may not be desired, so it can be switched off with the command-line parameter "-raw" beginning with version 2.2.2.

# **Parameter Description**

### ObjectName

Name of the object to process. The format depends on the object type, but you should be able to specify names in all common variations. If you want to access a remote machine, prepend "\\Servername\" to the name of the object.

**File system paths** can be relative, but should be absolute; only then can SetACL convert them to the format required by the kernel for very long paths – in other words, if you want SetACL to work with paths longer than MAX\_PATH (260 characters), use absolute paths. You can use drive letters for local and mapped network drives and UNC paths for remote computers. You can also use volume GUIDs, typically to specify a mounted volume. Here are some examples of valid file system paths:

C:\Data

Directory "Data" on local drive C:

M:\Data

Directory "Data" on mapped network drive M:

\\Server\Share\Data

Directory "Data" on file server "Server" below share "Share"

\\Server.domain.com\Share\Data

Same as before, but the server name specified as DNS name instead of NetBIOS name \\?\Volume{8a78ee92-4b22-11df-89ee-bb2cb99ebac7}\

Root directory of a volume identified by its GUID

\\?\Volume{8a78ee92-4b22-11df-89ee-bb2cb99ebac7}

(Note the missing backslash at the end) Volume identify by its GUID. Tip: typically you do not want to set permissions on a volume, but on it's root directory.

**Registry paths** can be specified flexibly, using the full hive names (e.g. HKEY\_LOCAL\_MACHINE) or the well-known short forms (e.g. HKLM).

Note: On remote computers only the hives HKU and HKLM can be accessed!

Examples:

HKLM\Software The registry key "Software" below hive HKEY\_LOCAL\_MACHINE. HKEY\_LOCAL\_MACHINE\Software The registry key "Software" below hive HKEY\_LOCAL\_MACHINE. \\Computer\HKEY\_LOCAL\_MACHINE\Software

The registry key "Software" below hive HKEY\_LOCAL\_MACHINE on the remote computer "Computer".

Paths to the **other supported object types** are specified in a way similar to file system paths. Here are some examples:

LanmanWorkstation The workstation service on the local computer. \\Server\LanmanWorkstation The workstation service on the remote computer "Server". "HP LaserJet 4200 PCL 6" The HP 4200 printer on the local computer. "\\Server\HP LaserJet 4200 PCL 6" The HP 4200 printer on the remote computer "Server". Data\$ The share Data\$ on the local computer. \\Server\Data\$ The share Data\$ on the remote computer "Server".

root

The root of the WMI namespace.

If the object name **ends with a backslash** and you enclose it in quotes, make sure to escape the last backslash with another backslash. For example, use  $C: \setminus \setminus$  instead of  $C: \setminus$ .

### ObjectType

Type of object:

file Directory/file reg Registry key srv Service prn Printer shr Network share wmi WMI object

### TrusteeAction

Action to perform on the trustee specified:

```
remtrst
```

Remove all ACEs belonging to trustee specified.

```
repltrst
Replace trustee 'n1' by 'n2' in all ACEs.
```

cpytrst

```
Copy the permissions for trustee 'n1' to 'n2'.
```

### TrusteeInputFile

Name (and optionally path) of a file that contains trustees for the specified trustee action. The format depends on the trustee action. For "remtrst" it is one trustee per line, for "repltrst" and "cpytrst" it is:

SourceTrustee,TargetTrustee

Trustees can be specified via their names or SIDs. Details can be found here.

### DomainAction

Action to perform on the domain specified:

remdom

Remove all ACEs belonging to trustees of the domain specified.

repldom

Replace trustees from domain 'n1' by trustees with the same name from domain 'n2' in all ACEs. *cpydom* 

Copy permissions from trustees from domain 'n1' to trustees with the same name from domain 'n2' in all ACEs.

#### Explanation:

For every SID in the ACEs of the ACL(s), the name of the domain and user/group of the corresponding account is looked up. If the domain name is equal to the domain name 'n1' specified, the ACE is deleted in the case of 'remdom'. In the case of 'repldom' or 'cpydom' a user/group of the same name is searched in the domain 'n2' specified. If such a user/group is found, either a new ACE with the same permissions and flags is created ('cpydom'), or the SID in the ACE is replaced with the SID of the user/group in the domain 'n2' specified ('repldom').

### Trustee

Name or SID of a trustee (a user or group). Supported formats:

- 1. [{computer | domain}\]name
- 2. SID string

Where:

- computer: DNS or NetBIOS name of a computer -> 'name' must be a local account on that computer.
- domain: DNS or NetBIOS name of a domain -> 'name' must be a domain user or group.
- name: user or group name.
- SID string: String representation of a SID, e.g. S-1-5-32-544 for the group Administrators. <u>Here</u> is a list of well-known SIDs that are the same across Windows versions and languages.

If no computer or domain name is given, SetACL tries to find a SID for 'name' in the following order:

- 1. built-in accounts and well-known SIDs
- 2. local accounts
- 3. primary domain
- 4. trusted domains

Specifying trustees as SIDs instead of using their names can be very useful in multi-language environments, because SIDs are language-independent, whereas predefined names are not. An example: the group 'administrators' is called 'administratoren' in German Windows versions. If you want your SetACL script to run on servers installed in either language you can use the well-known SID of the group 'administrators'. Well-known SIDs are identical on every system. A list can be found <u>here.</u>

### Domain

Name of a domain (NetBIOS or DNS name).

# Permission

Permission(s) to set. Comma-separated list. Here is a list of all permissions that can be set.

Example: 'read,write\_ea,write\_dacl'

# DisplaySID

Display trustee names as SIDs?

y Yes n No b Both (names and SIDs)

# Inheritance

Inheritance flags for the ACE. This may be a comma-separated list containing the following:

```
so
sub-objects
sc
sub-containers
np
no propagation
io
inherit only
```

Example: 'io,so'

# Mode

Access mode of this ACE.

The following access modes are valid with DACLs:

set

Replace all permissions for given trustee by those specified.

grant

Add permissions specified to existing permissions for given trustee.

deny

Deny permissions specified.

revoke

Remove permissions specified from existing permissions for given trustee.

These access modes are valid with SACLs:

```
aud_succ
```

Add an audit success ACE.

aud\_fail Add an audit failure ACE.

#### revoke

Remove permissions specified from existing permissions for given trustee.

### Where

Apply settings to DACL, SACL, or both (comma-separated list):

dacl

```
Process the DACL (permissions list). sacl
```

Process the SACL (auditing list). *dacl,sacl* 

Process DACL and SACL.

## Recursion

These recursion settings are valid for file system objects:

```
no
No recursion.
cont
Recurse, and process directories only.
obj
Recurse, and process files only.
cont_obj
Recurse, and process directories and files.
```

These recursion settings are valid for registry objects:

no Do not recurse. yes Do Recurse.

Recursion is not supported for other object types.

# Protection

Controls the flag 'allow inheritable permissions from the parent object to propagate to this object':

nc

Do not change the current setting.

np

Object is not protected, i.e. inherits from parent.

*p\_c* Object is protected, ACEs from parent are copied.

p\_nc

Object is protected, ACEs from parent are not copied.

### Format

Which list format to use:

sddl

Standardized SDDL format. Only listings in this format can be restored.

csv/own

SetACL's own format. Easier to read than SDDL.

tab

SetACL's tabular format. Humans prefers this over CSV and SDDL (especially over SDDL).

### What

Which components of security descriptors to include in the listing (comma-separated list):

d DACL s SACL o Owner g Primary group

Example: 'd,s'

# ListInherited

List inherited permissions? If no, only permissions set directly on an object are listed (default).

```
y
Yes
n
No
```

### Filename

Name of a file used for list/backup/restore operations or logging. The file is written in UNICODE.

# Keyword

Keyword to filter object names by. Names and paths containing this keyword are not processed.

# OrphanedOnly

Requires at least SetACL 3.0.

If enabled, only objects with orphaned SIDs are listed. Orphaned SIDs are SIDs that cannot be resolved to a name.

```
y
Yes
n
No
```

# Valid Standard Permissions

Standard permissions sets provide easy access to the most commonly used combinations of specific permissions (for a list of the latter, see below).

### Files / Directories

read

Read (FILE\_LIST\_DIRECTORY + FILE\_READ\_EA + FILE\_READ\_ATTRIBUTES + READ\_CONTROL) write

Write (FILE\_ADD\_FILE + FILE\_ADD\_SUBDIRECTORY + FILE\_WRITE\_EA + FILE\_WRITE\_ATTRIBUTES)

list\_folder

List folder (FILE\_LIST\_DIRECTORY + FILE\_READ\_EA + FILE\_TRAVERSE + FILE\_READ\_ATTRIBUTES + READ\_CONTROL)

read\_ex

Read, execute (FILE\_LIST\_DIRECTORY + FILE\_READ\_EA + FILE\_TRAVERSE + FILE\_READ\_ATTRIBUTES + READ\_CONTROL)

#### change

Change (FILE\_LIST\_DIRECTORY + FILE\_ADD\_FILE + FILE\_ADD\_SUBDIRECTORY + FILE\_READ\_EA + FILE\_WRITE\_EA + FILE\_TRAVERSE + FILE\_READ\_ATTRIBUTES + FILE\_WRITE\_ATTRIBUTES + READ\_CONTROL + DELETE)

full

Full access (FILE\_LIST\_DIRECTORY + FILE\_ADD\_FILE + FILE\_ADD\_SUBDIRECTORY + FILE\_READ\_EA + FILE\_WRITE\_EA + FILE\_TRAVERSE + FILE\_DELETE\_CHILD + FILE\_READ\_ATTRIBUTES + FILE\_WRITE\_ATTRIBUTES + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE)

### Registry

read

Read (KEY\_ENUMERATE\_SUB\_KEYS + KEY\_EXECUTE + KEY\_NOTIFY + KEY\_QUERY\_VALUE + KEY\_READ + READ\_CONTROL)

full

Full access (KEY\_CREATE\_LINK + KEY\_CREATE\_SUB\_KEY + KEY\_ENUMERATE\_SUB\_KEYS + KEY\_EXECUTE + KEY\_NOTIFY + KEY\_QUERY\_VALUE + KEY\_READ + KEY\_SET\_VALUE + KEY\_WRITE + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE)

### Printers

print

Print (PRINTER\_ACCESS\_USE + READ\_CONTROL)

#### man\_printer

Manage printer (PRINTER\_ACCESS\_ADMINISTER + PRINTER\_ACCESS\_USE + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE)

man\_docs

Manage documents (JOB\_ACCESS\_ADMINISTER + JOB\_ACCESS\_READ + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE)

full

Full access (manage printer + manage documents)

### Services

#### read

Read (SERVICE\_ENUMERATE\_DEPENDENTS + SERVICE\_INTERROGATE + SERVICE\_QUERY\_CONFIG + SERVICE\_QUERY\_STATUS + SERVICE\_USER\_DEFINED\_CONTROL + READ\_CONTROL)

#### start\_stop

Start / Stop (SERVICE\_ENUMERATE\_DEPENDENTS + SERVICE\_INTERROGATE + SERVICE\_PAUSE\_CONTINUE + SERVICE\_QUERY\_CONFIG + SERVICE\_QUERY\_STATUS + SERVICE\_START + SERVICE\_STOP + SERVICE\_USER\_DEFINED\_CONTROL + READ\_CONTROL)

full

Full access (SERVICE\_CHANGE\_CONFIG + SERVICE\_ENUMERATE\_DEPENDENTS + SERVICE\_INTERROGATE + SERVICE\_PAUSE\_CONTINUE + SERVICE\_QUERY\_CONFIG + SERVICE\_QUERY\_STATUS + SERVICE\_START + SERVICE\_STOP + SERVICE\_USER\_DEFINED\_CONTROL + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE)

### Shares

read

Read (SHARE\_CHANGE + READ\_CONTROL + SYNCHRONIZE)

change

Change (SHARE\_READ + DELETE + READ\_CONTROL + SYNCHRONIZE)

full

Full access (SHARE\_READ + SHARE\_CHANGE + SHARE\_WRITE + READ\_CONTROL + WRITE\_OWNER + WRITE\_DAC + DELETE + SYNCHRONIZE)

### WMI

full

Full access (WBEM\_ENABLE + WBEM\_METHOD\_EXECUTE + WBEM\_FULL\_WRITE\_REP + WBEM\_PARTIAL\_WRITE\_REP + WBEM\_WRITE\_PROVIDER + WBEM\_REMOTE\_ACCESS + READ\_CONTROL + WRITE\_DAC)

execute

Execute access (WBEM\_ENABLE + WBEM\_METHOD\_EXECUTE + WBEM\_WRITE\_PROVIDER) remote\_access

Remote access (WBEM\_ENABLE + WBEM\_REMOTE\_ACCESS + WBEM\_WRITE\_PROVIDER) enable\_account

Enable account (WBEM\_ENABLE)

# Valid Specific Permissions

### Files / Directories

#### traverse

**Traverse Folder/Execute File 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.) **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\_dir

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\_attr

**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\_ea

**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.

#### add\_file

**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).

#### add\_subdir

**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\_attr

**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\_ea

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".

#### del\_child

**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

**Delete** Allows or denies deleting the file or folder. If you don't 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 dacl* 

**Read Permissions** Allows or denies reading permissions of the file or folder, such as Full Control, Read, and Write.

#### write\_dacl

**Write Permissions** Allows or denies changing permissions of the file or folder, such as Full Control, Read, and Write.

#### write\_owner

**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.

### Registry

query val Query value set\_val Set value create subkey Create subkeys enum subkeys Enumerate subkeys notify Notify create link Create link delete Delete write dacl Write permissions write\_owner Take ownership read\_access Read control

### Services

SERVICE\_CHANGE\_CONFIG

Required to call the ChangeServiceConfig or ChangeServiceConfig2 function to change the service configuration.

SERVICE\_ENUMERATE\_DEPENDENTS

Required to call the EnumDependentServices function to enumerate all the services dependent on the service.

SERVICE\_INTERROGATE

Required to call the ControlService function to ask the service to report its status immediately. SERVICE\_PAUSE\_CONTINUE

Required to call the ControlService function to pause or continue the service.

#### SERVICE\_QUERY\_CONFIG

Required to call the QueryServiceConfig and QueryServiceConfig2 functions to query the service configuration.

#### SERVICE\_QUERY\_STATUS

Required to call the QueryServiceStatus or QueryServiceStatusEx function to ask the service control manager about the status of the service. Required to call the NotifyServiceStatusChange function to receive notification when a service changes status.

SERVICE\_START

Required to call the StartService function to start the service.

#### SERVICE\_STOP

Required to call the ControlService function to stop the service.

SERVICE\_USER\_DEFINED\_CONTROL

Required to call the ControlService function to specify a user-defined control code.

# Managing File System Permissions with SetACL.exe

### Contents

- Example 1.1 Setting Permissions
- Example 1.2 Setting Multiple Permissions
- Example 1.3 Adding Multiple ACEs
- <u>Example 1.4 Specifying SIDs</u>
- Example 1.5 Remote Systems
- Example 1.6 Auditing Entries (SACL)
- Example 1.7 Cleaning up ACLs
- Example 1.8 Resetting Child Objects
- Example 1.9 Using the Log File
- Example 1.10 Silent Mode
- Example 1.11 Filtering and Excluding Objects
- Example 1.12 Setting the Owner
- Example 2 Listing and Backup
- Example 3 Restore
- <u>Example 4 Copying Permissions Between Users</u>
- Example 5 Migrating Permissions Between Domains
- Example 6 Removing all Permissions of a User

# Example 1.1 – Setting Permissions

SetACL.exe -on "C:\my dir" -ot file -actn ace -ace "n:domainl\userl;p:change"

Sets 'change' permissions on the directory 'c:\my dir' for user 'user1' in domain 'domain1'.

# Example 1.2 – Setting Multiple Permissions

SetACL.exe -on "C:\my dir" -ot file -actn ace -ace "n:domainl\user1;p:read,write\_dacl"

Same as the previous example, but sets the following two permissions:

- read (standard permission set)
- write\_dacl (specific permission)

# Example 1.3 – Adding Multiple ACEs

```
SetACL.exe -on "C:\my dir" -ot file -actn ace
    -ace "n:domainl\user1;p:change"
    -ace "n:administrators;p:full"
```

Same as the first example, but additionally sets 'full' permissions for the group 'administrators'.

# Example 1.4 – Specifying SIDs

```
SetACL.exe -on "C:\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
```

An improved version of the previous example: 'administrators' is a built-in group, whose name is dependent on the language of the operating system. Therefore it is better to use its **well-known SID** which never changes.

### Example 1.5 – Remote Systems

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
```

Same as the previous example, but using a **UNC name** to access the server 'server1' remotely via the network share 'share1'.

# Example 1.6 – Auditing Entries (SACL)

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
    -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
```

Same as the previous example, but additionally setting an **auditing entry** for all ('full') failed attempts of 'user2' from domain 'domain2'.

### Example 1.7 – Cleaning up ACLs

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
    -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
    -actn clear -clr "dac1,sacl"
```

Same as the previous example, but first (see ordering of actions in the documentation) the DACL and SACL are **cleared of any non-inherited entries**, and then the specified ACEs are set. This effectively 'cleans up' messed-up ACLs.

# Example 1.8 – Resetting Child Objects

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
    -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
    -actn clear -clr "dac1,sacl"
    -actn rstchldrn -rst "dac1,sacl"
```

Same as the previous example, but even more housekeeping is done. **Propagation of inherited permissions** is enabled for all sub-objects whose permissions are also reset, resulting in only the specified permissions being active for a whole directory tree.

# Example 1.9 – Using the Log File

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
    -ace "n:domain1\user1;p:change"
    -ace "n:S-1-5-32-544;p:full"
    -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
    -actn clear -clr "dacl,sacl"
    -actn rstchldrn -rst "dacl,sacl"
    -log "c:\my files\setacl log.txt"
```

Same as the previous example, but all output is written both to the **screen and to the log file** 'c:\my files\setacl log.txt'.

# Example 1.10 – Silent Mode

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
  -ace "n:domain1\user1;p:change"
  -ace "n:S-1-5-32-544;p:full"
  -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
  -actn clear -clr "dac1,sacl"
  -actn rstchldrn -rst "dac1,sacl"
  -log "c:\my files\setacl log.txt"
  -silent
```

Same as the previous example, but **no output** is written to the screen, only to the log file 'c:\my files\setacl log.txt'.

# Example 1.11 – Filtering and Excluding Objects

```
SetACL.exe -on "\\server1\share1\my dir" -ot file -actn ace
  -ace "n:domain1\user1;p:change"
  -ace "n:S-1-5-32-544;p:full"
  -ace "n:domain2\user2;p:full;m:aud_fail;w:sacl"
  -actn clear -clr "dac1,sacl"
  -actn rstchldrn -rst "dac1,sacl"
  -log "c:\my files\setacl log.txt"
  -silent
  -fltr "secrets" -fltr "top secret"
```

Same as the previous example, but files/directories containing the strings 'secrets' or 'top secret' are **excluded**.

# Example 1.12 – Setting the Owner

```
SetACL.exe -on "\\server1\share1\users" -ot file -actn setprot
    -op "dacl:np;sacl:nc"
    -rec cont_obj
    -actn setowner -ownr "n:S-1-5-32-544"
```

**Resets a whole directory tree** to what most administrators dream of: the owner of all files and directories is set to the group 'administrators' and the flag 'allow inheritable permissions from the parent object to propagate to this object' is enabled for all object's DACLs; the SACLs are left unchanged.

# Example 2 – Listing and Backup

```
SetACL.exe -on "\\server1\share1\users" -ot file -actn list
    -lst "f:sddl;w:d,s,o,g"
    -rec cont
    -bckp "d:\data\setacl listing.txt"
```

Creates a complete **listing** of DACL, SACL, owner and primary group in SDDL format of the directory '\\server1\share1\users' and all sub-folders. The listing is stored in unicode format in the **backup file** specified.

### Example 3 – Restore

```
SetACL.exe -on "dummy entry" -ot file -actn restore
        -bckp "d:\data\setacl listing.txt"
```

**Restores** all (!) security descriptor data (DACL, SACL, owner, primary group) from the backup file to its original location.

BEWARE: If you have the appropriate user rights (usually, being a member of the administrators group on the target system is sufficient) ALL data in the security descriptor is overwritten!

Comment: Only data contained in the backup file is overwritten, i.e. if you create a backup of the SACL only, when you restore it, the DACL, owner and primary group are left unchanged!

### Example 4 – Copying Permissions Between Users

```
SetACL.exe -on "\\server1\share1\users" -ot file -actn trustee
     -rec cont_obj
     -trst "n1:domain1\user1;n2:domain2\user2;ta:cpytrst;w:dacl"
```

This command **copies** all **ACEs** belonging to 'domain1\user1' to 'domain2\user2' resulting in a duplication of permissions: after the process domain2\user2 has the same permissions as domain1\user1. This might be useful in a migration scenario where users from domain1 are migrated (copied) to domain2.

# Example 5 – Migrating Permissions Between Domains

This is useful in a **domain migration** scenario where users from domain1 are migrated (copied) to domain2. This command replaces all SIDs belonging to users/groups from domain1 with SIDs of users/groups with the same names from domain2 resulting in a replacement of permissions: after the process domain2\user1 has the permissions domain1\user1 previously had.

### Example 6 – Removing all Permissions of a User

Removes "UserOrGroup" from the ACLs of all files on drive C:.