Programmatically Configuring Permissions on a Share

Rate This  
http://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.pnghttp://blogs.msdn.com/Utility/images/star-left-on.pnghttp://blogs.msdn.com/Utility/images/star-right-on.png

[http://i1.social.microsoft.com/profile/u/avatar.jpg?displayname=youhadmeathelloworld&size=largeYouHadMeAtHelloWorld](http://blogs.msdn.com/HelloWorld/ProfileUrlRedirect.ashx)

6 Jun 2008 1:38 PM

* Comments[33](http://blogs.msdn.com/b/helloworld/archive/2008/06/06/programmatically-configuring-permissions-on-a-share-in-c.aspx#comments)

I was asked this problem on how to set up permission for a share programmatically using .Net Framework. Well, I am not aware of any API that can do that. Searching does not return any good result. There are lot of resources on how to configure permission settings for local folder, but not so much for UNC path. At the end, I dug msdn and had my solution, using WMI.

To setup a share, you need these information, the share that you want to setup (securable or trustee), whom you will give the permissions to the share (principal), what kind of permissions you want to give.

Using this scenario, you have a share [\\ContosoServer\JohnShare](file:///\\contososerver\JohnShare), and you want John Doe (contoso\johndoe) to have full access to this share. The steps to configure the share permissions are as follow:

* Create a WMI instance of the principal (Win32\_Trustee).   
  //Getting the Sid value is not required for Vista.   
  NTAccount account = newNTAccount(Domain, UserName);   
  SecurityIdentifiersid = (SecurityIdentifier)account.Translate(typeof(SecurityIdentifier));   
  byte[] sidArray = new byte[sid.BinaryLength];   
  sid.GetBinaryForm(sidArray, 0);   
    
  ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);   
  Trustee["Domain"] = "contoso";   
  Trustee["Name"] = "johndoe";   
  Trustee["SID"] = sidArray;
* Create a WMI instance of Win32\_Ace, assign the Trustee to this Win32\_Ace instance.   
  ManagementObject AdminACE = new ManagementClass(new ManagementPath("Win32\_Ace"), null);   
  AdminACE["AccessMask"] = 2032127; // 0x1f01ff  
  AdminACE["AceFlags"] = 3;   
  AdminACE["AceType"] = 0;   
  AdminACE["Trustee"] = Trustee;   
    
  To know what values you need to put there, check msdn ([link](http://msdn.microsoft.com/en-us/library/aa394063(VS.85).aspx)). I actually encourage you to write an enum flag to encapsulate those values.   
  In nut shell, 2032127 is for full access, Access Flags 3 is for non-container and container child objects to inherit the ACE, and Ace Type 0 is to allow the trustee to access it.
* Create a WMI instance of the security descriptor (Win32\_SecurityDescriptor)   
  ManagementObject secDescriptor = new ManagementClass(new ManagementPath("Win32\_SecurityDescriptor"), null);   
  secDescriptor["ControlFlags"] = 4; //SE\_DACL\_PRESENT   
  secDescriptor["DACL"] = new object[] { AdminACE};
* Now, create a WMI instance of Win32\_Share, and setup the security.   
  ManagementObject share = new ManagementObject(@"\\ContosoServer\root\cimv2:Win32\_Share.Name='JohnShare'");   
  share.InvokeMethod("SetShareInfo", new object[] {Int32.MaxValue, "This is John's share", secDescriptor});   
  Check the return value of the Invoke, the method returns an Object, convert it to Int32.

That code will overwrite the existing permission, so be careful. WMI stuff are available in System.Management assemblies.

For references, these are the links that you will be interested with, [Win32\_Trustee](http://msdn.microsoft.com/en-us/library/aa394501(VS.85).aspx), [Win32\_ACE](http://msdn.microsoft.com/en-us/library/aa394063(VS.85).aspx), [Win32\_SecurityDescriptor](http://msdn.microsoft.com/en-us/library/aa394402(VS.85).aspx), and [Win32\_Share](http://msdn.microsoft.com/en-us/library/aa394435.aspx).

*Update (6/9/2008)*

I updated the first step with the code to assign Sid, thanks to **David Smith** for his email. With Windows Vista, you do not need to supply Sid. You can supply just the domain name and the user name, it will work. Using Server 2003, and most likely XP, you have to supply all three, user name, domain, and Sid.

* [33 Comments](http://blogs.msdn.com/b/helloworld/archive/2008/06/06/programmatically-configuring-permissions-on-a-share-in-c.aspx#comments)



|  |  |
| --- | --- |
|  |  |

8 Jun 2008 7:32 AM

Wow! I can't tell you how grateful I am to have found your How-To. I've been banging my head against the wall on this all morning and the docs have been of little help. I did follow the link and create an enum for AccessMask, but none of the listed values are 2032127. Is this the sum of several values?

8 Jun 2008 7:51 AM

Just to clarify, I realize the docs say you have to add them together, but coming up with the right combination (as you did here) requires a few more steps.

Thanks for your feedback about enum value 2032127, you are correct, that value is sum of several enum values. I am going to write another post on that next week.

8 Jun 2008 1:30 PM

The code you provided for configuring the trustee works for local accounts by not for domain accounts. To create both local and domain permissions, I set the SID as follows, where domain and accountName are strings for the obvious values.

NTAccount ntAccount = new NTAccount(domain, accountName);

SecurityIdentifier sid = (SecurityIdentifier) ntAccount.Translate(typeof(SecurityIdentifier));

byte[] sidArray = new byte[sid.BinaryLength];

sid.GetBinaryForm(sidArray, 0);

ManagementObject trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);

trustee["SID"] = sidArray;

9 Jun 2008 2:36 PM

Sid is required when the code is executed on Windows Server 2003, Vista does not require Sid property to be assigned.

On Server 2003, you have to supply all, only Sid won't assign the permission correctly.

9 Jun 2008 7:31 PM

In my previous post , I have shown you how to modify share permission using .Net framework. Access Mask

26 Jun 2008 5:54 AM

Hi

I have been able to CHANGE the EVERYONE to any user or group that I want but I am not being able to ADD another user/group to my share list.

example: Administrators Full controll and domainname\user full control. (not talking NTFS) just shares permissions.

I wonder if you could help me out.

26 Jun 2008 12:14 PM

Hello foxer,

I can try to help you out. Please be aware that SetShareInfo will overwrite the security descriptor, so you cannot really add permissions, you define a new security permission.

In order for you to define a full access for Admin and a domain user, what you have to do is to create two instances of Win32\_Ace objects like what I have shown you in this blog, let us call it AdminAce and UserAce.

When you create Win32\_SecurityDescriptor, create it like this:

ManagementObject secDescriptor = new ManagementClass(new ManagementPath("Win32\_SecurityDescriptor"), null);   
secDescriptor["ControlFlags"] = 4; //SE\_DACL\_PRESENT   
secDescriptor["DACL"] = new object[] { AdminAce, **UserAce**};

Pass both Win32\_Ace objects to the DACL property.

27 Jun 2008 3:11 AM

Hi Thanks

It worked. At last! I should have thought of it.

3 Jul 2008 3:55 PM

Great article!

I was interested in getting the list of User,Permission pairs, like in the Sharing->Permissions menu. I've tried using the 'GetAccessMask' in Win32\_Share, but that only shows your current computer's access. The idea I had was to get the list of all domain-user/group pairs, then impersonate and use GetAccessmask, but that seems awful. If you had any ideas, I'd be grateful :)

* [YouHadMeAtHelloWorld](http://blogs.msdn.com/HelloWorld/ProfileUrlRedirect.ashx)

8 Dec 2008 12:42 AM

Hi Bhargavi,

Each users must be represented by an ace. Create Trustee for each users, and then assign each trustee to its own ace.

Then when creating security descriptor, assign those aces to the DACL property. For example:

secDescriptor["DACL"] = new object[] { AdminACE, YourACE, MyAce, ThirdPersonAce };

* subtile

20 Feb 2009 10:02 AM

Great post!

I keep getting error in this line:

ManagementObject share = new ManagementObject(@"\\MyRemoteServer\root\cimv2:Win32\_Share.Name='TestShare'");

If not my share name TestShare is equals to the shared folder name.

Any ideas how to solve this problem ?

Best Regards subtile

20 Feb 2009 11:13 AM

@Subtile: What is the error? Is the share on a different machine or on the same machine?

25 Mar 2009 10:31 PM

Hi!

Thanks for the post, this is really helpful. However, I'm having some trouble in the following scenario:

In a home network, user A and B are in the same workgroup, but on different machine. A wants to allow user B (on a different machine) to share a folder residing on user A's machine.

Is there a way to do this? Both machines run windows XP, with only workgroups, no domains.

--kim

2 Apr 2009 5:08 PM

@Kim.

Give this a try.

B must have an account on machine A, and B must be an admin. B's account on both machines must have identical password.

7 Jul 2009 3:57 PM

Hi

Could you tell me how to share a folder to everyone instead of a specific user?

Thanks!

* [http://i1.social.microsoft.com/profile/u/avatar.jpg?displayname=youhadmeathelloworld&size=largeYouHadMeAtHelloWorld](http://blogs.msdn.com/HelloWorld/ProfileUrlRedirect.ashx)

7 Jul 2009 5:15 PM

@David B:

If you need Everyone to have read-only access, then what you need to do is to set DACL to null.

secDescriptor["DACL"] = null;

This will give everyone a read access.

If you need to give 'Everyone' read-write access, then you have to create the Win32\_Trustee, using WellKnownSidType enum. I need to check my note again about this. I will put this for my next blog post.

Give this a try, when creating Trustee object, try to use 'NT Authority\Everyone'

ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);

Trustee["Domain"] = "NT Authority";

Trustee["Name"] = "Everyone";

8 Jul 2009 9:45 AM

@HelloWorld

Thanks for the reply. This works on my Vista machine, but it doesn't work on Windows Server 2003. Any thoughts? Here is what I have for code.

protected void CreateShare(string path, string shareName)  
{  
// Create a ManagementClass object  
ManagementClass managementClass = new ManagementClass("Win32\_Share");  
// Create ManagementBaseObjects for in and out parameters  
ManagementBaseObject inParams = managementClass.GetMethodParameters("Create");  
ManagementBaseObject outParams;  
// Set the input parameters  
inParams["Description"] = "IdeaServer";  
inParams["Name"] = shareName;  
inParams["Path"] = path;  
inParams["Type"] = 0x0; // Disk Drive  
//Another Type:  
// DISK\_DRIVE = 0x0  
// PRINT\_QUEUE = 0x1  
// DEVICE = 0x2  
// IPC = 0x3  
// DISK\_DRIVE\_ADMIN = 0x80000000  
// PRINT\_QUEUE\_ADMIN = 0x80000001  
// DEVICE\_ADMIN = 0x80000002  
// IPC\_ADMIN = 0x8000003  
//inParams["MaximumAllowed"] = int maxConnectionsNum;  
// Invoke the method on the ManagementClass object  
inParams["Access"] = SecurityDescriptor();  
outParams = managementClass.InvokeMethod("Create", inParams, null);  
// Check to see if the method invocation was successful  
uint rVal = (uint)(outParams.Properties["ReturnValue"].Value);  
if (rVal != 0 && rVal != 22) // ok if it already exists  
{  
throw new Exception("Unable to share directory.");  
}  
}

private static ManagementBaseObject SecurityDescriptor()  
{  
NTAccount account = new NTAccount("NT Authority", "Everyone");  
SecurityIdentifier sid = (SecurityIdentifier)account.Translate(typeof(SecurityIdentifier));  
byte[] sidArray = new byte[sid.BinaryLength];  
sid.GetBinaryForm(sidArray, 0);   
ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);  
Trustee["Domain"] = "NT Authority";  
Trustee["Name"] = "Everyone";  
Trustee["SID"] = sidArray;  
ManagementObject AdminACE = new ManagementClass(new ManagementPath("Win32\_Ace"), null);  
AdminACE["AccessMask"] = 2032127; //or 0x1f01ff  
AdminACE["AceFlags"] = 3;  
AdminACE["AceType"] = 0;  
AdminACE["Trustee"] = Trustee;  
ManagementObject SecurityDescriptor = new ManagementClass(new ManagementPath("Win32\_SecurityDescriptor"), null);  
SecurityDescriptor["ControlFlags"] = 4; //SE\_DACL\_PRESENT  
SecurityDescriptor["DACL"] = new object[] { AdminACE };  
return SecurityDescriptor;  
}

Thanks a lot!

David

* [http://i1.social.microsoft.com/profile/u/avatar.jpg?displayname=youhadmeathelloworld&size=largeYouHadMeAtHelloWorld](http://blogs.msdn.com/HelloWorld/ProfileUrlRedirect.ashx)

9 Jul 2009 1:17 PM

@David:

Try this, update your code from

ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);

Trustee["Domain"] = "NT Authority";

Trustee["Name"] = "Everyone";

Trustee["SID"] = sidArray;

to

SecurityIdentifier Sec = new SecurityIdentifier(System.Security.Principal.WellKnownSidType.WorldSid, null);

byte[] sidArray = new byte[Sec.BinaryLength];

Sec.GetBinaryForm(sidArray, 0);

ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);

Trustee["SID"] = sidArray;

14 Jul 2009 10:56 AM

@HelloWorld

Thanks! That worked perfectly!

29 Aug 2009 10:26 PM

How do you update the share permissions on an already created share? For example add a new user to the share etc

29 Aug 2009 10:58 PM

@Aditya:

Please check my other post. <http://blogs.msdn.com/helloworld/archive/2008/07/22/editing-share-permission.aspx>

29 Apr 2010 7:17 AM

Thank you SO much for posting this! I couldn't find an example of how to do this for Everyone, but you covered that for another reading in your comments.

Cheers!

-RG!

26 May 2010 4:17 PM

Just wanted to point out that there is a Windows API for creating shared folders and specifying their permissions, for anyone who does not want to use WMI. I've spent the last 2 days trying to get it to work from a VB.NET app and have finally been successful - see this thread on vbforums for more details: [www.vbforums.com/showthread.php](http://www.vbforums.com/showthread.php?t=615648)

Hope it is useful to someone

Chris

* Brad Garrison

12 Jul 2010 3:09 PM

Here is the breakdown for the access mask. Add them together as needed.

FOLDER\_LIST\_DIRECTORY = 1

FOLDER\_ADD\_FILE = 2

FOLDER\_ADD\_SUBDIRECTORY = 4

FILE\_READ\_EA = 8

FILE\_WRITE\_EA = 16

FOLDER\_TRAVERSE = 32

FILE\_DELETE\_CHILD = 64

FILE\_READ\_ATTRIBUTES = 128

FILE\_WRITE\_ATTRIBUTES = 256

FILE\_DELETE = 65536

FILE\_READ\_CONTROL = 131072

FILE\_WRITE\_DAC = 262144

FILE\_WRITE\_OWNER = 524288

FILE\_SYNCHRONIZE = 1048576

FILE\_ALL\_ACCESS = 2032127

In my previous post, I have shown you [how to set up permission on a share](http://blogs.msdn.com/helloworld/archive/2008/06/06/programmatically-configuring-permissions-on-a-share-in-c.aspx). The thing with Win32\_Share, when you set the permission, you basically overwrites the existing permission.

If you want to edit permission on the share (grant a new user access to the share, or revoke an existing user's permission), then you have to get the security descriptor for that share, and modify it, and then call [Win32\_Share.SetShareInfo](http://msdn.microsoft.com/en-us/library/aa393598(VS.85).aspx) to set the share permission.

To get security descriptor of a share, you can use [Win32\_LogicalShareSecuritySetting](http://msdn.microsoft.com/en-us/library/aa394188.aspx) class. Then update the security descriptor and set that security descriptor back to the share.

When calling ManagementObject.GetSecurityDescriptor, it will return a ManagementBaseObject instance, it has two properties, ReturnValue and Descriptor. ReturnValue is an integer value, that tells you whether the operation is successful or not. Look for the possible value [here](http://msdn.microsoft.com/en-us/library/aa390773(VS.85).aspx). The Descriptor property is an instance of SecurityDescriptor.

To summarize (for those who love bullet points):

* Get the Win32\_Ace instance for the new user.
* Get the current security descriptor.
* Get the DACL (Array of Win32\_Ace) from the security descriptor.
* **Add** the Win32\_Ace for the new user into the Win32\_Ace array.
* Reassign the edited DACL back to the security descriptor.
* Call Win32\_Share.SetShareInfo to set the permission.

You can delete a particular user, or changing the existing permission, by modifying the DACL or SACL in the Security Descriptor.

This snippet below is just an example on how to read, modify and assign permission on a share, this code was derived from the example on my previous [post](http://blogs.msdn.com/helloworld/archive/2008/06/06/programmatically-configuring-permissions-on-a-share-in-c.aspx).

//Create a new Win32\_Ace instance. Please refer to my previous post about creating Win32\_Ace.

NTAccount account = new NTAccount("contoso", "janedoe");

SecurityIdentifier sid = (SecurityIdentifier)account.Translate(typeof(SecurityIdentifier));

byte[] sidArray = new byte[sid.BinaryLength];

sid.GetBinaryForm(sidArray, 0);

ManagementObject Trustee = new ManagementClass(new ManagementPath("Win32\_Trustee"), null);

Trustee["Domain"] = "contoso";

Trustee["Name"] = "janedoe";

Trustee["SID"] = sidArray;

ManagementObject ACE = new ManagementClass(new ManagementPath("Win32\_Ace"), null);

ACE["AccessMask"] = 2032127;

ACE["AceFlags"] = 3;

ACE["AceType"] = 0;

ACE["Trustee"] = Trustee;

//After we have the new Win\_32Ace, now we need to get the existing Ace instances (DACL).

//Create an instance of Win32\_LogicalSecuritySetting, set the path to the server and the share.

ManagementObject Win32LogicalSecuritySetting = new ManagementObject(@"\\ContosoServer\root\cimv2:Win32\_LogicalShareSecuritySetting.Name='JohnShare'");

//Call the GetSecurityDescriptor method. This method returns one out parameter.

ManagementBaseObject Return = Win32LogicalSecuritySetting.InvokeMethod("GetSecurityDescriptor", null, null);

//The return value of that call above has two properties, ReturnValue, which you can use

//to read the status of the call (failed, success, etc.), and Descriptor, which is an instance

//of Win32\_SecurityDescriptor.

Int32 ReturnValue = Convert.ToInt32(Return.Properties["ReturnValue"].Value);

if (ReturnValue != 0)

throw new Exception(String.Format("Error when calling GetSecurityDescriptor. Error code : {0}.", ReturnValue));

//Retrieve the array of DACL from the Security Descriptor.

ManagementBaseObject SecurityDescriptor = Return.Properties["Descriptor"].Value as ManagementBaseObject;

ManagementBaseObject[] DACL = SecurityDescriptor["DACL"] as ManagementBaseObject[];

if (DACL == null)

DACL = new ManagementBaseObject[] { ACE };

else

{

Array.Resize(ref DACL, DACL.Length + 1);

DACL[DACL.Length - 1] = ACE;

}

//Reassign the new DACL array with the new user Ace back to the Win32\_SecurityDescriptor instance, and call the

//SetSecurityDescriptor method.

SecurityDescriptor["DACL"] = DACL;

ManagementObject Share = new ManagementObject(@"\\ContosoServer\root\cimv2:Win32\_Share.Name='JohnShare'");

ReturnValue = Convert.ToInt32(Share.InvokeMethod("SetShareInfo", new object[] {Int32.MaxValue, "This is John's share", SecurityDescriptor}));

if (ReturnValue != 0)

throw new Exception(String.Format("Error when calling GetSecurityDescriptor. Error code : {0}.", ReturnValue));