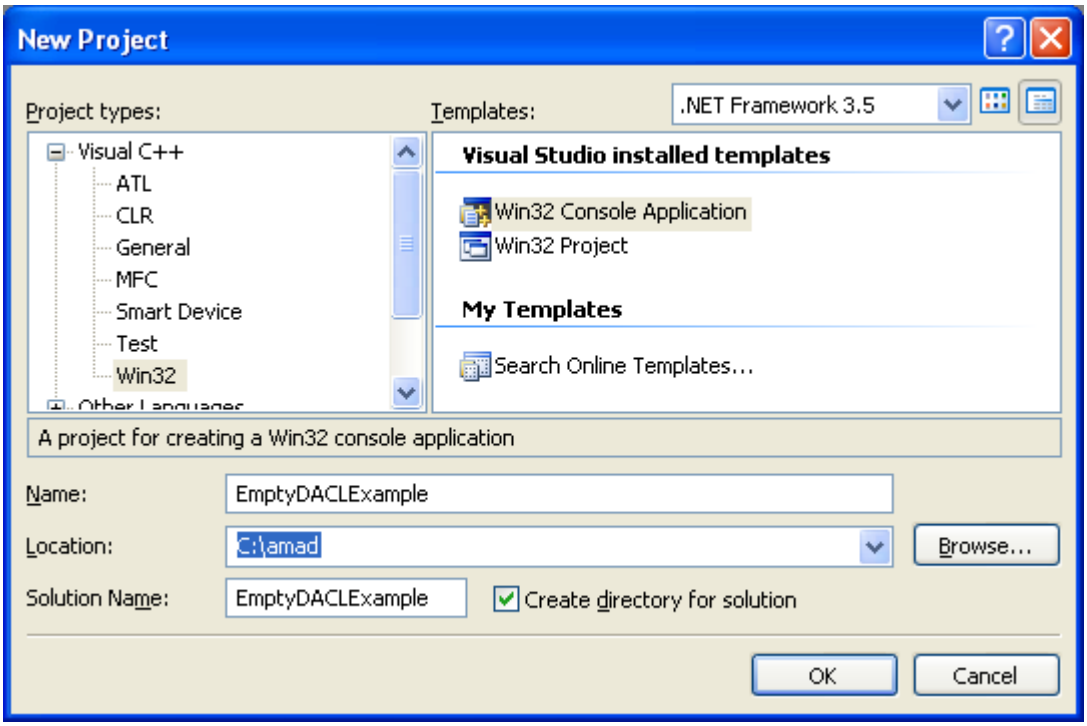
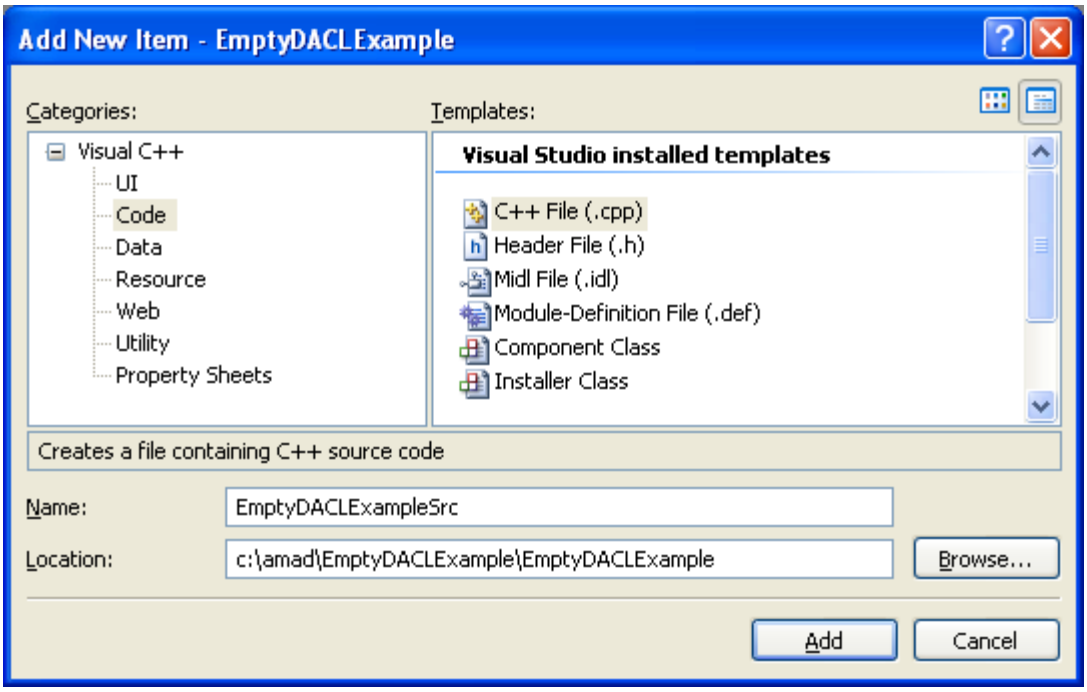
1. An Empty DACL program example
2. The NULL DACL program example
3. Modifying existing DACLs of an Object program example
4. Modifying the SACL and Privilege Program Example

An Empty DACL Program Example

The following program example demonstrates creating an empty DACL.
Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.

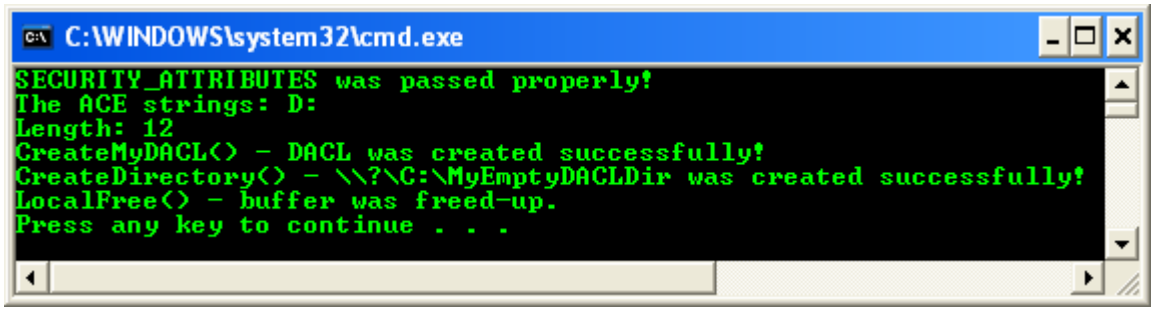


Next, add the following source code.

```
// An empty DACL program example
// #define _WIN32_WINNT 0x0500
#include <windows.h>
#include <sddl.h>
#include <stdio.h>

// Prototype
BOOL CreateMyDACL(SECURITY_ATTRIBUTES *);

int main(int argc, WCHAR **argv)
{
     SECURITY_ATTRIBUTES  sa;
     // The SECURITY_ATTRIBUTE structure size
     sa.nLength = sizeof(SECURITY_ATTRIBUTES);
     // The return handle not inherited
     sa.bInheritHandle = FALSE;
       // Directory that will be assigned the empty DACL
       WCHAR DirName[] = L"\\\\?\\C:\\MyEmptyDACLDir";

       // Call CreateMyDACL() function to set the DACL. The DACL
     // is set in the SECURITY_ATTRIBUTES
     // lpSecurityDescriptor member
       if(!CreateMyDACL(&sa))
     {
         //Error encountered; generate message and just exit.
         wprintf(L"CreateMyDACL() failed, error %d\n", GetLastError());
         exit(1);
     }
     else
         printf("CreateMyDACL() - DACL was created successfully!\n");
```

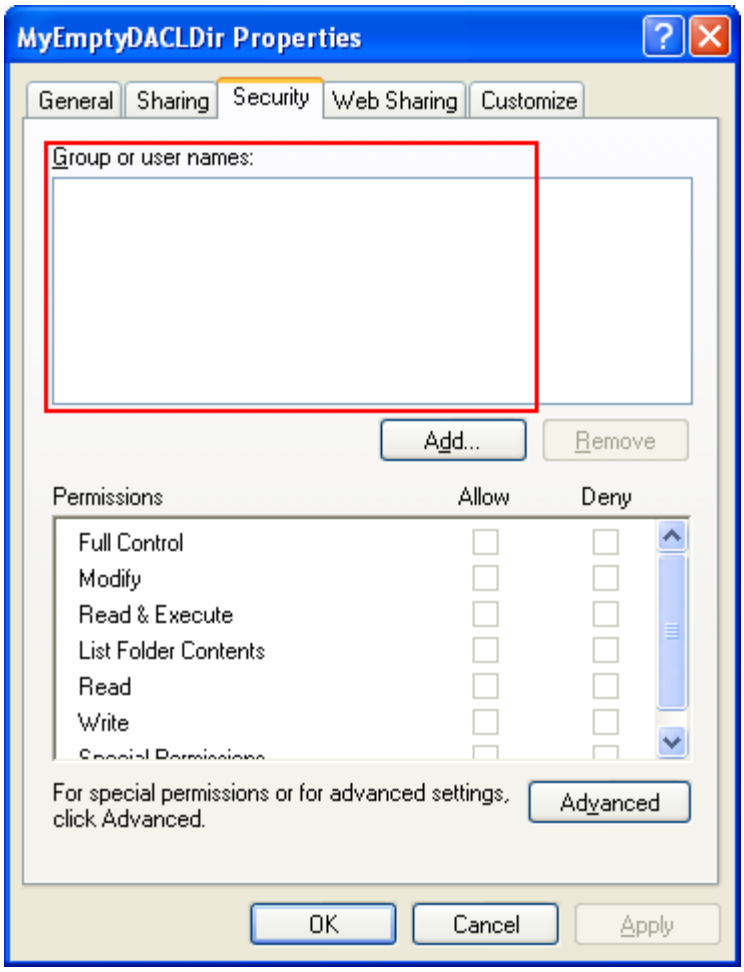
```
    // Use the updated SECURITY_ATTRIBUTES to specify
    // security attributes for securable objects.
    // This example uses security attributes during
    // creation of a new directory.
    if(CreateDirectory(DirName, &sa) == 0)
    {
        // If error encountered; generate message and exit.
        wprintf(L"failed to create %s directory!, error %u\n", DirName,
GetLastError());
        exit(1);
    }
    else
        wprintf(L"CreateDirectory() - %s was created successfully!\n",
DirName);

    // Release the memory allocated for the SECURITY_DESCRIPTOR.
    if(LocalFree(sa.lpSecurityDescriptor) != NULL)
    {
        // Error encountered; generate message and exit.
        wprintf(L"LocalFree() failed, error %u.\n", GetLastError());
        exit(1);
    }
    else
       printf("LocalFree() - buffer was freed-up.\n");
      return 0;
}

// Create a security descriptor that contains the DACL you want.
BOOL CreateMyDACL(SECURITY_ATTRIBUTES * pSA)
{
     PULONG nSize = 0;
     // An empty DACL
     WCHAR * szSD = L"D:";

     if(pSA == NULL)
          return FALSE;
     else
          wprintf(L"SECURITY_ATTRIBUTES was passed properly!\n");

     // Do some verification
     wprintf(L"The ACE strings: %s \n", szSD);
     wprintf(L"Length: %u\n", pSA->nLength);

     // Convert the string to the security descriptor binary and return
     return ConvertStringSecurityDescriptorToSecurityDescriptor(
          szSD,                                        // The ACE strings
          SDDL_REVISION_1,                      // Standard revision level
          &(pSA->lpSecurityDescriptor), // Pointer to the converted
security descriptor
          nSize);                                              // The size in
byte the converted security descriptor
}
```

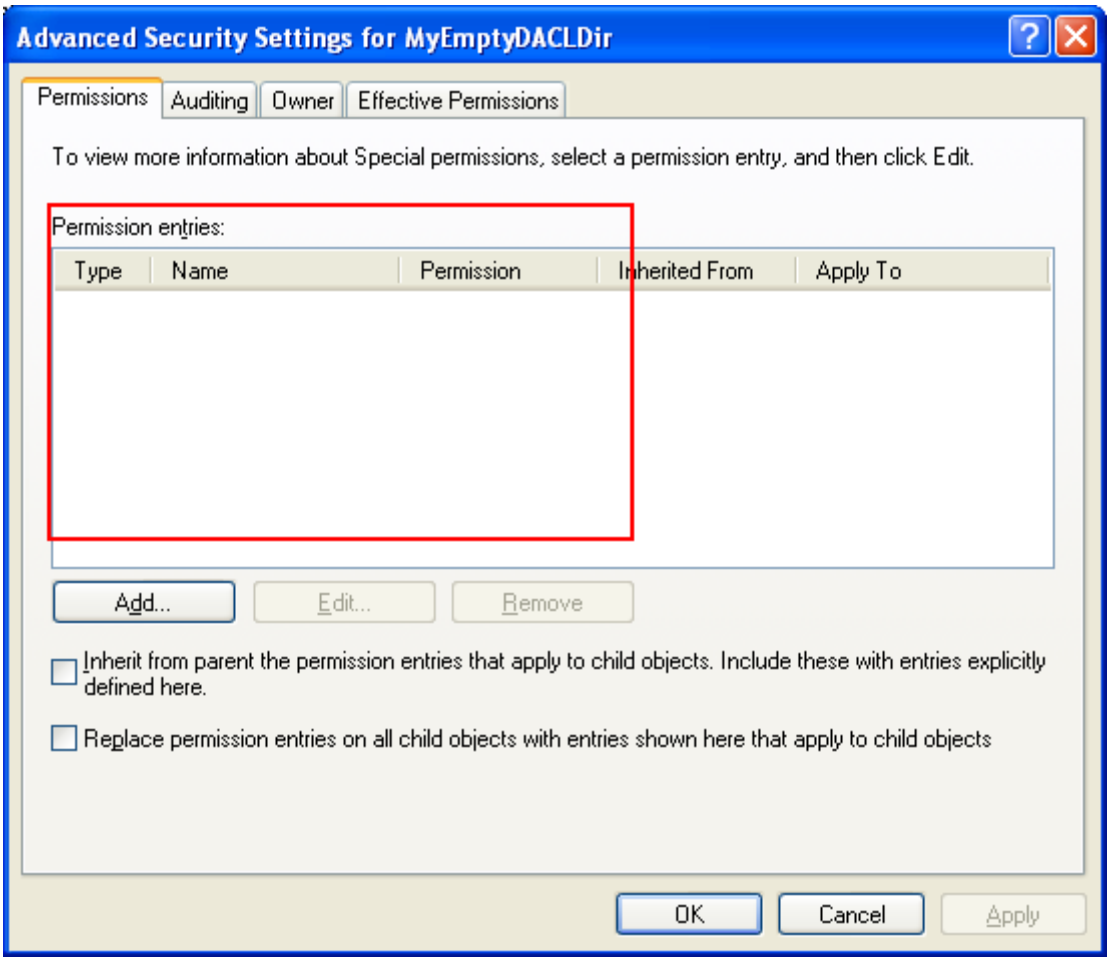
Build and run the project. The following screenshot is a sample output.



In this case, when user Mike spoon (a member of an Administrators group) tries to open (or delete) the `C:\\MyEmptyDACLDir` directory, the following message was displayed.

When we verify through the `MyEmptyDACLDir`'s property page, there is no ACE at all. Well, do not create an empty DACL.


MyEmptyDACLDir Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Add...
Remove
Permissions
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply



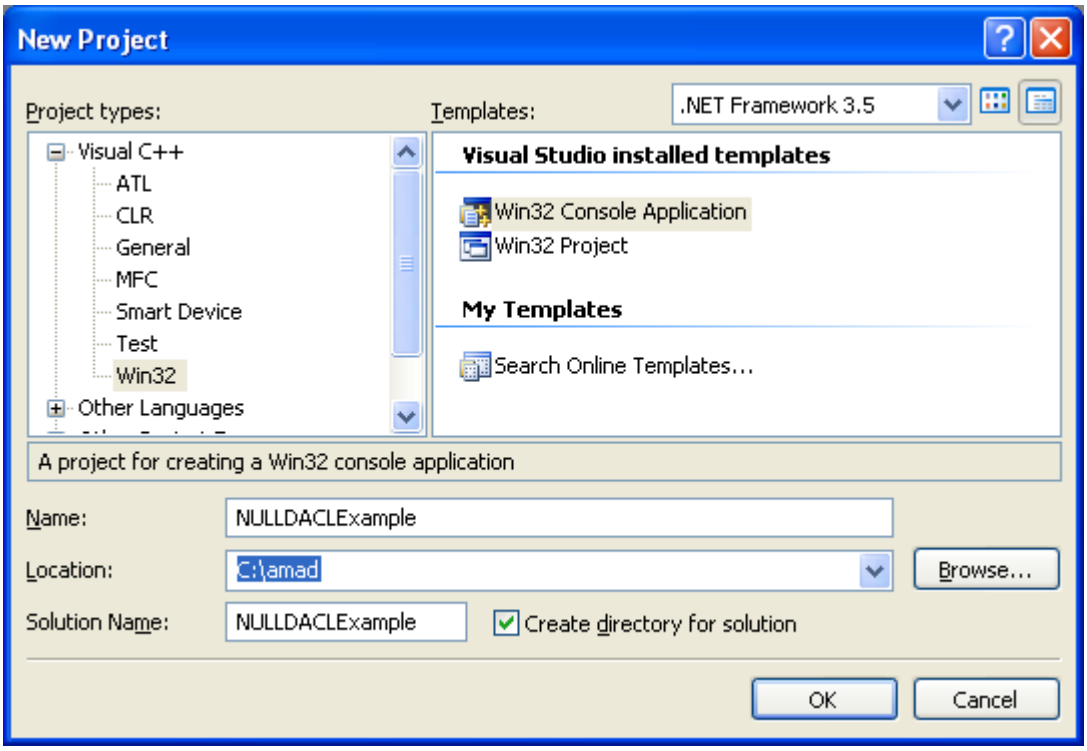
By the way, Administrator user (also any user which is a member of Administrators group) still has the permission to modify the permission (use the Add button to add the permission) or he/she can take the ownership of this directory object.

The NULL DACL Program Example: Everyone get Full Control

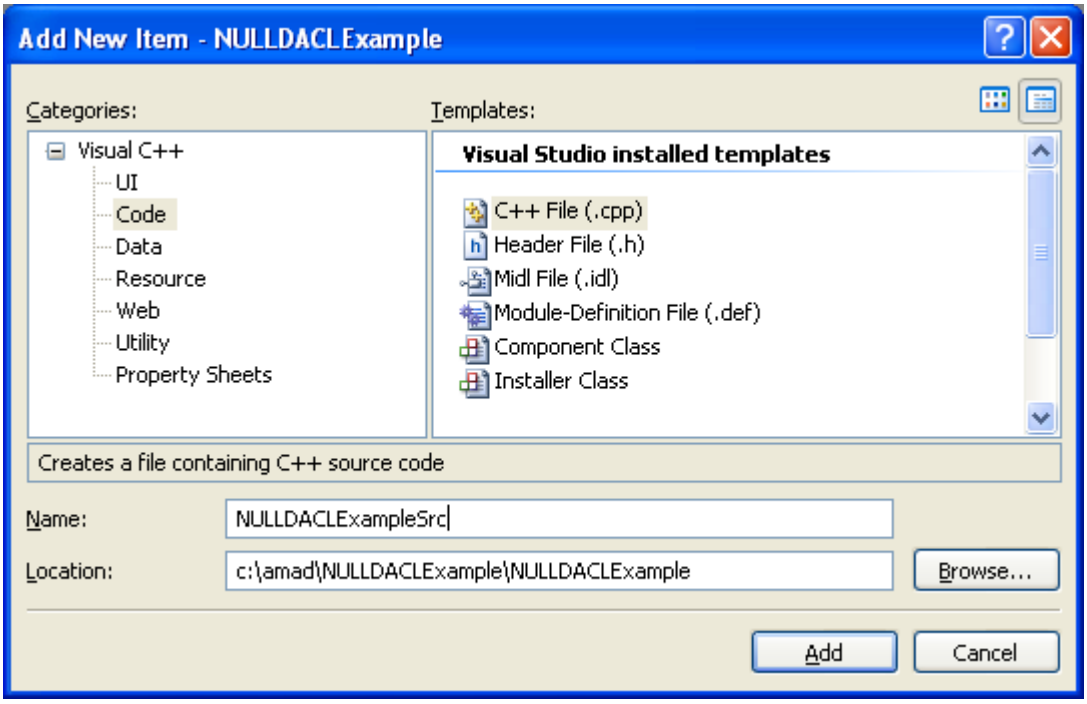
The following program example demonstrates how to create a NULL DACL. Previously we can assign NULL DACL directly through CreateDirectory(),

```
CreateDirectory(pathname, NULL);
```

However it will fail because it new directory object will inherit the parent security descriptor. By default, the Everyone group also no longer includes anonymous users on a computer that is running Windows XP Service Pack 2 (SP2).
Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.

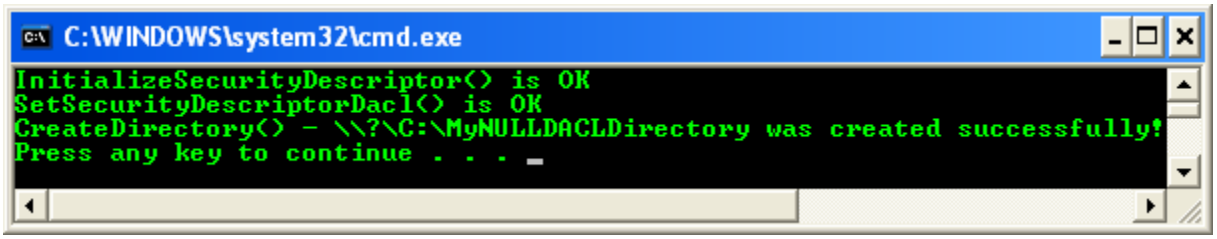


Next, add the following source code.

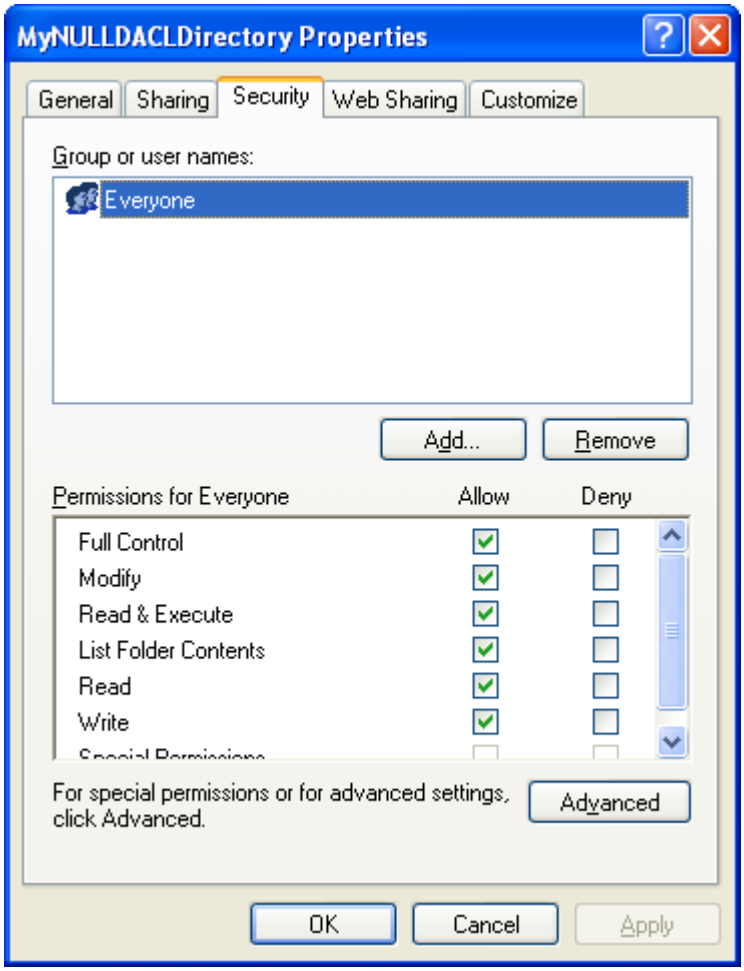
```
#include <windows.h>
#include <stdio.h>

int wmain(int argc, WCHAR **argv)
{
	// The securable object to be created
	WCHAR DirName[] = L"\\\\?\\C:\\MyNULLDACLDirectory";
	// declare and initialize a security attributes structure
	SECURITY_ATTRIBUTES sa;
	// Same as SECURITY_ATTRIBUTES sa = {0};
	ZeroMemory(&sa, sizeof(sa));
	sa.nLength = sizeof(sa);
	// Object handle uninheritable
	sa.bInheritHandle = FALSE;
	BOOL bInitOk = FALSE;
	BOOL bSetOk = FALSE;

	// declare a security descriptor
	SECURITY_DESCRIPTOR SD;

	// initializes a new security descriptor. The
InitializeSecurityDescriptor()
	// function initializes a security descriptor to have no system access
control list (SACL),
	// no discretionary access control list (DACL), no owner, no primary
group,
	// and all control flags set to FALSE (NULL). Thus, except for its
revision level, it is empty.
	bInitOk = InitializeSecurityDescriptor(&SD,
SECURITY_DESCRIPTOR_REVISION);

	if(bInitOk)
	{
		wprintf(L"InitializeSecurityDescriptor() is OK\n");
		// sets information in a discretionary access control list
(DACL).
		// If a DACL is already present in the security descriptor, the
DACL is replaced.
		// give the security descriptor a Null Dacl
		// done using the  "TRUE, (PACL)NULL" here
		bSetOk = SetSecurityDescriptorDacl(&SD, TRUE,(PACL)NULL, FALSE);

		if (bSetOk)
		{
			wprintf(L"SetSecurityDescriptorDacl() is OK\n");
			// Make the security attributes point
			// to the security descriptor
			sa.lpSecurityDescriptor = &SD;

			// Then create a directory with the NULL security
descriptor
			if(CreateDirectory(DirName, &sa) == 0)
			{
				// Error encountered; generate message and exit
				wprintf(L"Failed to create %s directory! Error %u\n",
DirName, GetLastError());
				// Just exit
```

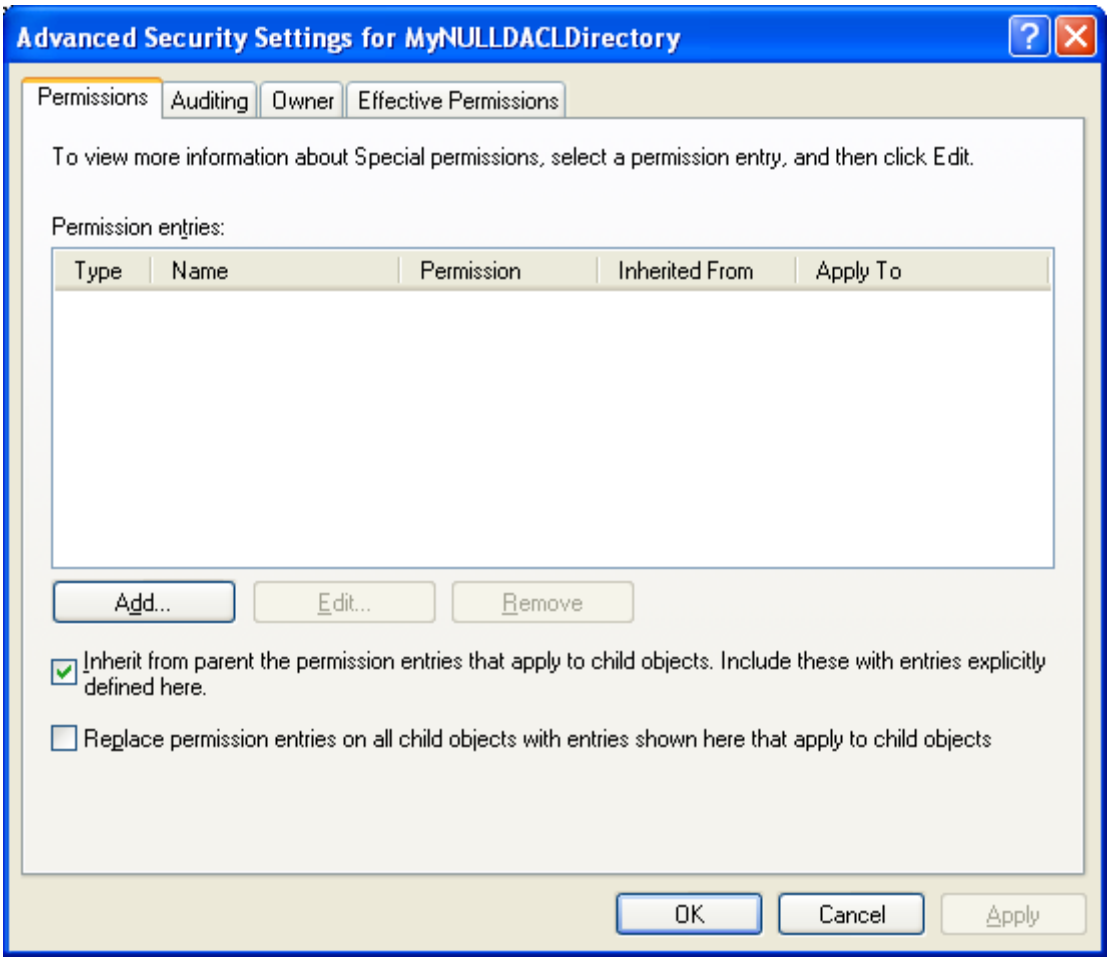
```
                        exit(1);
                }
                else
                        wprintf(L"CreateDirectory() - %s was created
successfully!\n", DirName);
            }
            else
                wprintf(L"SetSecurityDescriptorDacl() failed, error %u\n",
GetLastError());
      }
      else
            wprintf(L"InitializeSecurityDescriptor() failed, error %u\n",
GetLastError());
/*
      // Release the memory allocated for the SECURITY_DESCRIPTOR.
      if(LocalFree(sa.lpSecurityDescriptor) != NULL)
      {
            // Error encountered; generate message and exit
            wprintf(L"sa.lpSecurityDescriptor NOT NULL!\n");
            wprintf(L"LocalFree() failed, error %u\n", GetLastError());
            exit(1);
      }
      else
            wprintf(L"LocalFree() - buffer was freed...\n");
*/

            return 0;
}

/*
There is an important difference between an empty and a nonexistent DACL.
When a DACL is empty, it contains no access control entries (ACEs);
therefore, no access rights are explicitly granted. As a result, access
to the object is implicitly denied.
When an object has no DACL (when the pDacl parameter is NULL), no protection
is assigned to the object, and all access requests are granted. To help
maintain
security, restrict access by using a DACL. There are three possible outcomes
in different configurations of the bDaclPresent flag and the pDacl parameter:

    * When the pDacl parameter points to a DACL and the bDaclPresent flag is
TRUE,
        a DACL is specified and it must contain access-allowed ACEs to allow
access to the object.
    * When the pDacl parameter does not point to a DACL and the bDaclPresent
flag is TRUE,
        a NULL DACL is specified. All access is allowed. You should not use a
NULL DACL with
        an object because any user can change the DACL and owner of the
security descriptor.
        This will interfere with use of the object.
    * When the pDacl parameter does not point to a DACL and the bDaclPresent
flag is FALSE,
        a DACL can be provided for the object through an inheritance or
default mechanism.
*/
```

Build and run the project. The following screenshot is a sample output.



When we verify through the `C:\MyNULLDACLDirectory`'s property page, Everyone has Full Control permission! It is very dangerous if misused.

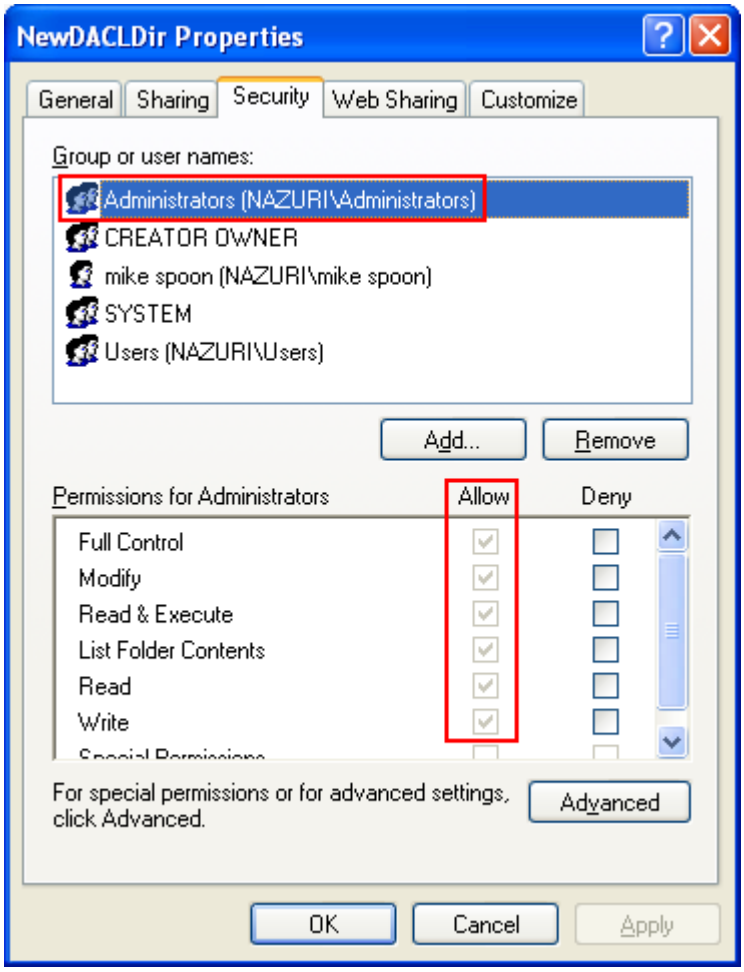


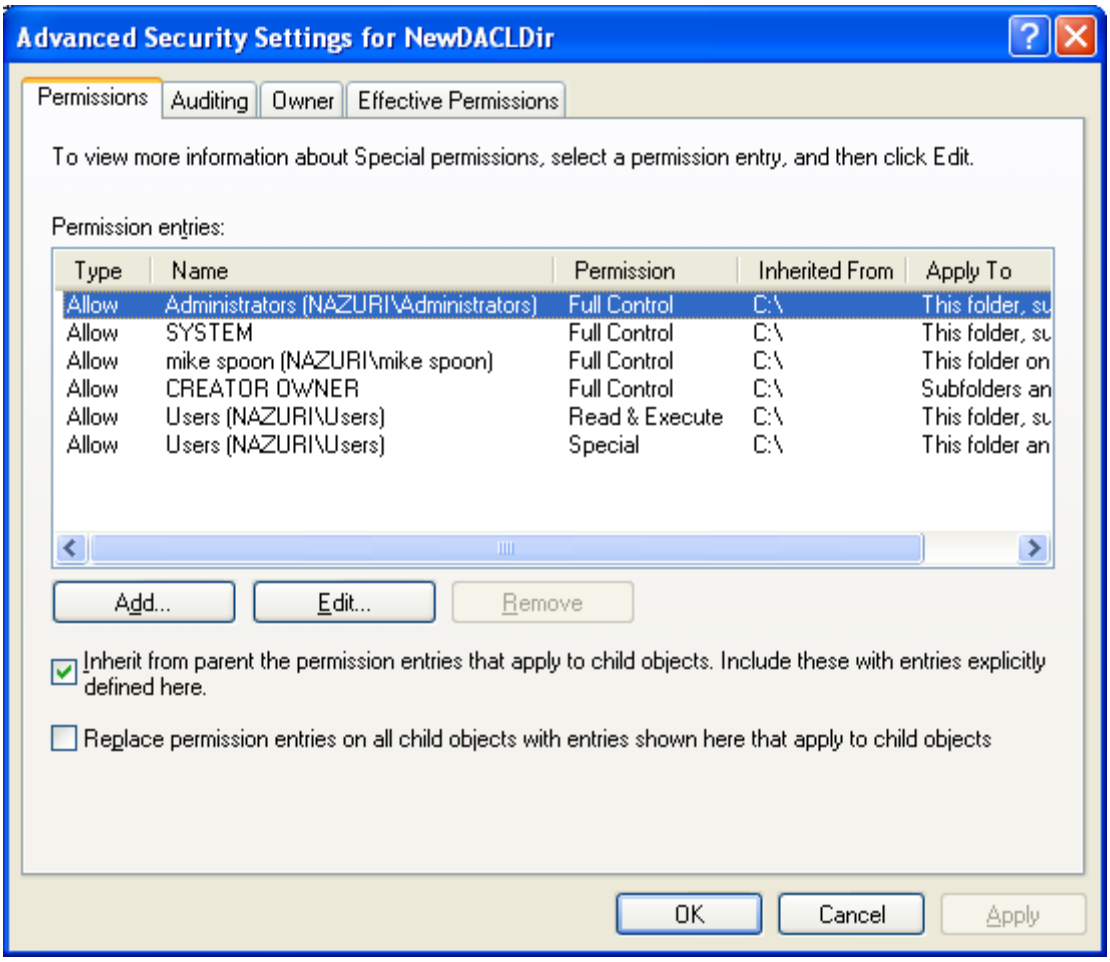


Modifying Existing DACLs of an Object Program Example

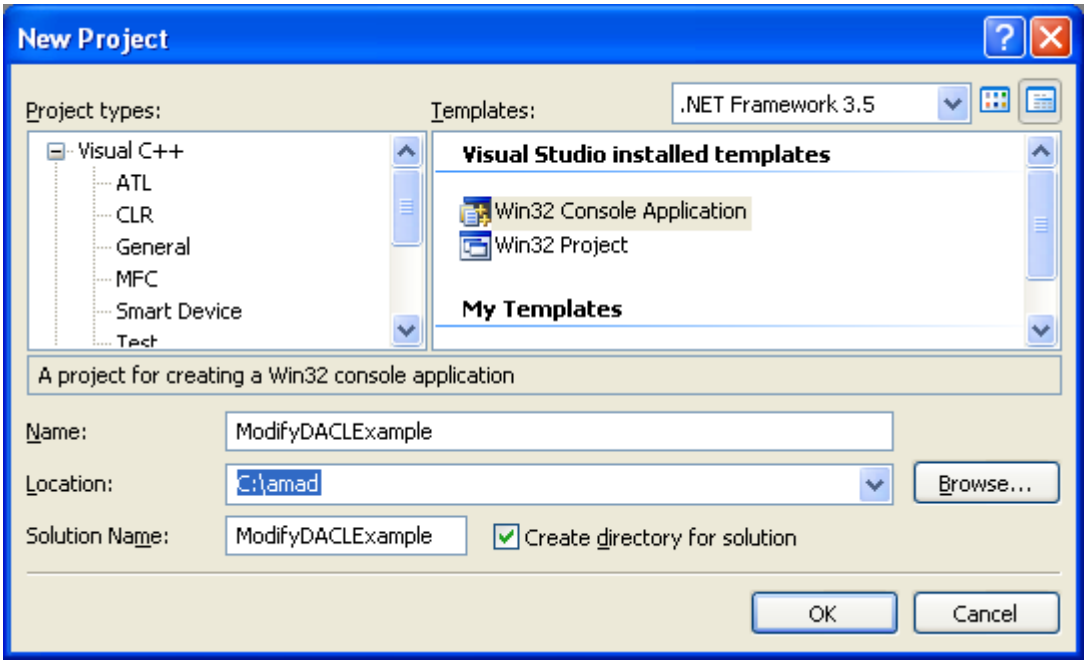
The following example adds an ACE to the DACL of an existing object. The AccessMode parameter, a member of the EXPLICIT_ACCESS structure determines the type of new ACE and how the new ACE is combined with any existing ACEs for the specified trustee. You can use the GRANT_ACCESS, SET_ACCESS, DENY_ACCESS, or REVOKE_ACCESS flags in the AccessMode parameter. Similar code can be used to work with a SACL. Specify SACL_SECURITY_INFORMATION in the GetNamedSecurityInfo() and SetNamedSecurityInfo() functions to get and set the SACL for the object. Use the SET_AUDIT_SUCCESS, SET_AUDIT_FAILURE, and REVOKE_ACCESS flags in the AccessMode parameter. This code example can be used to add an object-specific ACE to the DACL of a directory service object. To specify the GUIDs in an object-specific ACE, set the TrusteeForm parameter to TRUSTEE_IS_OBJECTS_AND_NAME or TRUSTEE_IS_OBJECTS_AND_SID and set the pszTrustee parameter to be a pointer to an OBJECTS_AND_NAME or OBJECTS_AND_SID structure. This example uses the GetNamedSecurityInfo() function to get the existing DACL. Then it fills an EXPLICIT_ACCESS structure with information about an ACE and uses the SetEntriesInAcl()

function to merge the new ACE with any existing ACEs in the DACL. Finally, the example calls the SetNamedSecurityInfo() function to attach the new DACL to the security descriptor of the object. The following figures are the NewDACLDir directory property pages before the program been run.

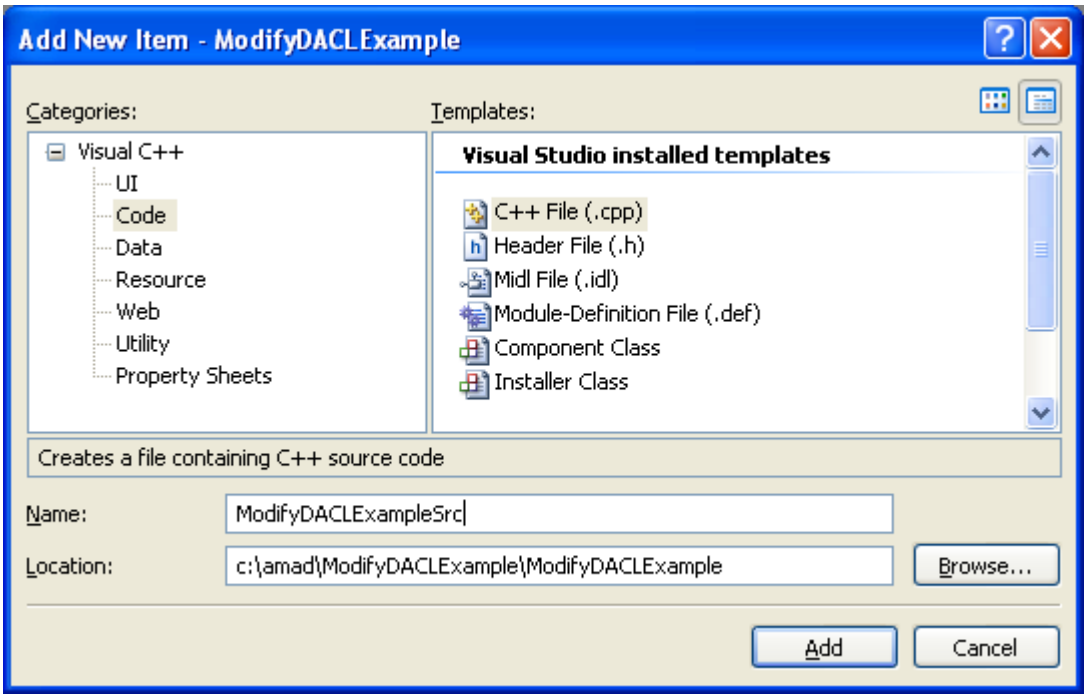




Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.



Next, add the following source code.

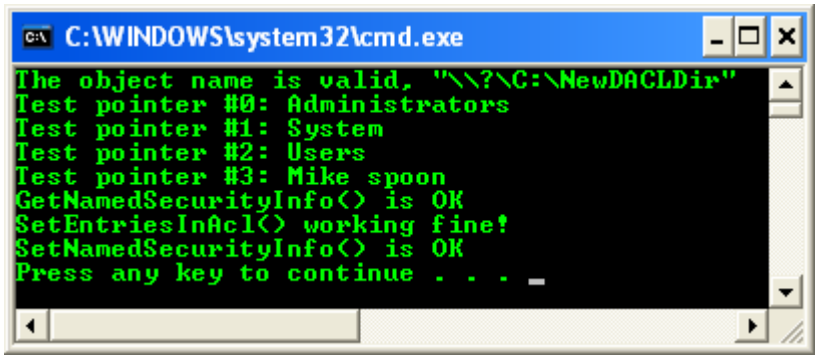
```
// Modifying DACL of an object. In ACL there are ACEs...
// Here we are going to add deny standard right
// access for Administrators group. This Win XP machine is
```

```
// logged in by user named Mike spoon who is a member of Administrators group
#include <windows.h>
#include <aclapi.h>
#include <stdio.h>

// Clean up the allocated resources
void Cleanup(PSECURITY_DESCRIPTOR pSD, PACL pNewDACL)
{
	if(pSD != NULL)
		LocalFree((HLOCAL) pSD);
	else
		wprintf(L"pSD freed up\n");
	if(pNewDACL != NULL)
		LocalFree((HLOCAL) pNewDACL);
	else
		wprintf(L"pNewDACL freed up\n");
}

int wmain(int argc, WCHAR **argv)
{
	// Name of object, here we will add ACE for a directory
	// the directory is already created
	LPTSTR pszObjName = L"\\\\?\\C:\\NewDACLDir";
	// Type of object, file or directory.  Here we test on directory
	SE_OBJECT_TYPE ObjectType = SE_FILE_OBJECT;
	// Access mask for new ACE equal to 0x001F0000 flags (bit 0 till 15)
	DWORD dwAccessRights = STANDARD_RIGHTS_ALL;
	// Type of ACE, Access denied ACE
	ACCESS_MODE AccessMode = DENY_ACCESS;
	// Inheritance flags for new the ACE. The OBJECT_INHERIT_ACE and
	// CONTAINER_INHERIT_ACE flags are
	// not propagated to an inherited ACE.
	DWORD dwInheritance = NO_PROPAGATE_INHERIT_ACE;
	// format of trustee structure, the trustee is name
	TRUSTEE_FORM TrusteeForm = TRUSTEE_IS_NAME;

	// Trustee for new ACE.  This just for fun...When you run once, only
one
	// element will take effect.  By changing the first array element we
	// can change to other trustee and re run the program....
	// Other than Mike spoon, they are all well known trustees
	// Take note the localization issues
	WCHAR pszTrustee[4][15] = {L"Administrators", L"System", L"Users",
L"Mike spoon"};

	// Result
	DWORD dwRes = 0;
	// Existing and new DACL pointers...
	PACL pOldDACL = NULL, pNewDACL = NULL;
	// Security descriptor
	PSECURITY_DESCRIPTOR pSD = NULL;
	SecureZeroMemory(&pSD, sizeof(PSECURITY_DESCRIPTOR));
	// EXPLICIT_ACCESS structure.  For more than one entries,
	// declare an array of the EXPLICIT_ACCESS structure
	EXPLICIT_ACCESS ea;

	// Verify the object name validity
```

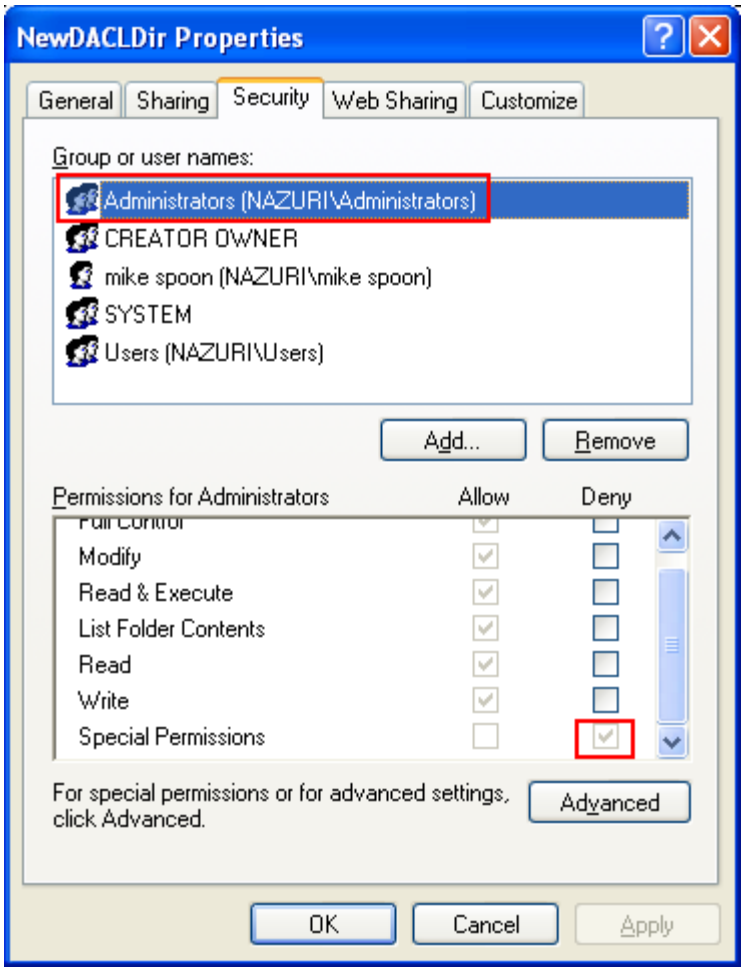
```
if(pszObjName == NULL)
{
	wprintf(L"The object name is invalid!\n");
	return ERROR_INVALID_PARAMETER;
}
else
	wprintf(L"The object name is valid, \"%s\"\n", pszObjName);

// Verify that our new trustee strings is OK
for(int i = 0; i <= 3; i++)
	wprintf(L"Test pointer #%d: %s\n", i, pszTrustee[i]);

// Get a pointer to the existing DACL.
dwRes = GetNamedSecurityInfo(pszObjName, ObjectType,
	DACL_SECURITY_INFORMATION,
	NULL,
	NULL,
	&pOldDACL,
	NULL,
	&pSD);

// Verify
if(dwRes != ERROR_SUCCESS)
{
	wprintf(L"GetNamedSecurityInfo() failed, error %u\n", dwRes);
	Cleanup(pSD, pNewDACL);
}
else
	wprintf(L"GetNamedSecurityInfo() is OK\n");

// Initialize an EXPLICIT_ACCESS structure for the new ACE.
// For more entries, declare an array of the EXPLICIT_ACCESS structure
SecureZeroMemory(&ea, sizeof(EXPLICIT_ACCESS));
ea.grfAccessPermissions = dwAccessRights;
ea.grfAccessMode = AccessMode;
ea.grfInheritance= dwInheritance;
ea.Trustee.TrusteeForm = TrusteeForm;

// Test for Administrators group, a new trustee for the ACE
// For other trustees, you can try changing
// the array index to 1, 2 and 3 and rerun, see the effect
ea.Trustee.ptstrName = (LPTSTR)(pszTrustee[0]);

// Create a new ACL that merges the new ACE into the existing DACL.
dwRes = SetEntriesInAcl(1, &ea, pOldDACL, &pNewDACL);

//  Verify
if(dwRes != ERROR_SUCCESS)
{
	wprintf(L"SetEntriesInAcl() failed, error %u\n", dwRes);
	Cleanup(pSD, pNewDACL);
}
else
	wprintf(L"SetEntriesInAcl() working fine!\n");

// Attach the new ACL as the object's DACL.
dwRes = SetNamedSecurityInfo(pszObjName, ObjectType,
```

```
                DACL_SECURITY_INFORMATION,
                NULL,
                NULL,
                pNewDACL,
                NULL);

        if(dwRes != ERROR_SUCCESS)
        {
                wprintf(L"SetNamedSecurityInfo() failed, error %u\n", dwRes);
                Cleanup(pSD, pNewDACL);
        }
        wprintf(L"SetNamedSecurityInfo() is OK\n");
        return 0;
}
```

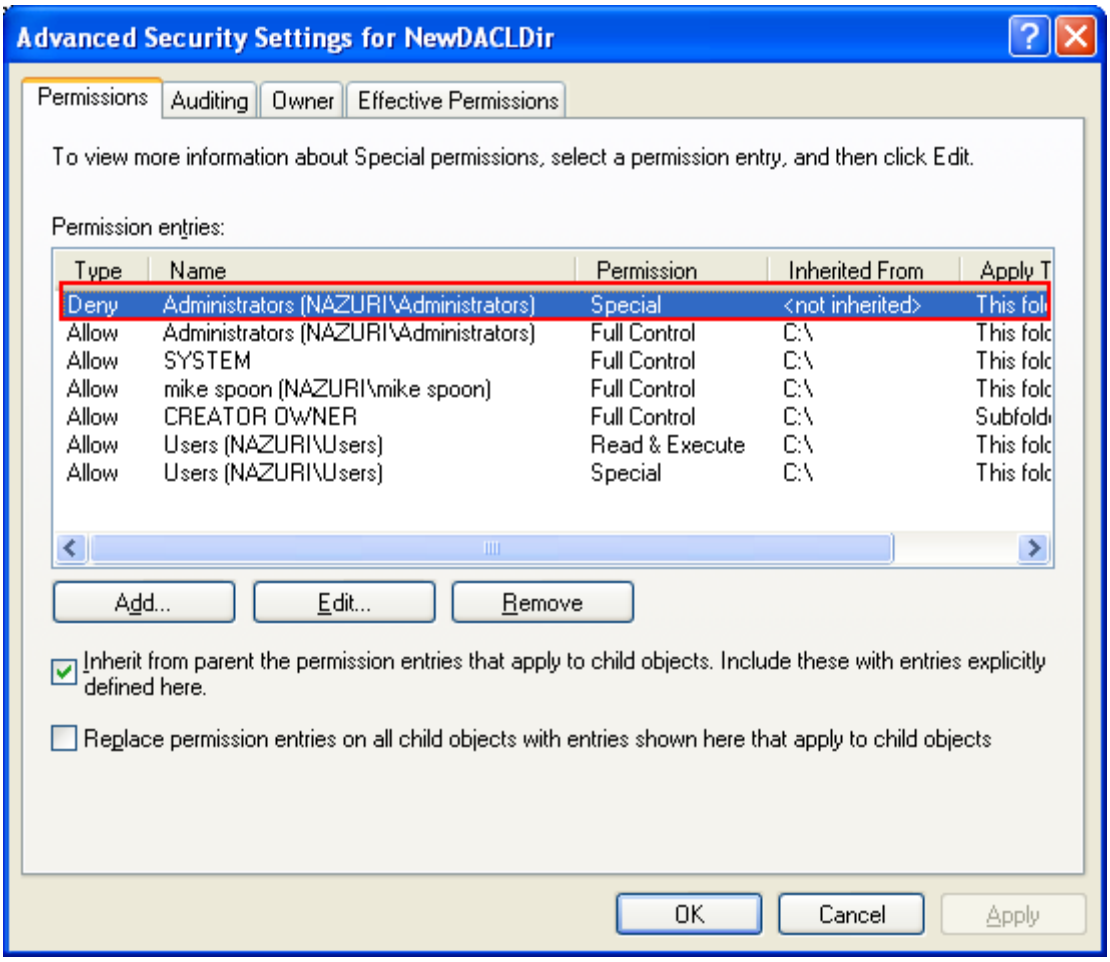
Build and run the project. The following screenshot is a sample output.



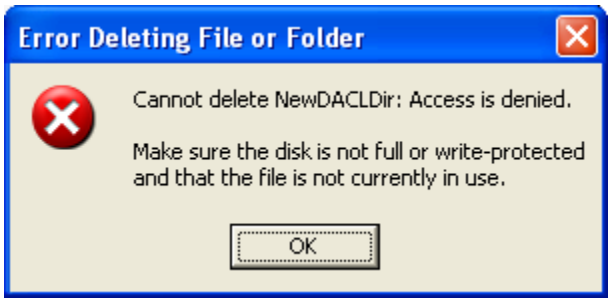
Then verify through the `C:\NewDACLDir` directory property pages again.



Under the Permission for Administrators group, the Special Permissions is ticked for the Deny permission. Keep in mind that Deny overrides the Allow permission. The following figure also confirmed that our new ACE for the Deny is not inherited.

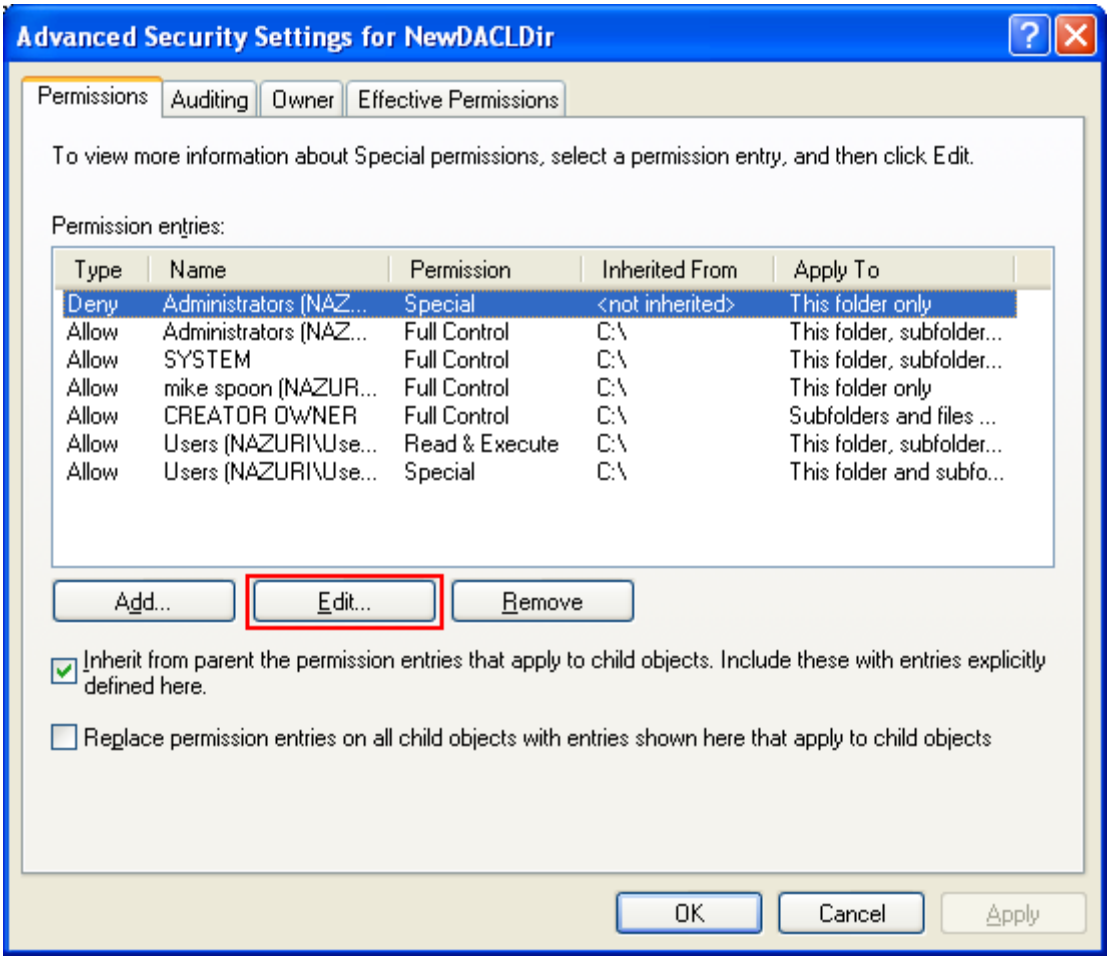


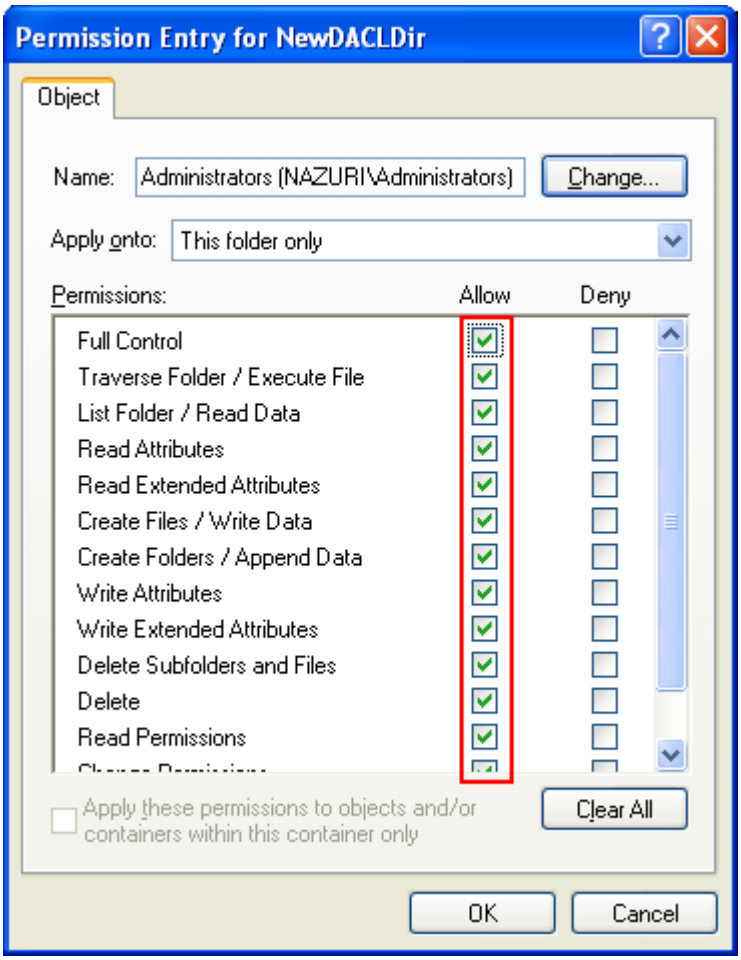
Remember that, DENY overwrites ALLOW permission. For this case because Mike spoon is a member of the Administrators group, however he cannot delete or open the `C:\\NewDACLDir` directory. We have to log off and login again as other Administrators group user **or** create another user that is a member of Administrators **or** take the object's ownership or just edit the permission entries (shown at the end of this section). When we try to delete the folder, the following message was displayed. Well, it funny isn't it? A user of Administrators group cannot delete/open his/her own folder.



It is same when we want to open the folder, it is not accessible.

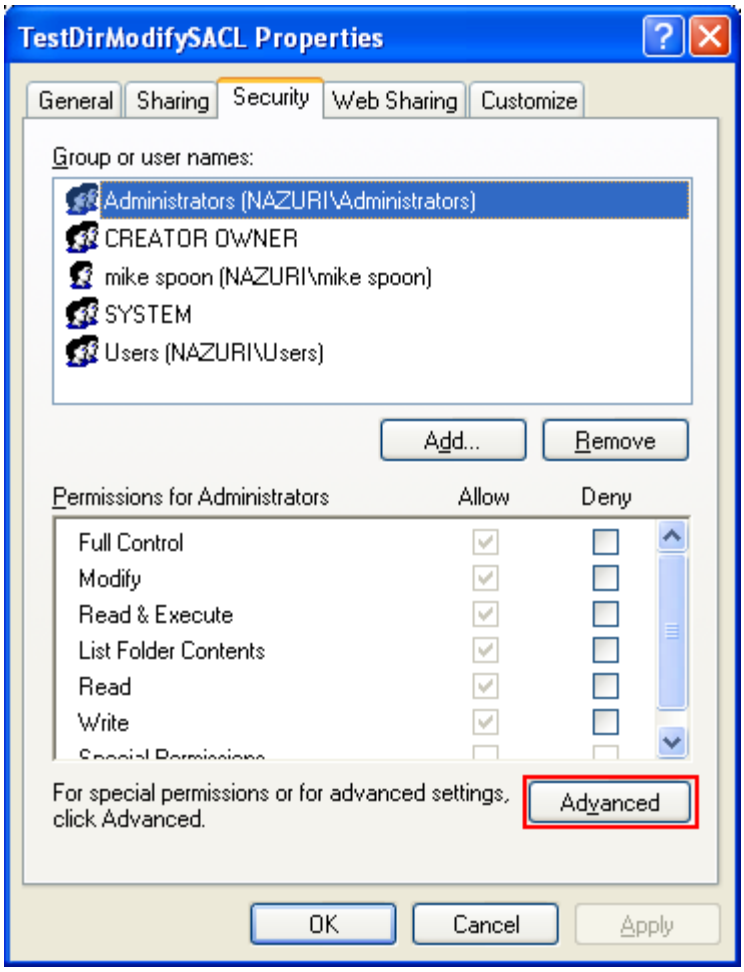
So, whatever it is, please don't mess up your machine. The following steps show how to re-enable the Full Control for Administrators group.

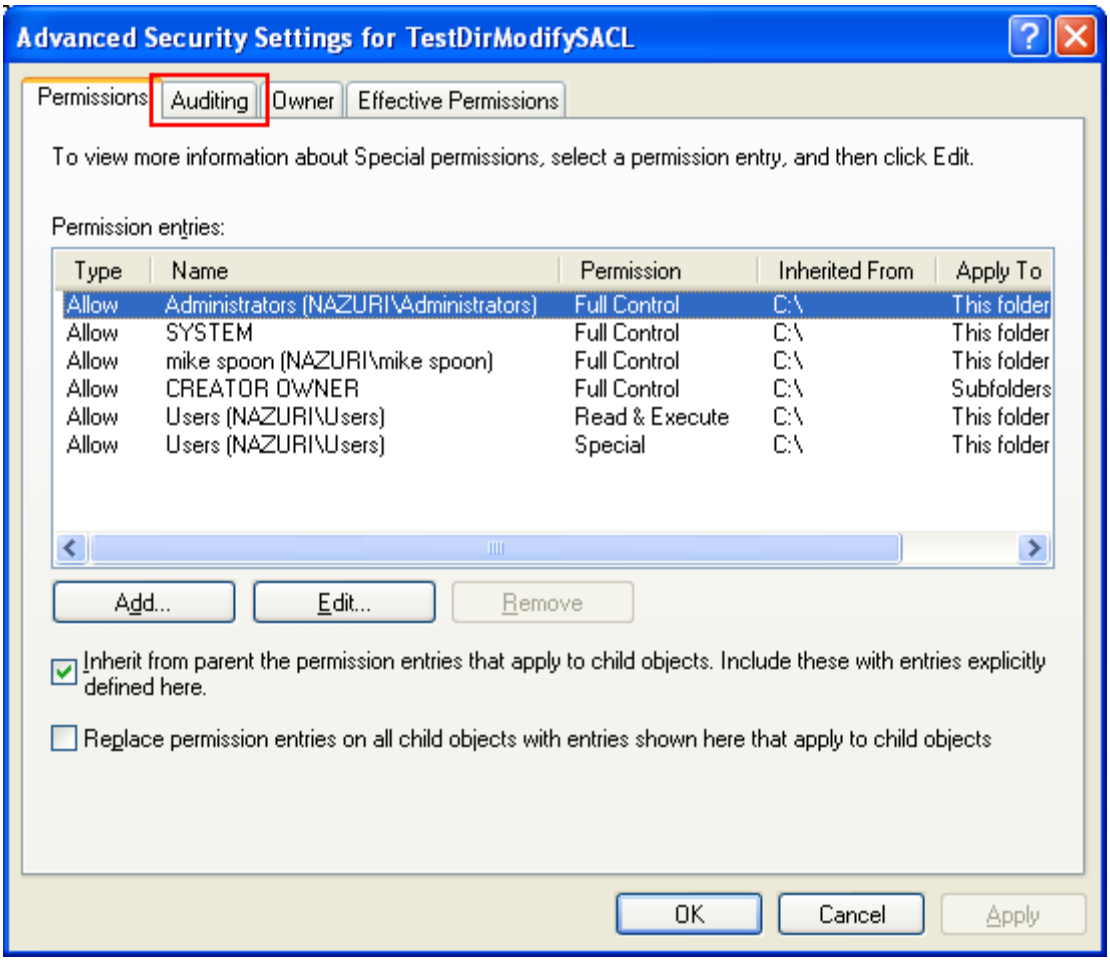


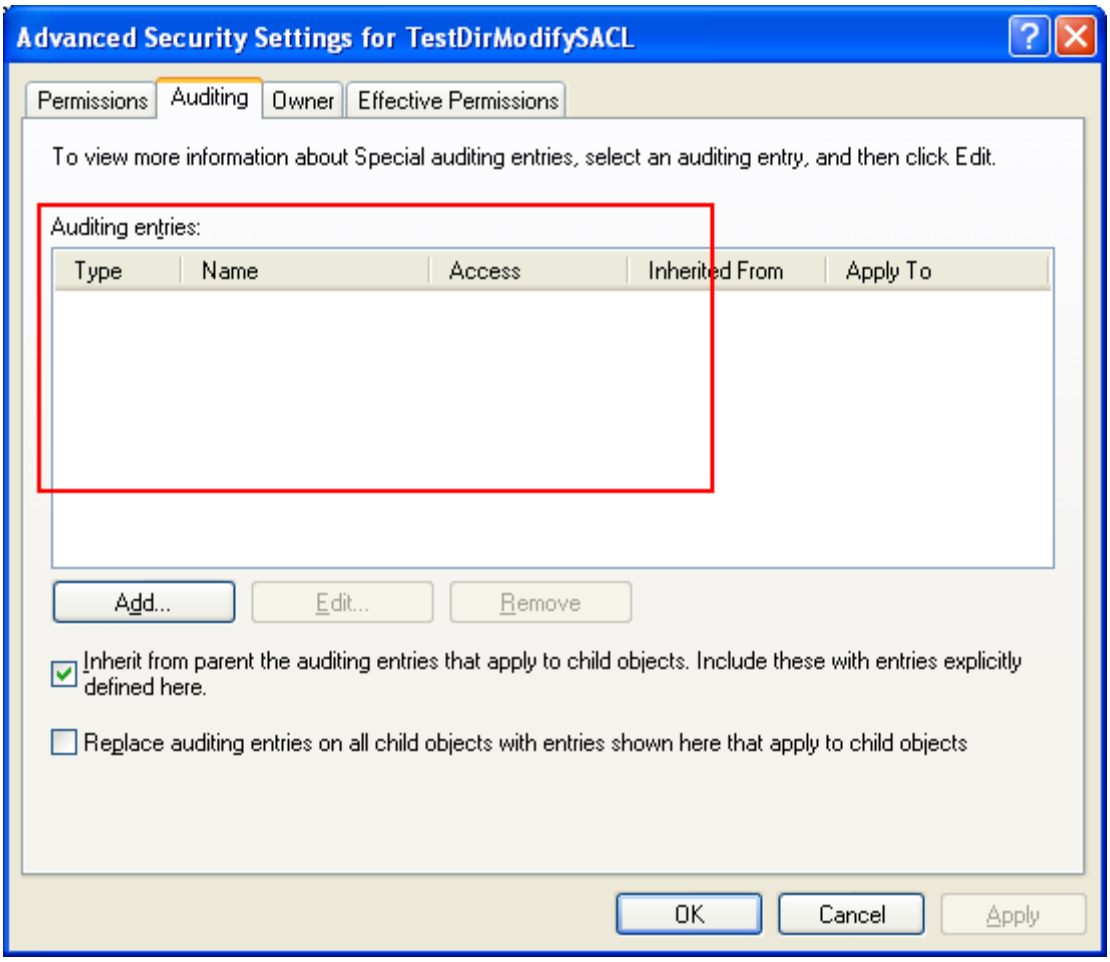


Modifying the SACL and Privilege Program Example

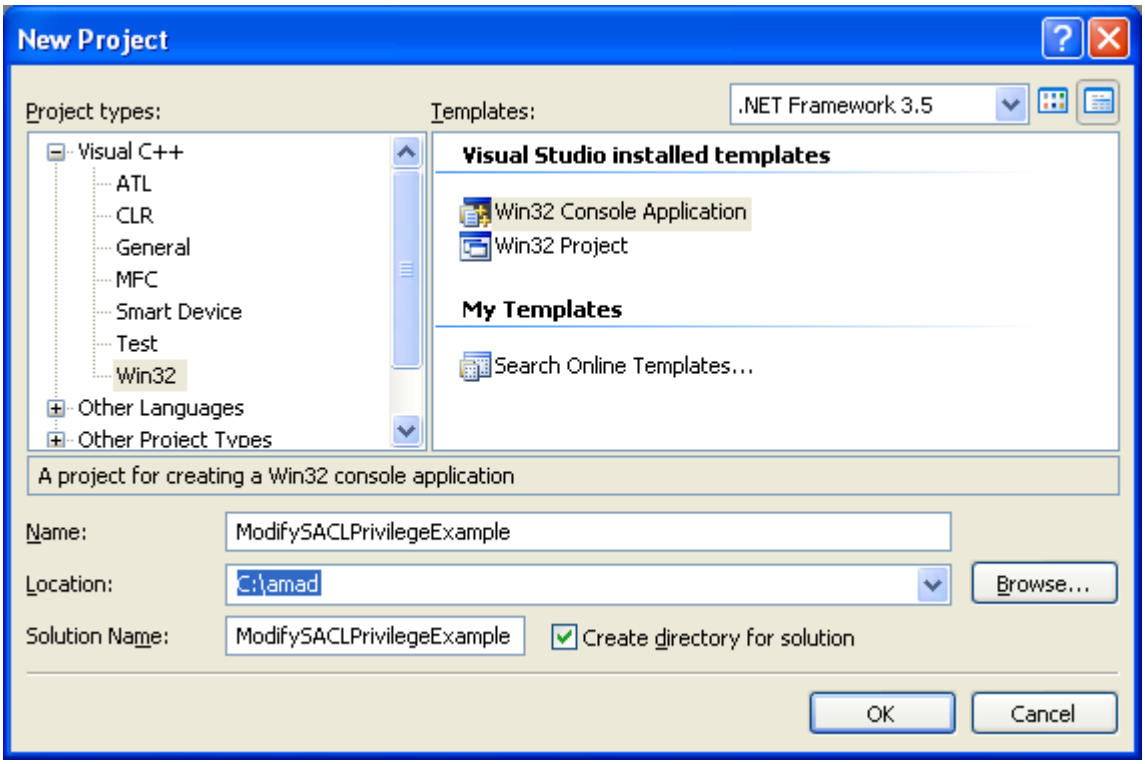
The following program example tries to modify the SACL. Before we run the program example, we create a directory named `C:\TestDirModifySACL`. The directory property pages before the program is run, shows there is no ACE in the Auditing entries. We will add an entry (SACL) to the ACE for auditing which needs a privilege.


TestDirModifySACL Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrators (NAZURI\Administrators)
CREATOR OWNER
mike spoon (NAZURI\mike spoon)
SYSTEM
Users (NAZURI\Users)
Add...
Remove
Permissions for Administrators
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply

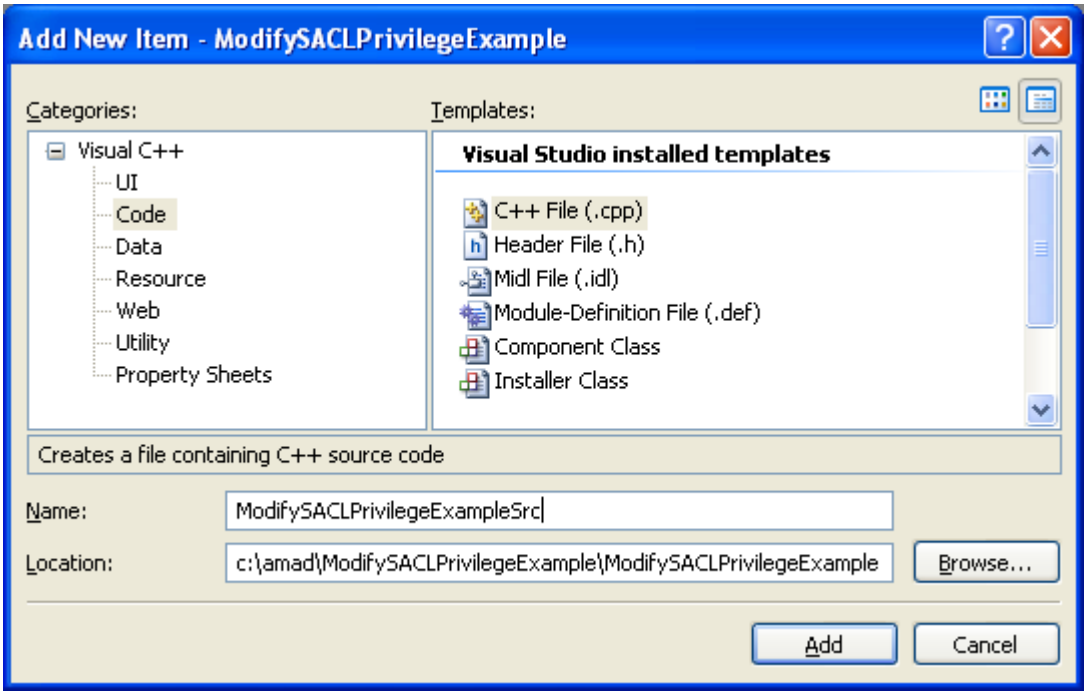

Advanced Security Settings for TestDirModifySACL
Permissions
Auditing
Owner
Effective Permissions
To view more information about Special permissions, select a permission entry, and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Allow
Administrators (NAZURI\Administrators)
Full Control
C:\
This folder
Allow
SYSTEM
Full Control
C:\
This folder
Allow
mike spoon (NAZURI\mike spoon)
Full Control
C:\
This folder
Allow
CREATOR OWNER
Full Control
C:\
Subfolders
Allow
Users (NAZURI\Users)
Read & Execute
C:\
This folder
Allow
Users (NAZURI\Users)
Special
C:\
This folder
Add...
Edit...
Remove
Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.
Replace permission entries on all child objects with entries shown here that apply to child objects
OK
Cancel
Apply



Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.



Next, add the following source code.

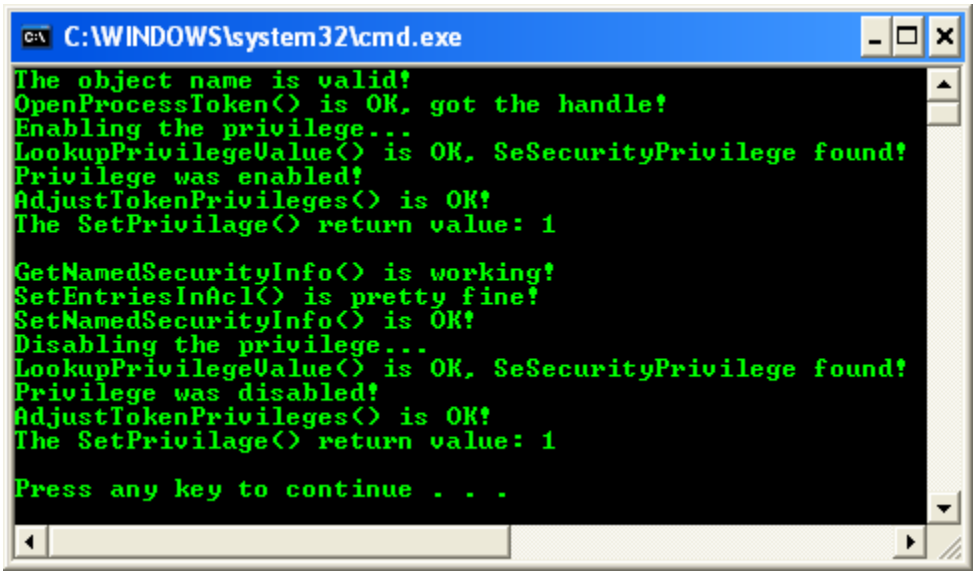
```
// Modifying ACL (SACL) of an object.
// Here we are going to add Allow standard right access and SACL.
// This Win XP machine is logged in by user named Mike spoon who
// is a member of Administrators group...
// To access a SACL using the GetNamedSecurityInfo() or
// SetNamedSecurityInfo() functions, we have to enable the SE_SECURITY_NAME
privilege
#include <windows.h>
#include <aclapi.h>
#include <stdio.h>

// ********** Enabling/disabling the privilege ***************
BOOL SetPrivilege(
    HANDLE hToken,          // access token handle
    LPCTSTR lpszPrivilege,  // name of privilege to enable/disable
    BOOL bEnablePrivilege   // to enable (or disable privilege)
    )
{
      TOKEN_PRIVILEGES tp;
      LUID luid;

      if(!LookupPrivilegeValue(
            NULL,            // lookup privilege on local system
            lpszPrivilege,   // privilege to lookup
            &luid))          // receives LUID of privilege
      {
            wprintf(L"LookupPrivilegeValue() failed, error: %u\n",
GetLastError());
            return FALSE;
      }
      else
            wprintf(L"LookupPrivilegeValue() is OK, %s found!\n",
lpszPrivilege);

      // the number of entries in the Privileges array
      tp.PrivilegeCount = 1;
      // an array of LUID_AND_ATTRIBUTES structures
      tp.Privileges[0].Luid = luid;

      // If TRUE
      if(bEnablePrivilege)
      {
            tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
            wprintf(L"Privilege was enabled!\n");
      }
      else
      {
            tp.Privileges[0].Attributes = 0;
            wprintf(L"Privilege was disabled!\n");
      }

      // Enable the privilege (or disable all privileges)
      if(!AdjustTokenPrivileges(
            hToken,
            FALSE,      // If TRUE, function disables all privileges,
                        // if FALSE the function modifies privileges based on
the tp
```

```
            &tp,
            sizeof(TOKEN_PRIVILEGES),
            (PTOKEN_PRIVILEGES) NULL,
            (PDWORD) NULL))
      {
            wprintf(L"AdjustTokenPrivileges() failed, error: %u\n",
GetLastError());
            return FALSE;
      }
      else
            wprintf(L"AdjustTokenPrivileges() is OK!\n");

      return TRUE;
}

// Clean up routine
void Cleanup(PSECURITY_DESCRIPTOR pSS, PACL pNewSACL)
{
      if(pSS != NULL)
            LocalFree((HLOCAL) pSS);
      else
            wprintf(L"pSS freed...\n");

      if(pNewSACL != NULL)
            LocalFree((HLOCAL) pNewSACL);
      else
            wprintf(L"pNewSACL freed...\n");
}

int main(int argc, WCHAR **argv)
{
      // Handle to the running process that is this program
      HANDLE hToken;
      BOOL bTestRetVal = FALSE;
      // Initially we try to enable
      BOOL bEnablePrivilege = TRUE;
      // The needed privilege
      LPCTSTR lpszPrivilege = L"SeSecurityPrivilege";
      // Name of object, here we will add an ACE for a directory
      LPTSTR pszObjName = L"C:\\TestDirModifySACL";
      // type of object, file or directory, a directory
      SE_OBJECT_TYPE ObjectType = SE_FILE_OBJECT;
      // Access mask for new ACE equal to 0X11000000 - GENERIC_ALL and
ACCESS_SYSTEM_SECURITY
      DWORD dwAccessRights = 0X11000000;
      // type of ACE, set audit for success
      ACCESS_MODE AccessMode = SET_AUDIT_SUCCESS;
      // Inheritance flags for new ACE. The OBJECT_INHERIT_ACE and
CONTAINER_INHERIT_ACE flags are
      // not propagated to an inherited ACE.
      DWORD dwInheritance = NO_PROPAGATE_INHERIT_ACE;
      // format of trustee structure, the trustee is name
      TRUSTEE_FORM TrusteeForm = TRUSTEE_IS_NAME;
      // The new trustee for the ACE is set to Johnny (logon name, full name
is John Doe),
      // a valid normal user f the local computer. However, this program run
by user Mike spoon
```

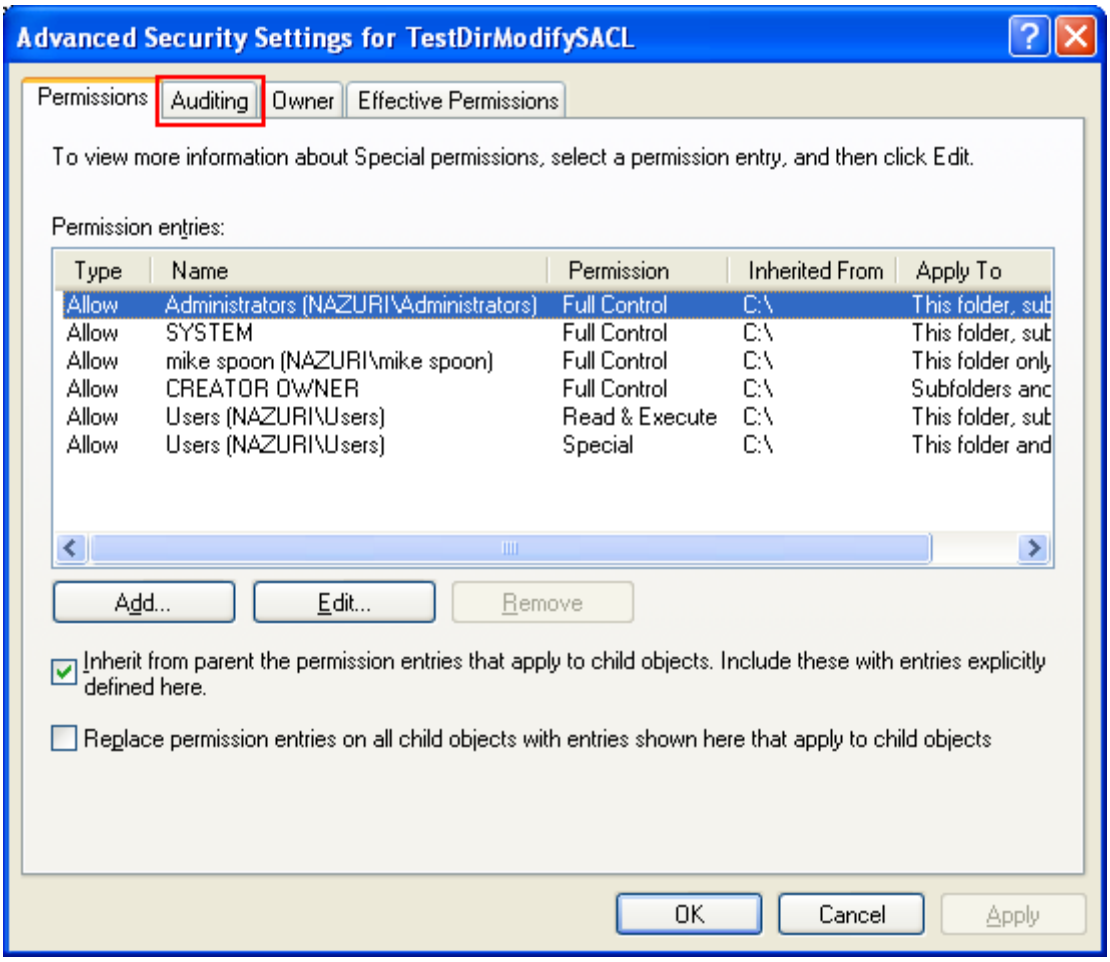
```
	LPTSTR pszTrustee = L"Johnny";
	// Result
	DWORD dwRes = 0;
	// Existing and new SACL pointers...
	PACL pOldSACL = NULL, pNewSACL = NULL;
	// Security descriptor
	PSECURITY_DESCRIPTOR pSS = NULL;
	// EXPLICIT_ACCESS structure
	EXPLICIT_ACCESS ea;

	// Verify the object name validity
	if(pszObjName == NULL)
	{
		wprintf(L"The object name is not valid!\n");
		return ERROR_INVALID_PARAMETER;
	}
	else
		wprintf(L"The object name is valid!\n");

	//**************Privilege routine here*****************
	//*************** Get the handle to the process ***************
	// Open a handle to the access token for the calling process.
	if(!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES,
&hToken))
	{
		wprintf(L"OpenProcessToken() failed, error %u\n",
GetLastError());
		return FALSE;
	}
	else
		wprintf(L"OpenProcessToken() is OK, got the handle!\n");

	//************ Enabling privilege *****************
	// Call the user defined SetPrivilege() function to enable privilege
	wprintf(L"Enabling the privilege...\n");
	bTestRetVal = SetPrivilege(hToken, lpszPrivilege, bEnablePrivilege);
	// Verify
	wprintf(L"The SetPrivilage() return value: %d\n\n", bTestRetVal);
	//*************** End enabling privilege ********************

	// Get a pointer to the existing SACL.
	dwRes = GetNamedSecurityInfo(pszObjName,
		ObjectType,
		SACL_SECURITY_INFORMATION,
		NULL,
		NULL,
		NULL,
		&pOldSACL,
		&pSS);

	// Verify
	if(dwRes != ERROR_SUCCESS)
	{
		wprintf(L"GetNamedSecurityInfo() failed, error %u\n", dwRes);
		Cleanup(pSS, pNewSACL);
	}
	else
```

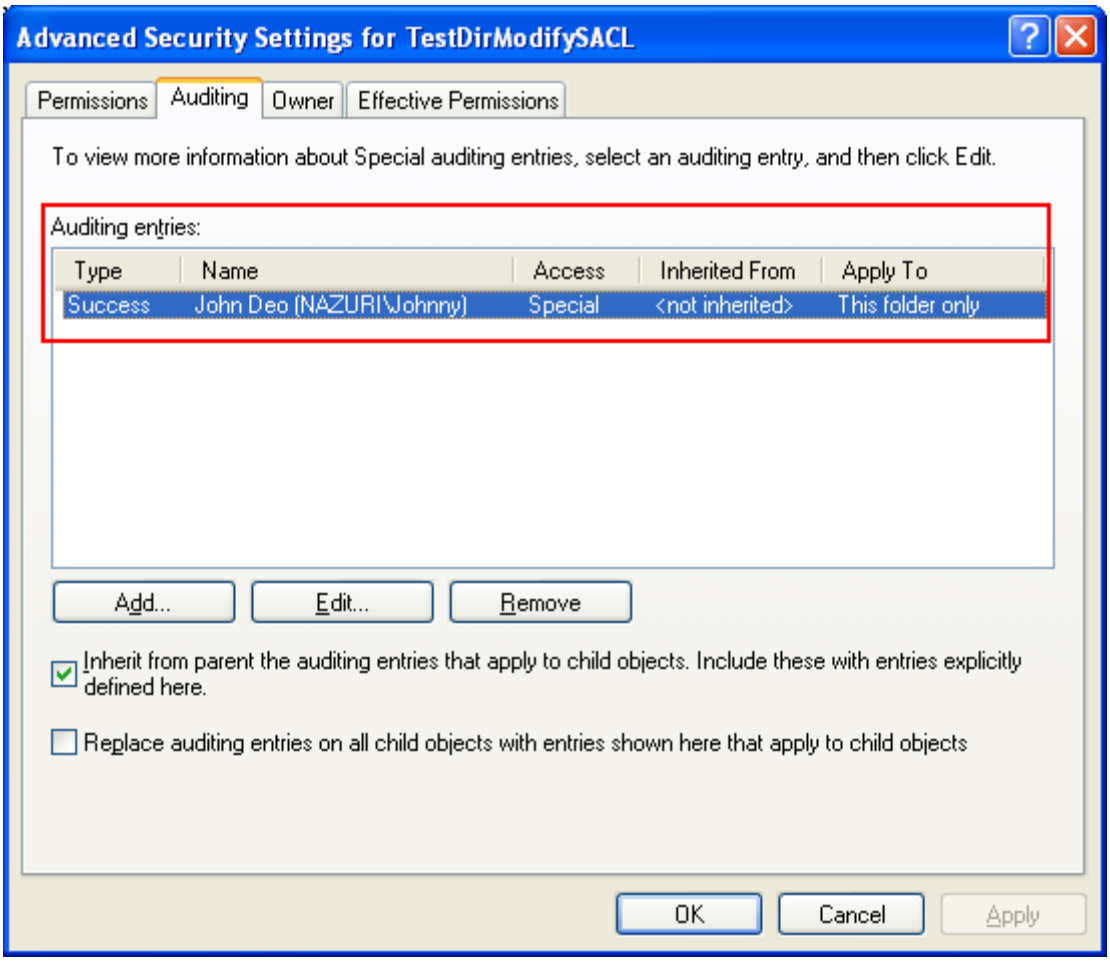
```
		wprintf(L"GetNamedSecurityInfo() is working!\n");

	// Initialize an EXPLICIT_ACCESS structure for the new ACE.
	// If more entries needed, you can create an array
	// of the ea variable of the EXPLICIT_ACCESS
	ZeroMemory(&ea, sizeof(EXPLICIT_ACCESS));
	ea.grfAccessPermissions = dwAccessRights;
	ea.grfAccessMode = AccessMode;
	ea.grfInheritance= dwInheritance;
	ea.Trustee.TrusteeForm = TrusteeForm;

	// Other structure elements...
	// ea.Trustee.TrusteeType = TRUSTEE_IS_GROUP;
	// ea.Trustee.TrusteeType = TRUSTEE_IS_USER;

	// The trustee is testuser
	ea.Trustee.ptstrName = pszTrustee;
	// Create a new ACL that merges the new ACE into the existing ACL.
	dwRes = SetEntriesInAcl(1, &ea, pOldSACL, &pNewSACL);
	if(dwRes != ERROR_SUCCESS)
	{
		wprintf(L"SetEntriesInAcl() failed, error %u\n", dwRes);
		Cleanup(pSS, pNewSACL);
	}
	else
		wprintf(L"SetEntriesInAcl() is pretty fine!\n");

	// Attach the new ACL as the object's SACL.
	dwRes = SetNamedSecurityInfo(pszObjName,
		ObjectType,
		SACL_SECURITY_INFORMATION,
		NULL,
		NULL,
		NULL,
		pNewSACL);

	if(dwRes != ERROR_SUCCESS)
	{
		wprintf(L"SetNamedSecurityInfo() failed, error %u\n", dwRes);
		Cleanup(pSS, pNewSACL);
	}
	else
		wprintf(L"SetNamedSecurityInfo() is OK!\n");

	//************* Disable the privilege ***************
	wprintf(L"Disabling the privilege...\n");
	bEnablePrivilege = FALSE;
	SetPrivilege(hToken, lpszPrivilege, bEnablePrivilege);
	// Verify
	wprintf(L"The SetPrivilage() return value: %d\n\n", bTestRetVal);
	//************* End disabling the privilege ***************

	return 0;
}
```

Build and run the project. The following screenshot is a sample output.



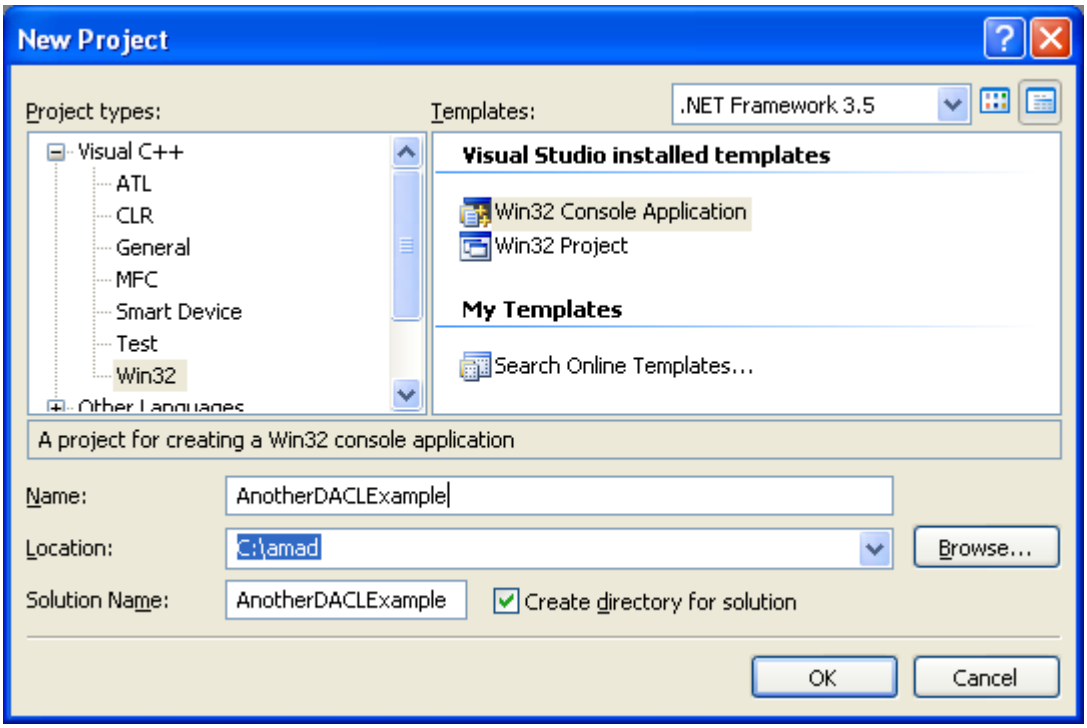
From the program output and the Advanced Security Setting for `C:\TestDirModifySACL` directory property pages as shown below, we can see that an entry was added for auditing.


Advanced Security Settings for TestDirModifySACL
Permissions
Auditing
Owner
Effective Permissions
To view more information about Special permissions, select a permission entry, and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Apply To
Allow
Administrators (NAZURI\Administrators)
Full Control
C:\
This folder, sub
Allow
SYSTEM
Full Control
C:\
This folder, sub
Allow
mike spoon (NAZURI\mike spoon)
Full Control
C:\
This folder only
Allow
CREATOR OWNER
Full Control
C:\
Subfolders anc
Allow
Users (NAZURI\Users)
Read & Execute
C:\
This folder, sub
Allow
Users (NAZURI\Users)
Special
C:\
This folder and
Add...
Edit...
Remove
Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.
Replace permission entries on all child objects with entries shown here that apply to child objects
OK
Cancel
Apply

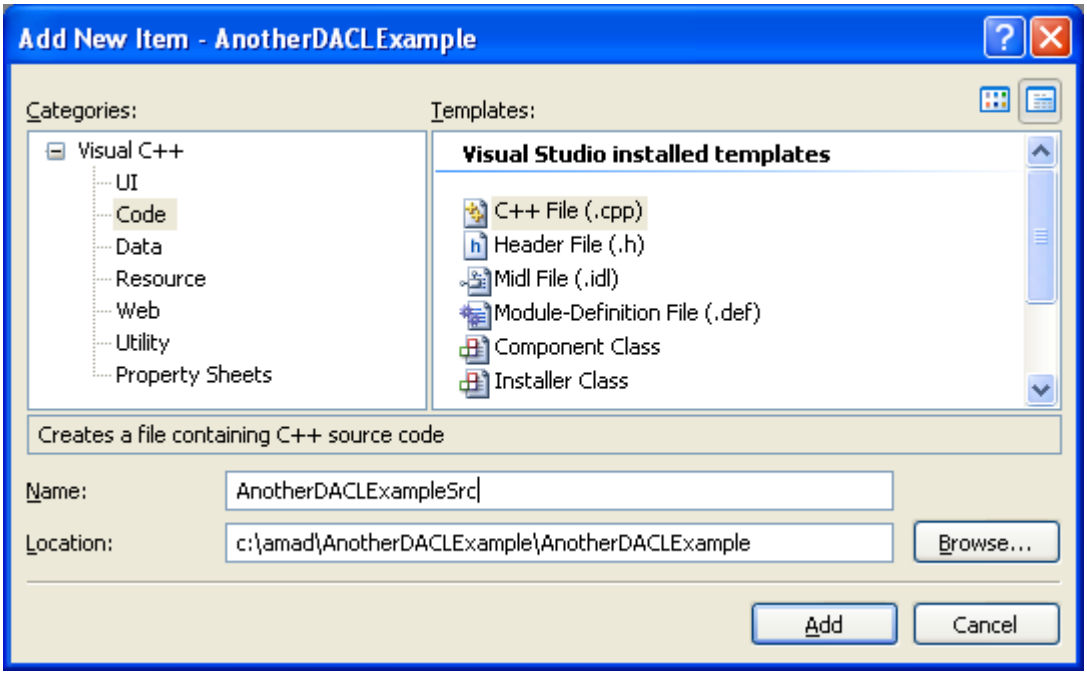


Another New DACL Which Does Not Inherit Program Example

In this program example, we directly create a new DACL for a new object which does not inherit the parent or default DACL. In this example, we will create a directory that grants "Full Control" to the "Everyone" group modifying the default DACL (inherited).
Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.



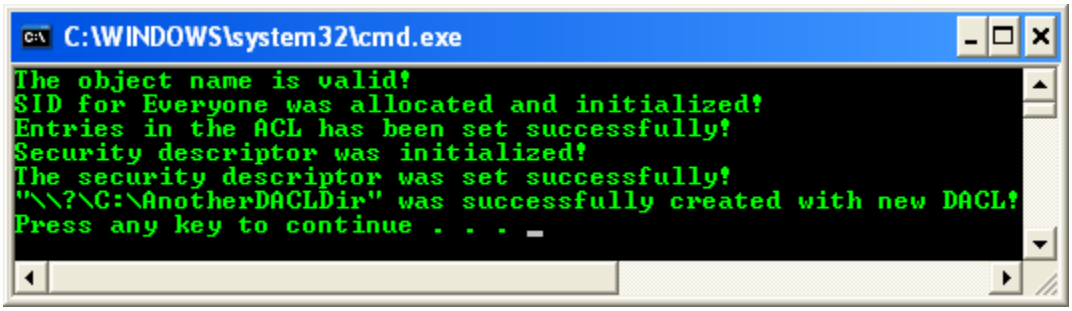
Next, add the following source code.

```
#include <windows.h>
#include <accCtrl.h>
```

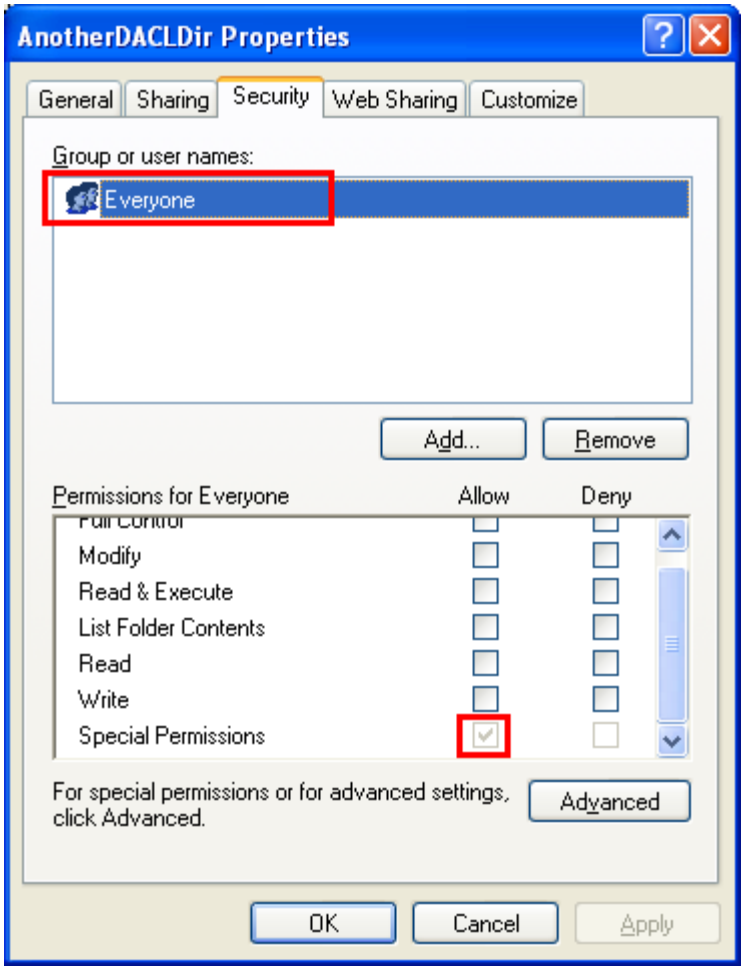
```
#include <aclapi.h>
#include <stdio.h>

int wmain(int argc, WCHAR **argv)
{
	// Directory object to be assigned new DACL
	LPTSTR pszPath = L"\\\\?\\C:\\AnotherDACLDir";
	EXPLICIT_ACCESS ea = {0};
	PACL acl = NULL;
	SECURITY_ATTRIBUTES sa = {0};
	SID_IDENTIFIER_AUTHORITY SIDAuthWorld = SECURITY_WORLD_SID_AUTHORITY;
	// Everyone SID
	PSID everyone_sid = NULL;
	DWORD dwRetVal = 0;
	PSECURITY_DESCRIPTOR sd = {0};

	// Verify the object validity
	if(pszPath == NULL)
	{
		wprintf(L"The object name is not valid!\n");
		return ERROR_INVALID_PARAMETER;
	}
	else
		wprintf(L"The object name is valid!\n");

	// allocates and initializes a security identifier (SID),
	// and can be up to eight subauthorities
	// This function creates a SID with a 32-bit RID value.
	// For applications that require longer RID values, use
CreateWellKnownSid()
	// - http://msdn.microsoft.com/en-us/library/aa379650%28VS.85%29.aspx
	if(AllocateAndInitializeSid(&SIDAuthWorld,
		1,
		SECURITY_WORLD_RID,
		0, 0, 0, 0, 0, 0, 0,
		&everyone_sid) != 0)

		wprintf(L"SID for Everyone was allocated and initialized!\n");
	else
	{
		wprintf(L"AllocateAndInitializeSid() failed, error %u\n",
GetLastError());
		exit(1);
	}

	// Initializes SECURITY_ATTRIBUTES
	SecureZeroMemory(&ea, sizeof(EXPLICIT_ACCESS));

	ea.grfAccessPermissions = SPECIFIC_RIGHTS_ALL | STANDARD_RIGHTS_ALL;
	ea.grfAccessMode = SET_ACCESS;
	ea.grfInheritance = NO_INHERITANCE;
	ea.Trustee.TrusteeForm = TRUSTEE_IS_SID;
	ea.Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
	ea.Trustee.ptstrName  = (LPWSTR)everyone_sid;

	dwRetVal = SetEntriesInAcl(1, &ea, NULL, &acl);
```

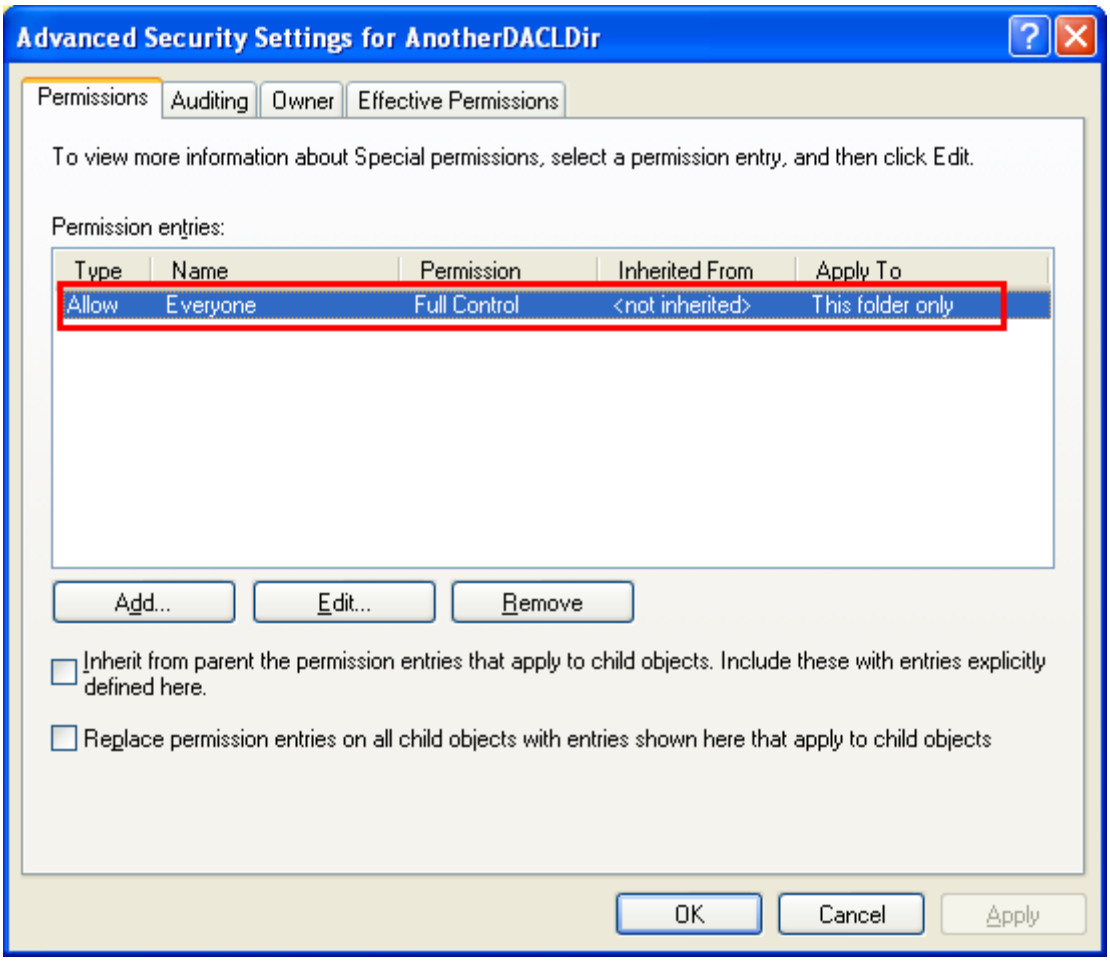
```
	if(dwRetVal == ERROR_SUCCESS)
		wprintf(L"Entries in the ACL has been set successfully!\n");
	else
	{
		wprintf(L"Faile to set the ACL entries, error %u\n", dwRetVal);
		exit(1);
	}

	sd =
(PSECURITY_DESCRIPTOR)LocalAlloc(LPTR,SECURITY_DESCRIPTOR_MIN_LENGTH);

	// initializes a new security descriptor
	if(InitializeSecurityDescriptor(sd, SECURITY_DESCRIPTOR_REVISION) != 0)
		wprintf(L"Security descriptor was initialized!\n");
	else
	{
		wprintf(L"Failed to Security descriptor, error %u\n",
GetLastError());
		exit(1);
	}

	// sets information in a discretionary access control list (DACL)
	// If a DACL is already present in the security descriptor, the DACL is
replaced.
	if(SetSecurityDescriptorDacl(sd, TRUE, acl, FALSE) != 0)
		wprintf(L"The security descriptor was set successfully!\n");
	else
	{
		wprintf(L"Failed to set security descriptor, error %u\n",
GetLastError());
		exit(1);
	}

	sa.nLength = sizeof(SECURITY_ATTRIBUTES);
	sa.lpSecurityDescriptor = sd;
	sa.bInheritHandle = FALSE;

	// Create directory object with new DACL
	if(CreateDirectory(pszPath, &sa) != 0)
		wprintf(L"\"%s\" was successfully created with new DACL!\n",
pszPath);
	else
	{
		wprintf(L"Failed to create \"%s\" with new DACL, error %u\n",
pszPath, GetLastError());
		exit(1);
	}

	// Relesing the allocated resources
	FreeSid(everyone_sid);
	LocalFree(sd);
	LocalFree(acl);

	return 0;
}
```

Build and run the project. The following screenshot is a sample output.



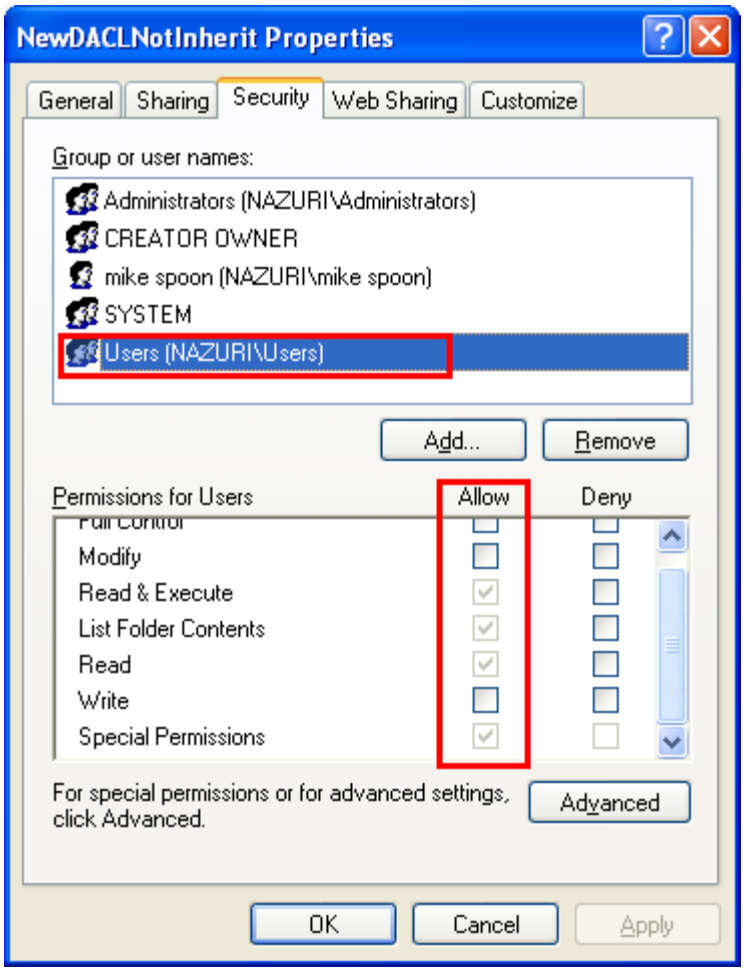
Then, verify through the folder property's page.

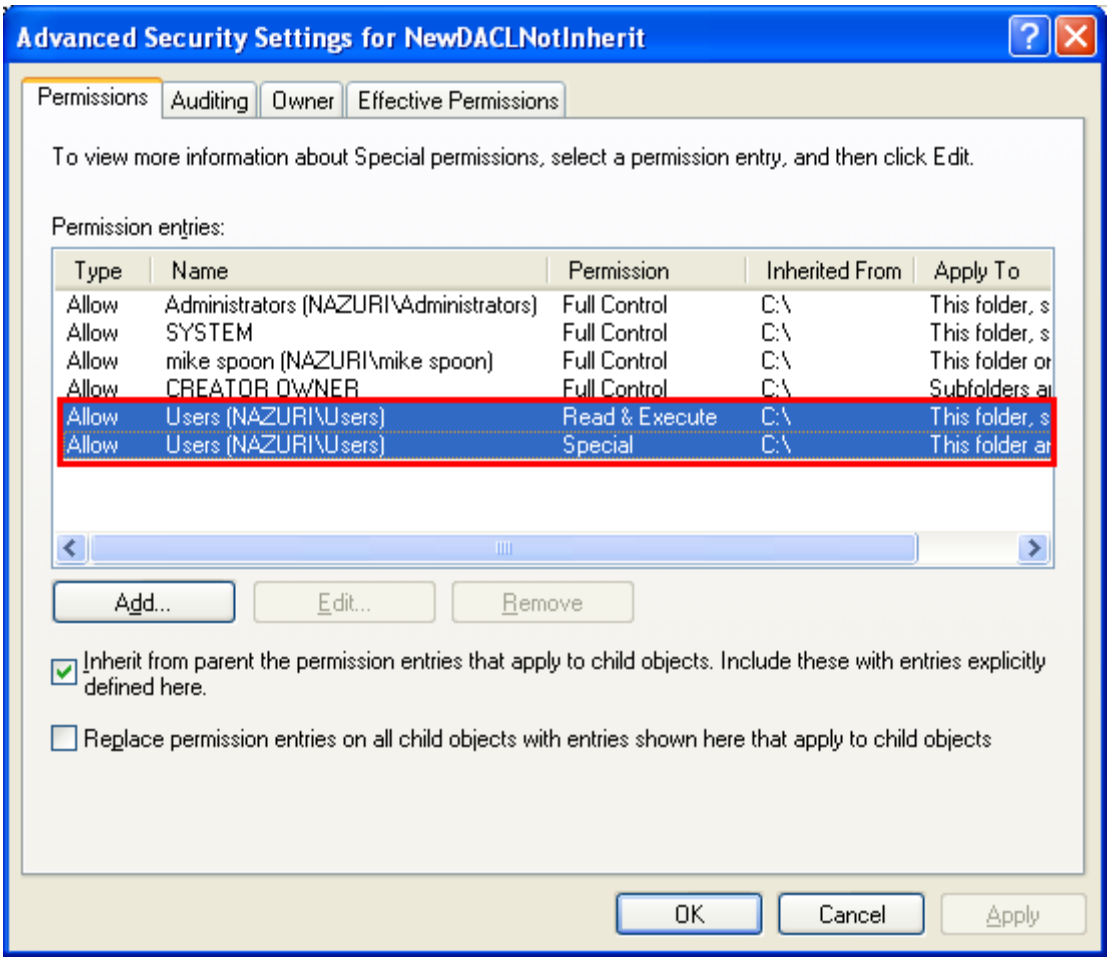




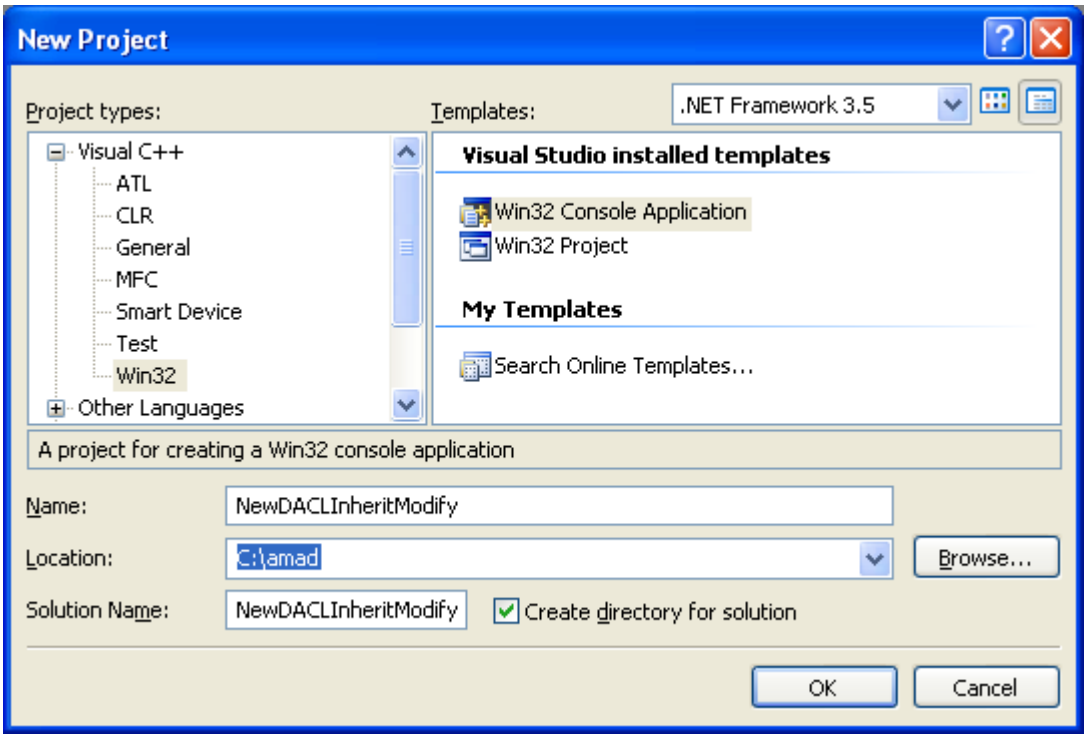
More New DACL Which Inherit Program Example

In this example, we create a Windows object (text file) which inherits the default DACL and then we modify the DACL to a new one. Firstly, a folder is created which inherit the default rights. Then we adds a Full Control explicit access entry for the built in "Users" group for text file object created under the folder. The following folder properties page shows the permission entries before running the program example.

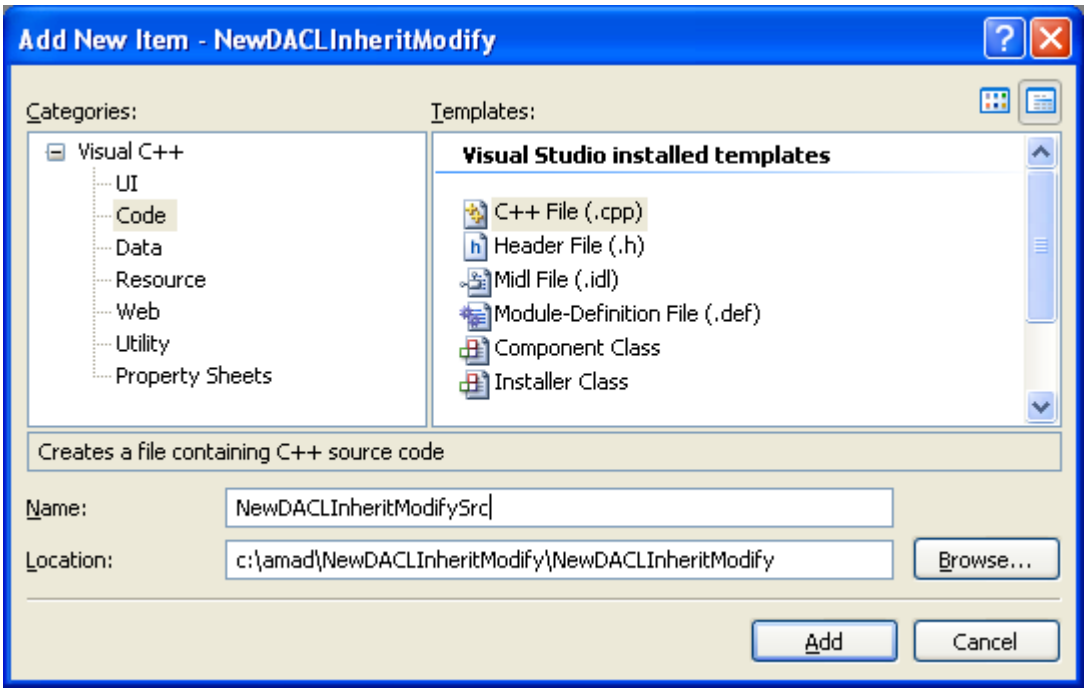

NewDACLNotInherit Properties
General
Sharing
Security
Web Sharing
Customize
Group or user names:
Administrators (NAZURI\Administrators)
CREATOR OWNER
mike spoon (NAZURI\mike spoon)
SYSTEM
Users (NAZURI\Users)
Add...
Remove
Permissions for Users
Allow
Deny
Modify
Read & Execute
List Folder Contents
Read
Write
Special Permissions
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply



Create a new empty Win32 console application project. Give a suitable project name and change the project location if needed.



Then, add the source file and give it a suitable name.



Next, add the following source code.

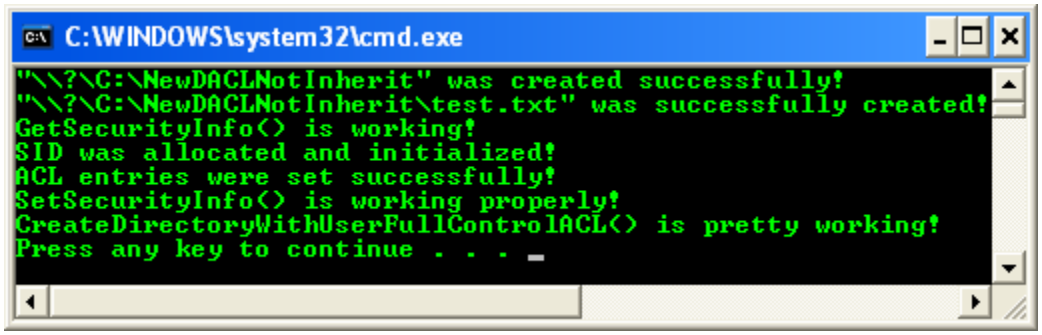
```
#include <windows.h>
```

```
#include <aclapi.h>
#include <stdio.h>

BOOL CreateDirectoryWithUserFullControlACL(LPCTSTR lpPath, LPCTSTR lpFile)
{
	HANDLE hDir;
	ACL* pOldDACL;
	PSID pSid = NULL;
	SECURITY_DESCRIPTOR* pSD = NULL;
	DWORD dwErr, dwRetVal;
	EXPLICIT_ACCESS ea={0};

	// The directory should inherit the parent DACL (C:)
	if(!CreateDirectory(lpPath,NULL))
	{
		wprintf(L"Failed to create \"%s\", error %u\n", lpPath,
GetLastError());
		return FALSE;
	}
	else
		wprintf(L"\"%s\" was created successfully!\n", lpPath);

	// This file DACL will be modified later
	hDir =
CreateFile(lpFile,READ_CONTROL|WRITE_DAC,0,NULL,CREATE_ALWAYS,FILE_FLAG_BACKU
P_SEMANTICS,NULL);

	if(hDir == INVALID_HANDLE_VALUE)
	{
		wprintf(L"CreateFile() failed, error %u\n", GetLastError());
		return FALSE;
	}
	else
		wprintf(L"\"%s\" was successfully created!\n", lpFile);

	dwRetVal = GetSecurityInfo(hDir, SE_FILE_OBJECT,
DACL_SECURITY_INFORMATION,NULL, NULL, &pOldDACL, NULL, (void**)&pSD);

	if(dwRetVal == ERROR_SUCCESS)
		wprintf(L"GetSecurityInfo() is working!\n");
	else
	{
		wprintf(L"GetSecurityInfo() failed, error %u\n", GetLastError());
		return FALSE;
	}

	SID_IDENTIFIER_AUTHORITY authNt = SECURITY_NT_AUTHORITY;

	if(AllocateAndInitializeSid(&authNt,
		2,
		SECURITY_BUILTIN_DOMAIN_RID,
		DOMAIN_ALIAS_RID_USERS,
		0,0,0,0,0,0,&pSid) != 0)
		wprintf(L"SID was allocated and initialized!\n");
	else
	{
```

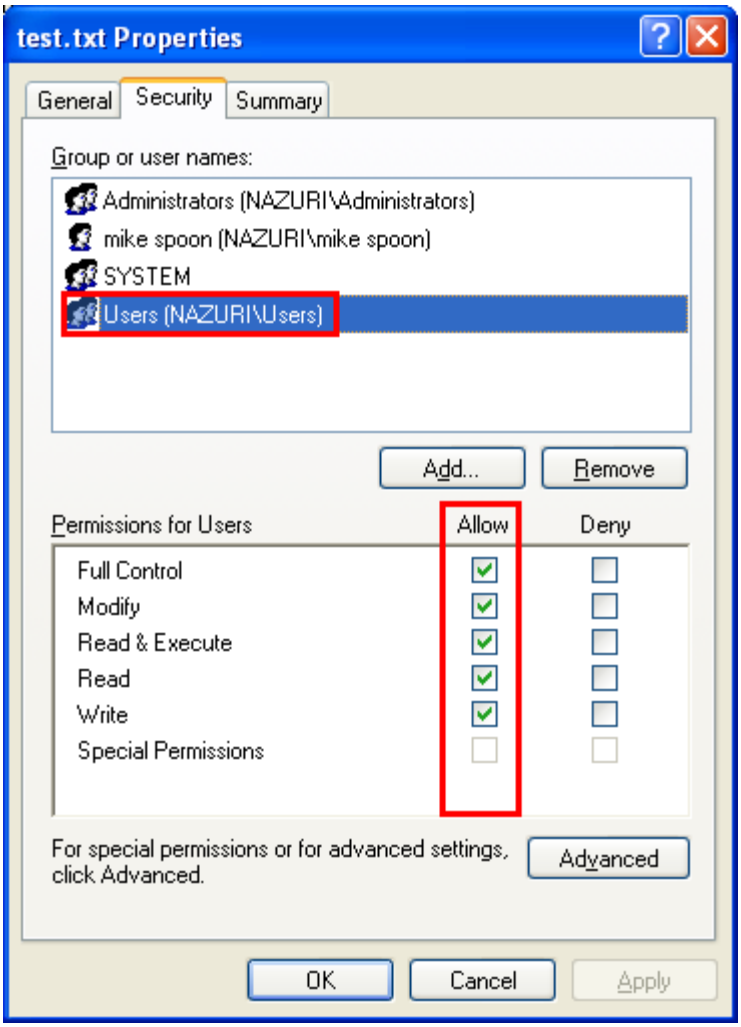
```
            wprintf(L"Failed to allocate and initialize SID, error %u\n",
GetLastError());
            return FALSE;
      }

      // Initialize EXPLICIT_ACCESS structure
      ea.grfAccessMode = GRANT_ACCESS;
      ea.grfAccessPermissions = GENERIC_ALL;
      ea.grfInheritance = CONTAINER_INHERIT_ACE|OBJECT_INHERIT_ACE;
      ea.Trustee.TrusteeType = TRUSTEE_IS_GROUP;
      ea.Trustee.TrusteeForm = TRUSTEE_IS_SID;
      ea.Trustee.ptstrName = (LPTSTR)pSid;

      ACL* pNewDACL = 0;

      // Setting new DACL (ACE) of the file
      dwErr = SetEntriesInAcl(1,&ea,pOldDACL,&pNewDACL);

      if(dwErr == ERROR_SUCCESS)
            wprintf(L"ACL entries were set successfully!\n");
      else
      {
            wprintf(L"Failed to set the ACL entries, error %u\n", dwErr);
            return FALSE;
      }

      if(pNewDACL)
      {
            if((dwErr = SetSecurityInfo(hDir,
                  SE_FILE_OBJECT,
                  DACL_SECURITY_INFORMATION,
                  NULL,
                  NULL,
                  pNewDACL,
                  NULL)) == ERROR_SUCCESS)

                  wprintf(L"SetSecurityInfo() is working properly!\n");
            else
            {
                  wprintf(L"SetSecurityInfo() failed miserably, error %u\n",
dwErr);
                  return FALSE;
            }
      }

      // Free-up all the allocated resources
      FreeSid(pSid);
      LocalFree(pNewDACL);
      LocalFree(pSD);
      LocalFree(pOldDACL);
      CloseHandle(hDir);

      return TRUE;
}

int wmain(int argc, WCHAR **argv)
{
```

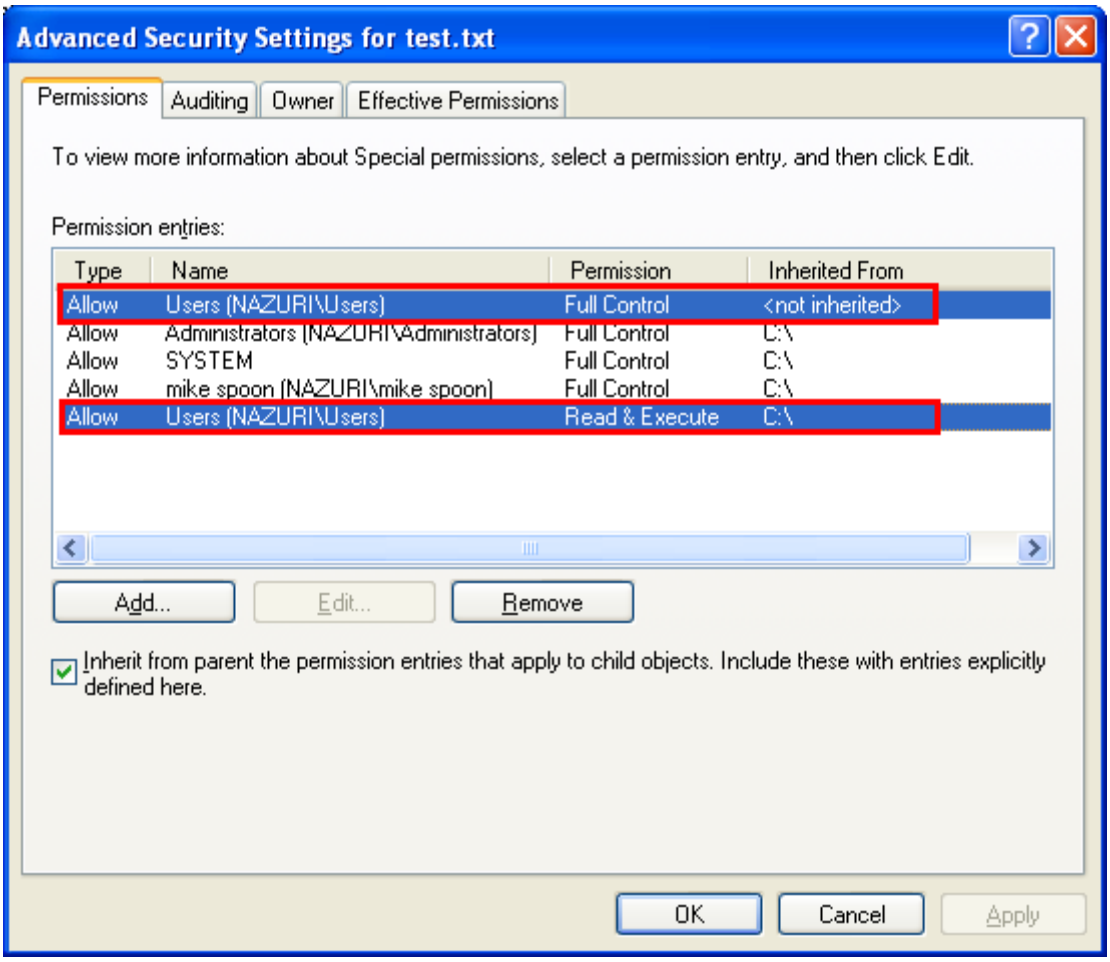
```
	BOOL bRetVal = FALSE;
	// The directory will inherit parent DACL/ACE
	LPCTSTR lpPath = L"\\\\?\\C:\\NewDACLNotInherit";
	// We will modify the file DACL/ACE for Users
	LPCTSTR lpFile = L"\\\\?\\C:\\NewDACLNotInherit\\test.txt";


	if(CreateDirectoryWithUserFullControlACL(lpPath, lpFile) == FALSE)
	{
		wprintf(L"CreateDirectoryWithUserFullControlACL() failed, error
%u\n", GetLastError());
		exit(1);
	}
	else
		wprintf(L"CreateDirectoryWithUserFullControlACL() is pretty
working!\n");

	return 0;
}
```

Build and run the project. The following screenshot is a sample output.



Then verify through the folder property's page.


test.txt Properties
General
Security
Summary
Group or user names:
Administrators (NAZURI\Administrators)
mike spoon (NAZURI\mike spoon)
SYSTEM
Users (NAZURI\Users)
Add...
Remove
Permissions for Users
Allow
Deny
Full Control
Modify
Read & Execute
Read
Write
Special Permissions
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply


Advanced Security Settings for test.txt
Permissions
Auditing
Owner
Effective Permissions
To view more information about Special permissions, select a permission entry, and then click Edit.
Permission entries:
Type
Name
Permission
Inherited From
Allow
Users (NAZURI\Users)
Full Control
<not inherited>
Allow
Administrators (NAZURI\Administrators)
Full Control
C:\
Allow
SYSTEM
Full Control
C:\
Allow
mike spoon (NAZURI\mike spoon)
Full Control
C:\
Allow
Users (NAZURI\Users)
Read & Execute
C:\
Add...
Edit...
Remove
Inherit from parent the permission entries that apply to child objects. Include these with entries explicitly defined here.
OK
Cancel
Apply