Creating a Security Descriptor for a New Object in C++

The following example creates a ***security descriptor*** for a new registry key using the following process. Similar code can be used to create a security descriptor for other object types.

- The example fills an array of **EXPLICIT_ACCESS** structures with the information for two ACEs. One ACE allows read access to everyone; the other ACE allows full access to administrators.
- The **EXPLICIT_ACCESS** array is passed to the **SetEntriesInAcl** function to create a DACL for the security descriptor.
- After allocating memory for the security descriptor, the example calls the **InitializeSecurityDescriptor** and **SetSecurityDescriptorDacl** functions to initialize the security descriptor and attach the DACL.
- The security descriptor is then stored in a SECURITY_ATTRIBUTES structure and passed to the **RegCreateKeyEx** function, which attaches the security descriptor to the newly created key.

C++

```
#pragma comment(lib, "advapi32.lib")

#include <windows.h>
#include <stdio.h>
#include <aclapi.h>
#include <tchar.h>

void main()
{

    DWORD dwRes, dwDisposition;
    PSID pEveryoneSID = NULL, pAdminSID = NULL;
    PACL pACL = NULL;
    PSECURITY_DESCRIPTOR pSD = NULL;
    EXPLICIT_ACCESS ea[2];
    SID_IDENTIFIER_AUTHORITY SIDAuthWorld =
            SECURITY_WORLD_SID_AUTHORITY;
    SID_IDENTIFIER_AUTHORITY SIDAuthNT = SECURITY_NT_AUTHORITY;
    SECURITY_ATTRIBUTES sa;
    LONG lRes;
    HKEY hkSub = NULL;

    // Create a well-known SID for the Everyone group.
    if(!AllocateAndInitializeSid(&SIDAuthWorld, 1,
                     SECURITY_WORLD_RID,
```

```
                    0, 0, 0, 0, 0, 0, 0,
                    &pEveryoneSID))
    {
        _tprintf(_T("AllocateAndInitializeSid Error %u\n"), GetLastError());
        goto Cleanup;
    }

    // Initialize an EXPLICIT_ACCESS structure for an ACE.
    // The ACE will allow Everyone read access to the key.
    ZeroMemory(&ea, 2 * sizeof(EXPLICIT_ACCESS));
    ea[0].grfAccessPermissions = KEY_READ;
    ea[0].grfAccessMode = SET_ACCESS;
    ea[0].grfInheritance= NO_INHERITANCE;
    ea[0].Trustee.TrusteeForm = TRUSTEE_IS_SID;
    ea[0].Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
    ea[0].Trustee.ptstrName  = (LPTSTR) pEveryoneSID;

    // Create a SID for the BUILTIN\Administrators group.
    if(! AllocateAndInitializeSid(&SIDAuthNT, 2,
                    SECURITY_BUILTIN_DOMAIN_RID,
                    DOMAIN_ALIAS_RID_ADMINS,
                    0, 0, 0, 0, 0, 0,
                    &pAdminSID))
    {
        _tprintf(_T("AllocateAndInitializeSid Error %u\n"), GetLastError());
        goto Cleanup;
    }

    // Initialize an EXPLICIT_ACCESS structure for an ACE.
    // The ACE will allow the Administrators group full access to
    // the key.
    ea[1].grfAccessPermissions = KEY_ALL_ACCESS;
    ea[1].grfAccessMode = SET_ACCESS;
    ea[1].grfInheritance= NO_INHERITANCE;
    ea[1].Trustee.TrusteeForm = TRUSTEE_IS_SID;
    ea[1].Trustee.TrusteeType = TRUSTEE_IS_GROUP;
    ea[1].Trustee.ptstrName  = (LPTSTR) pAdminSID;

    // Create a new ACL that contains the new ACEs.
    dwRes = SetEntriesInAcl(2, ea, NULL, &pACL);
    if (ERROR_SUCCESS != dwRes)
    {
        _tprintf(_T("SetEntriesInAcl Error %u\n"), GetLastError());
        goto Cleanup;
    }
```

```
    // Initialize a security descriptor.
    pSD = (PSECURITY_DESCRIPTOR) LocalAlloc(LPTR,
                            SECURITY_DESCRIPTOR_MIN_LENGTH);
    if (NULL == pSD)
    {
        _tprintf(_T("LocalAlloc Error %u\n"), GetLastError());
        goto Cleanup;
    }

    if (!InitializeSecurityDescriptor(pSD,
            SECURITY_DESCRIPTOR_REVISION))
    {
        _tprintf(_T("InitializeSecurityDescriptor Error %u\n"),
                                GetLastError());
        goto Cleanup;
    }

    // Add the ACL to the security descriptor.
    if (!SetSecurityDescriptorDacl(pSD,
            TRUE,     // bDaclPresent flag
            pACL,
            FALSE))   // not a default DACL
    {
        _tprintf(_T("SetSecurityDescriptorDacl Error %u\n"),
                GetLastError());
        goto Cleanup;
    }

    // Initialize a security attributes structure.
    sa.nLength = sizeof (SECURITY_ATTRIBUTES);
    sa.lpSecurityDescriptor = pSD;
    sa.bInheritHandle = FALSE;

    // Use the security attributes to set the security descriptor
    // when you create a key.
    lRes = RegCreateKeyEx(HKEY_CURRENT_USER, _T("mykey"), 0, _T(""), 0,
            KEY_READ | KEY_WRITE, &sa, &hkSub, &dwDisposition);
    _tprintf(_T("RegCreateKeyEx result %u\n"), lRes );

Cleanup:

    if (pEveryoneSID)
        FreeSid(pEveryoneSID);
    if (pAdminSID)
```

```
            FreeSid(pAdminSID);
        if (pACL)
            LocalFree(pACL);
        if (pSD)
            LocalFree(pSD);
        if (hkSub)
            RegCloseKey(hkSub);

        return;

    }
```