# Access Masks

Unless otherwise specified, references in this document are from the Windows Software Development Toolkit (SDK) (v7.0) and the Windows Driver Kit (WDK) (v7600.16385.0), which defines at least 46 mask sets and 500+ bit flag value declarations. The document attached to this blog entry contains extensive toolkit header extracts and MSDN Library cross-references.

The `ACCESS_MASK` structure is defined in http://msdn.microsoft.com/en-us/library/aa374892.aspx, and is declared in WinNT.h ([SDK], [WDK]).

The `ACCESS_MASK` data type is a 32-bit (double word) flag set that is the primary means of specifying (encoding) the requested or granted access to a (securable) object by a user. Specifically, it is a value that defines standard, specific, and generic rights used in [Access Control Entries] (ACEs) as well as parameters to Windows API functions.

A securable object is an object that can have a security descriptor ([Security Descriptors]). All named Windows objects are securable. Some unnamed objects, such as process and thread objects, can have security descriptors too. For most securable objects, you can specify an object's security descriptor in the function call that creates the object. For example, you can specify a security descriptor in the CreateFile and CreateProcess functions.

Each type of securable object defines its own set of specific access rights and its own mapping of generic access rights. For information about the specific and generic access rights for each type of securable object, see the overview for that type of object [Securable Objects].

The following is a list of some common securable objects, from "Windows Internals®, Fifth Edition", Microsoft Press, 2009, Library of Congress Control Number 2009927697, p458:

1. Files, directories and volumes (NTFS file system)
2. Devices
3. Mailslots
4. Named and anonymous pipes
5. Jobs
6. Processes
7. Threads
8. Events, keyed events and event pairs
9. Mutexes, semaphores
10. Shared memory sections
11. I/O completion ports
12. LPC ports
13. Waitable timers
14. Access tokens

15. Windows stations
16. Desktops
17. Network shares
18. Services

# 1 ACCESS_MASK

The general rule – for proper Windows executable citizenship – when setting ACCESS_MASK flags, is to always use as few flags as possible. Securable object creation is, of course, a case where this rule may not apply. Other popular (and no doubt necessary) exceptions involve temporary files, configuration file management, backup operations and so on.

Of special note is the `MAXIMUM_ALLOWED` bit, which is generally used with the AccessCheck(…) function to determine whether a security descriptor grants a specified set of access rights to the client identified by an access token. Typically, server applications use this function to check access to a private object.

As noted previously, each type of securable object has its own defined set of *specific access rights*; in almost all cases, these rights include a definition for all bits valid on the underlying object. These should ideally match the result of a call to AccessCheck(…) – if and only if the caller has full control permissions on the object. For example, File Security and Access Rights defines the following.

```
#define FILE_ALL_ACCESS (STANDARD_RIGHTS_REQUIRED | SYNCHRONIZE | 0x1FF)
```

`STANDARD_RIGHTS_REQUIRED` is a mask meant to be used when defining access masks for object types – that is to say, the set of access masks that all securable objects must support (this value is the set of flags defined as `DELETE`, `READ_CONTROL`, `WRITE_DAC` and `WRITE_OWNER`).

From WinNT.h ([SDK], [WDK]) (with modifications for clarity, and to show the ‘MA’ bit):

```
//
//  Define the access mask as a longword sized structure divided up as
//  follows:
//
//    3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
//    1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
//   +-------+-------+---------------+--------------------------------+
//   |Generic|Special| StandardRights|         SpecificRights         |
//   |Rights |Rights |   Rights      |                                |
//   +-------+-------+---------------+--------------------------------+
//   |G|G|G|G|Res|M|A|Res’d|S|W|W|R|D|                                |
//   |R|W|E|A|’d |A|S|Res’d|Y|O|D|C|E|                                |
//   +-------+-------+---------------+--------------------------------+
//
//      typedef struct _ACCESS_MASK {
//          WORD  SpecificRights;
//          BYTE  StandardRights;
//          BYTE  AccessSystemAcl : 1;
//          BYTE  Reserved        : 3;
//          BYTE  GenericAll      : 1;
//          BYTE  GenericExecute  : 1;
//          BYTE  GenericWrite    : 1;
//          BYTE  GenericRead     : 1;
//      } ACCESS_MASK;
//      typedef ACCESS_MASK *PACCESS_MASK;
//
//  but to make life simple for programmer's we'll allow them to specify
//  a desired access mask by simply OR'ing together multiple single rights
//  and treat an access mask as a DWORD.  For example
//
//      DesiredAccess = DELETE | READ_CONTROL
//
//  So we'll declare ACCESS_MASK as DWORD
//
// begin_wdm
typedef DWORD ACCESS_MASK;
typedef ACCESS_MASK *PACCESS_MASK;
```

The bits in positions 28 through 31 are generic rights that can be mapped to object-specific user rights by the resource manager for the requested object. The mapping of these rights is implementation-specific.

The bits in positions 24 and 25 are for maximum allowed and access system security rights.

The bits in positions 0 through 15 are standard rights that are common to all objects.

The document attached to this blog entry contains a variant of the above, with all flags defined.

## 2 Generic Access Rights

The four high-order bits of an `ACCESS_MASK` specify generic access rights. Each type of securable object maps these bits to a set of its standard and object-specific access rights.

For example, a Windows file object maps the `GENERIC_READ` bit to the `READ_CONTROL` and `SYNCHRONIZE` standard access rights and to the `FILE_READ_DATA`, `FILE_READ_EA`, and `FILE_READ_ATTRIBUTES` object-specific access rights. Other types of objects map the `GENERIC_READ` bit to whatever set of access rights is appropriate for that type of object.

You can use generic access rights to specify the type of access you need when you are opening a handle to an object. This is typically simpler than specifying all the corresponding standard and specific rights.

Applications that define private securable objects can also use the generic access rights.

| Constant | Value | Description |
|---|---|---|
| GENERIC_ALL | 0x10000000 | The right to read, write, and execute the object. |
| GENERIC_EXECUTE | 0x20000000 | The right to execute or alternatively look into the object. |
| GENERIC_WRITE | 0x40000000 | The right to write the information maintained by the object. |
| GENERIC_READ | 0x80000000 | The right to read the information maintained by the object. |

From WinNT.h ([SDK], [WDK]):

```
//
//  These are the generic Rights
//
#define GENERIC_READ                     (0x80000000L)
#define GENERIC_WRITE                    (0x40000000L)
#define GENERIC_EXECUTE                  (0x20000000L)
#define GENERIC_ALL                      (0x10000000L)
```

# 3 ACCESS_SYSTEM_SECURITY

Access system security (`ACCESS_SYSTEM_SECURITY`) is used to request or indicate access to a system access control list (SACL). This is also known as the SACL Access Right ([SACL Access Right]).

This type of access requires the calling process to have the `SE_SECURITY_NAME` (Manage auditing and security log) privilege. If this flag is set in the access mask of an audit access ACE (successful or unsuccessful access), the SACL access will be audited ([ACCESS_MASK]).

The system grants this access right only if the `SE_SECURITY_NAME` privilege is enabled in the access token of the requesting thread ([SACL Access Right]).

If a backup application must have access to the system-level access control settings, the `ACCESS_SYSTEM_SECURITY` flag must be specified in the `dwDesiredAccess` parameter value passed to CreateFile ([File Access Rights]).

Privileges determine the type of system operations that a user account can perform. An administrator assigns privileges to user and group accounts. Each user's privileges include those granted to the user and to the groups to which the user belongs. The `SE_BACKUP_NAME` and `SE_RESTORE_NAME` privileges are required to grant `ACCESS_SYSTEM_SECURITY` for backup operations ([Privilege Constants]).

To read the SACL from a security descriptor, the calling process must have been granted `ACCESS_SYSTEM_SECURITY` access when the handle was opened. The proper way to get this access is to enable the `SE_SECURITY_NAME` privilege ([Privilege Constants]) in the caller's current token, open the handle for `ACCESS_SYSTEM_SECURITY` access, and then disable the privilege ([GetSecurityInfo]).

From WinNT.h ([SDK], [WDK]):

```
//
// AccessSystemAcl access type
//
#define ACCESS_SYSTEM_SECURITY           (0x01000000L)
```

# 4 MAXIMUM_ALLOWED

The `MAXIMUM_ALLOWED` access type is generally used with the AccessCheck(…) function to determine whether a security descriptor grants a specified set of access rights to the client identified by an access token. Typically, server applications use this function to check access to a private object. Note that `MAXIMUM_ALLOWED` cannot be used in an ACE (see access control entries).

When using AccessCheck(…) for this purpose, perform the following steps:

1. Obtain a security descriptor that has owner, group, and DACL information.

   If you are not impersonating a client, obtain an impersonation token by calling ImpersonateSelf(…). This token is passed as the client token in the AccessCheck(…) call.

2. Create a generic mapping structure ([GENERIC_MAPPING]). The contents of this structure will vary depending on the object being used.

3. Call AccessCheck(…) and request `MAXIMUM_ALLOWED` as the desired access.

   If the AccessCheck(…) call succeeds after the above steps have been completed, the `GrantedAccess` parameter contains a mask of the object-specific rights that are granted by the security descriptor.

When used in an Access Request operation, the Maximum Allowed bit grants the requestor the maximum permissions allowed to the object through the Access Check Algorithm. This bit can only be requested, it cannot be set in an ACE ([MS-DTYP 2.4.3]).

When used to set the Security Descriptor on an object, the Maximum Allowed bit in the `SECURITY_DESCRIPTOR` has no meaning. The MA bit SHOULD NOT be set and SHOULD be ignored when part of a `SECURITY_DESCRIPTOR` structure ([MS-DTYP 2.4.3]).

From WinNT.h ([SDK], [WDK]):

```
//
// MaximumAllowed access type
//
#define MAXIMUM_ALLOWED                      (0x02000000L)
```

# 5 Standard Access Rights

Standard access rights are those rights corresponding to operations common to most types of securable objects.

| Access Right | Value | Description |
|---|---|---|
| DELETE | 0x00010000 | The right to delete the object. |

| Access Right | Value | Description |
|---|---|---|
| READ_CONTROL | 0x00020000 | The right to read the information in the file or directory object's security descriptor. This does not include the information in the SACL. |
| WRITE_DAC | 0x00040000 | The right to modify the DACL in the object's security descriptor. |
| WRITE_OWNER | 0x00080000 | The right to change the owner in the object's security descriptor. |
| SYNCHRONIZE | 0x00100000 | The right to use the object for synchronization. This enables a thread to wait until the object is in the signaled state. Some object types do not support this access right. |
| STANDARD_RIGHTS_REQUIRED | 0x000F0000 | Combines DELETE, READ_CONTROL, WRITE_DAC, and WRITE_OWNER access. [1] |
| STANDARD_RIGHTS_READ | READ_CONTROL | Currently defined to equal READ_CONTROL. |
| STANDARD_RIGHTS_WRITE | READ_CONTROL | Currently defined to equal READ_CONTROL. |
| STANDARD_RIGHTS_EXECUTE | READ_CONTROL | Currently defined to equal READ_CONTROL. |
| STANDARD_RIGHTS_ALL | 0x001F0000 | Combines DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER, and SYNCHRONIZE access. |

[1] `STANDARD_RIGHTS_REQUIRED` is a mask meant to be used when defining access masks for object types - it's the set of access masks that all securable objects must support.

From WinNT.h ([SDK], [WDK]):

```
//
//  The following are masks for the predefined standard access types
//
#define DELETE                           (0x00010000L)
#define READ_CONTROL                     (0x00020000L)
#define WRITE_DAC                        (0x00040000L)
#define WRITE_OWNER                      (0x00080000L)
#define SYNCHRONIZE                      (0x00100000L)

#define STANDARD_RIGHTS_REQUIRED         (0x000F0000L)

#define STANDARD_RIGHTS_READ             (READ_CONTROL)
#define STANDARD_RIGHTS_WRITE            (READ_CONTROL)
#define STANDARD_RIGHTS_EXECUTE          (READ_CONTROL)

#define STANDARD_RIGHTS_ALL              (0x001F0000L)
```

# 6 Specific Access Rights

Each type of securable object has a set of access rights that correspond to operations specific to that type of object. These rights occupy the low order 16 bits of the `ACCESS_MASK` data type.

From WinNT.h ([SDK], [WDK]):

```
#define SPECIFIC_RIGHTS_ALL                 (0x0000FFFFL)
```

The remainder of this text lists a quorum of `ACCESS_MASK` definition sets available online (MSDN), as well as in the Windows SDK and the Windows Driver Kit).

## 6.1 Administrations and Management

### 6.1.1 Active Directory Access Rights and Service Interfaces (ADSI)

- See this blog entry: Active Directory Technical Specification Control Access Rights Concordance
- ADS_RIGHTS_ENUM Enumeration
- Directory Services Access Rights
- [MS-ADTS 5.1.3.2 Access Rights
- [SDK] NtDsAPI.h

## 6.2 Component Development (COM)

- ACTRL_ACCESS_ENTRY Structure
- [SDK], [WDK] AccCtrl.h

## 6.3 Diagnostics

### 6.3.1 WMI Access Rights

- EventAccessControl Function
- [WDK] wmistr.h

### 6.3.2 Namespace Access Rights

- Namespace Access Rights Constants
- [SDK], [WDK] WbemCli.h

## 6.4 Networking

### 6.4.1 Fax Service

- Fax Client User Access Rights
- Specific Fax Access Rights
- Generic Fax Access Rights
- Required Fax Access Rights by Function
- [SDK] WinFax.h

### 6.4.2 Windows Filtering Platform (WFP)

- WFP Access Control
- [SDK], [WDK] fwpmk.h
- [WDK] fwpmu.h

### 6.4.3 Wireless Networking

- WlanGetSecuritySettings Function
- WlanSetSecuritySettings Function
- [SDK] wlanapi.h

## 6.5 Authorization

### 6.5.1 Token

- Access Rights for Access-Token Objects
- [SDK], [WDK] WinNT.h
- [WDK] ntifs.h

### 6.5.2 Access Control Entry (ACE)

- ACE
- ACE Strings
- [MS-DTYP] 2.4.4 ACE
- [SDK] Iads.h

### 6.5.3 Audit

- AuditSetSystemPolicy Function
- [SDK], [WDK] NTSecAPI.h

### 6.5.4 Local Security Authority (LSA)

- Security Management Objects
- Account Object Access Rights
- Policy Object Access Rights
- TrustedDomain Object Access Rights
- Private Data Object Access Rights
- [WDK] ntlsa.h

### 6.5.5 Security Accounts Manager Alias Specific Access Rights

- [MS-SAMR]: Security Account Manager (SAM) Remote Protocol Specification (Client-to-Server)
- [MS-SAMR] 2.2.1.6 Alias ACCESS_MASK Values
- [MS-SAMR] 2.2.1.4 Domain ACCESS_MASK Values
- [MS-SAMR] 2.2.1.5 Group ACCESS_MASK Values
- [MS-SAMR] 2.2.1.3 Server ACCESS_MASK Values
- [MS-SAMR] 2.2.1.7 User ACCESS_MASK Values
- [WDK] ntsam.h

## 6.6 System Services

#### *6.6.1.1 Console*

- Console Buffer Security and Access Rights

## 6.6.2 DLLs, Processes and Threads

### *6.6.2.1 Process*

- Process Security and Access Rights
- [SDK], [WDK] WinNT.h

### *6.6.2.2 Job*

- Job Object Security and Access Rights
- [SDK], [WDK] WinNT.h

### *6.6.2.3 Thread*

- Thread Security and Access Rights
- [SDK], [WDK] WinNT.h

## 6.6.3 Window Station

- Window Station Security and Access Rights
- [SDK], [WDK] WinUser.h

## 6.6.4 Desktop

- Desktop Security and Access Rights
- [SDK], [WDK] WinUser.h

## 6.6.5 Services

### *6.6.5.1 Service Control Manager*

- Service Security and Access Rights
- [SDK], [WDK] WinSvc.h

### *6.6.5.2 Service*

- Service Security and Access Rights
- [SDK], [WDK] WinSvc.h

## 6.6.6 Synchronization Objects

- Synchronization Object Security and Access Rights

### *6.6.6.1 Event*

- Synchronization Object Security and Access Rights
- [SDK], [WDK] WinNT.h

### *6.6.6.2 Mutex*

- Synchronization Object Security and Access Rights
- [SDK], [WDK] WinNT.h
- [SDK], [WDK] winbase.h

### *6.6.6.3 Semaphore*

- Synchronization Object Security and Access Rights

- [SDK], [WDK] WinNT.h

#### *6.6.6.4 Timer*

- Synchronization Object Security and Access Rights
- [SDK], [WDK] WinNT.h

### 6.6.7 File Services

#### *6.6.7.1 File Access Rights*

- File Security and Access Rights
- [SDK], [WDK] WinNT.h

#### *6.6.7.2 File Mapping*

- File Mapping Security and Access Rights
- ZwCreateSection
- [SDK], [WDK] WinNT.h

#### *6.6.7.3 Pipes*

- About Pipes

##### 6.6.7.3.1 Anonymous Pipes

- Anonymous Pipe Security and Access Rights

##### 6.6.7.3.2 Named Pipes

- Named Pipe Security and Access Rights
- [SDK], [WDK] WinNT.h

### 6.6.8 Registry

- Registry Key Security and Access Rights
- [SDK], [WDK] WinNT.h

### 6.6.9 Kernel Transaction Manager (KTM)

- Kernel Transaction Manager Constants

#### *6.6.9.1 Enlistment (KTM)*

- Enlistment Access Masks
- [SDK], [WDK] WinNT.h

#### *6.6.9.2 Resource Manager (KTM)*

- Resource Manager Access Masks
- [SDK], [WDK] WinNT.h

#### *6.6.9.3 Transaction (KTM)*

- Transaction Access Masks
- [SDK], [WDK] WinNT.h

#### *6.6.9.4 Transaction Manager*

- Transaction Manager Access Masks
- [SDK], [WDK] WinNT.h

### 6.6.10 Memory Management

- ZwCreateSection (Section Access Rights)
- Section Objects and Views
- [SDK], [WDK] WinNT.h

### 6.6.11 Installable File System Drivers (Windows Driver Kit)

- FltBuildDefaultSecurityDescriptor (FLT Access Rights)
- [WDK] fltKernel.h

## 6.7 Open Specifications

### 6.7.1 Printing (Windows Communication Protocols (MCPP))

- [MS-RPRN]: Print System Remote Protocol Specification
- Using the Windows Headers
- [MS-RPRN] 2.2.3.1 Access Values (Print Jobs)
- [MS-RPRN] 2.2.3.1 Access Values (Print Server Printer)
- [MS-RPRN] 2.2.3.1 Access Values (Print Server Remote Protocol)
- [SDK], [WDK] WinSpool.h

### 6.7.2 Windows Internet Naming Service (WINS)

- [MS-RAIW]: Remote Administrative Interface: WINS Specification
- [MS-RAIW] 2.1.1 Server Security Settings