Contents

NetQueryDisplayInformation function

NetUserAdd function

NetUserChangePassword function

NetUserDel function

NetUserEnum function

NetUserGetGroups function

NetUserGetInfo function

NetUserGetLocalGroups function

NetUserModalsGet function

NetUserModalsSet function

NetUserSetGroups function

NetUserSetInfo function

NetValidatePasswordPolicy function

NetValidatePasswordPolicyFree function

USER_INFO_0 structure

USER_INFO_1 structure

USER_INFO_10 structure

USER_INFO_1003 structure

USER_INFO_1005 structure

USER_INFO_1006 structure

USER_INFO_1007 structure

USER_INFO_1008 structure

USER_INFO_1009 structure

USER_INFO_1010 structure

USER_INFO_1011 structure

USER_INFO_1012 structure

USER_INFO_1013 structure

USER_INFO_1014 structure

USER_INFO_1017 structure

USER_INFO_1018 structure

USER_INFO_1020 structure

USER_INFO_1023 structure

Network Management

6/30/2020 • 26 minutes to read • Edit Online

Overview of the Network Management technology.

To develop Network Management, you need these headers:

- atacct.h
- lmalert.h
- lmapibuf.h
- lmat.h
- lmaudit.h
- lmconfig.h
- lmerrlog.h
- lmjoin.h
- lmmsg.h
- lmremutl.h
- lmserver.h
- lmsvc.h
- lmuse.h
- lmwksta.h

For programming guidance for this technology, see:

- Network Management

Enumerations

TITLE	DESCRIPTION
DSREG_JOIN_TYPE	Specifies the possible ways that a device can be joined to Microsoft Azure Active Directory.

Functions

TITLE	DESCRIPTION
ALERT_OTHER_INFO	The ALERT_OTHER_INFO macro returns a pointer to the alert-specific data in an alert message. The data follows a STD_ALERT structure, and can be an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.
ALERT_VAR_DATA	The ALERT_VAR_DATA macro returns a pointer to the variable-length portion of an alert message. Variable-length data can follow an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.
GetNetScheduleAccountInformation	The GetNetScheduleAccountInformation function retrieves the AT Service account name.

TITLE	DESCRIPTION
NetAccessAdd	Not supported.
NetAccessDel	Not supported.
NetAccessEnum	Not supported.
NetAccessGetInfo	Not supported.
NetAccessGetUserPerms	Not supported.
NetAccessSetInfo	Not supported.
NetAddAlternateComputerName	Adds an alternate name for the specified computer.
NetAlertRaise	The NetAlertRaise function notifies all registered clients when a particular event occurs.
NetAlertRaiseEx	The NetAlertRaiseEx function notifies all registered clients when a particular event occurs. You can call this extended function to simplify the sending of an alert message because NetAlertRaiseEx does not require that you specify a STD_ALERT structure.
NetApiBufferAllocate	The NetApiBufferAllocate function allocates memory from the heap. Use this function only when compatibility with the NetApiBufferFree function is required. Otherwise, use the memory management functions.
NetApiBufferFree	The NetApiBufferFree function frees the memory that the NetApiBufferAllocate function allocates. Applications should also call NetApiBufferFree to free the memory that other network management functions use internally to return information.
NetApiBufferReallocate	The NetApiBufferReallocate function changes the size of a buffer allocated by a previous call to the NetApiBufferAllocate function.
NetApiBufferSize	The NetApiBufferSize function returns the size, in bytes, of a buffer allocated by a call to the NetApiBufferAllocate function.
NetAuditClear	The NetAuditClear function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetAuditRead	The NetAuditRead function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetAuditWrite	The NetAuditWrite function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

TITLE	DESCRIPTION
NetConfigGet	The NetConfigGet function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.
NetConfigGetAll	The NetConfigGetAll function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.
NetConfigSet	The NetConfigSet function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.
NetCreateProvisioningPackage	Creates a provisioning package that provisions a computer account for later use in an offline domain join operation. The package may also contain information about certificates and policies to add to the machine during provisioning.
NetEnumerateComputerNames	Enumerates names for the specified computer.
NetErrorLogClear	The NetErrorLogClear function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetErrorLogRead	The NetErrorLogRead function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetErrorLogWrite	The NetErrorLogWrite function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetFreeAadJoinInformation	Frees the memory allocated for the specified DSREG_JOIN_INFO structure, which contains join information for a tenant and which you retrieved by calling the NetGetAadJoinInformation function.
NetGetAadJoinInformation	Retrieves the join information for the specified tenant. This function examines the join information for Microsoft Azure Active Directory and the work account that the current user added.
NetGetAnyDCName	The NetGetAnyDCName function returns the name of any domain controller (DC) for a domain that is directly trusted by the specified server.
NetGetDCName	The NetGetDCName function returns the name of the primary domain controller (PDC). It does not return the name of the backup domain controller (BDC) for the specified domain. Also, you cannot remote this function to a non-PDC server.
NetGetDisplayInformationIndex	The NetGetDisplayInformationIndex function returns the index of the first display information entry whose name begins with a specified string or whose name alphabetically follows the string.

TITLE	DESCRIPTION
NetGetJoinableOUs	The NetGetJoinableOUs function retrieves a list of organizational units (OUs) in which a computer account can be created.
NetGetJoinInformation	The NetGetJoinInformation function retrieves join status information for the specified computer.
NetGroupAdd	The NetGroupAdd function creates a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupAddUser	The NetGroupAddUser function gives an existing user account membership in an existing global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupDel	The NetGroupDel function deletes a global group from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupDelUser	The NetGroupDelUser function removes a user from a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupEnum	The NetGroupEnum function retrieves information about each global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupGetInfo	The NetGroupGetInfo function retrieves information about a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupGetUsers	The NetGroupGetUsers function retrieves a list of the members in a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupSetInfo	The NetGroupSetInfo function sets the parameters of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupSetUsers	The NetGroupSetUsers function sets the membership for the specified global group.
NetJoinDomain	The NetJoinDomain function joins a computer to a workgroup or domain.

TITLE	DESCRIPTION
NetLocalGroupAdd	The NetLocalGroupAdd function creates a local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupAddMember	The NetLocalGroupAddMember function is obsolete. You should use the NetLocalGroupAddMembers function instead.
NetLocalGroupAddMembers	The NetLocalGroupAddMembers function adds membership of one or more existing user accounts or global group accounts to an existing local group.
NetLocalGroupDel	The NetLocalGroupDel function deletes a local group account and all its members from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupDelMember	The NetLocalGroupDelMember function is obsolete. You should use the NetLocalGroupDelMembers function instead.
NetLocalGroupDelMembers	The NetLocalGroupDelMembers function removes one or more members from an existing local group. Local group members can be users or global groups.
NetLocalGroupEnum	The NetLocalGroupEnum function returns information about each local group account on the specified server.
NetLocalGroupGetInfo	The NetLocalGroupGetInfo function retrieves information about a particular local group account on a server.
NetLocalGroupGetMembers	The NetLocalGroupGetMembers function retrieves a list of the members of a particular local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupSetInfo	The NetLocalGroupSetInfo function changes the name of an existing local group. The function also associates a comment with a local group.
NetLocalGroupSetMembers	The NetLocalGroupSetMembers function sets the membership for the specified local group.
NetMessageBufferSend	The NetMessageBufferSend function sends a buffer of information to a registered message alias.
NetMessageNameAdd	The NetMessageNameAdd function registers a message alias in the message name table. The function requires that the messenger service be started.
NetMessageNameDel	The NetMessageNameDel function deletes a message alias in the message name table. The function requires that the messenger service be started.

TITLE	DESCRIPTION
NetMessageNameEnum	The NetMessageNameEnum function lists the message aliases that receive messages on a specified computer. The function requires that the messenger service be started.
NetMessageNameGetInfo	The NetMessageNameGetInfo function retrieves information about a particular message alias in the message name table. The function requires that the messenger service be started.
NetProvisionComputerAccount	Provisions a computer account for later use in an offline domain join operation.
NetQueryDisplayInformation	The NetQueryDisplayInformation function returns user account, computer, or group account information. Call this function to quickly enumerate account information for display in user interfaces.
NetRemoteComputerSupports	The NetRemoteComputerSupports function queries the redirector to retrieve the optional features the remote system supports.
NetRemoteTOD	The NetRemoteTOD function returns the time of day information from a specified server.
NetRemoveAlternateComputerName	Removes an alternate name for the specified computer.
NetRenameMachineInDomain	The NetRenameMachineInDomain function changes the name of a computer in a domain.
NetRequestOfflineDomainJoin	Executes locally on a machine to modify a Windows operating system image mounted on a volume.
NetRequestProvisioningPackageInstall	Executes locally on a machine to modify a Windows operating system image mounted on a volume.
NetScheduleJobAdd	The NetScheduleJobAdd function submits a job to run at a specified future time and date. This function requires that the schedule service be started on the computer to which the job is submitted.
NetScheduleJobDel	The NetScheduleJobDel function deletes a range of jobs queued to run at a computer. This function requires that the schedule service be started at the computer to which the job deletion request is being sent.
NetScheduleJobEnum	The NetScheduleJobEnum function lists the jobs queued on a specified computer. This function requires that the schedule service be started.
NetScheduleJobGetInfo	The NetScheduleJobGetInfo function retrieves information about a particular job queued on a specified computer. This function requires that the schedule service be started.
NetServerComputerNameAdd	The NetServerComputerNameAdd function enumerates the transports on which the specified server is active, and binds the emulated server name to each of the transports.

TITLE	DESCRIPTION
NetServerComputerNameDel	The NetServerComputerNameDel function causes the specified server to cease supporting the emulated server name set by a previous call to the NetServerComputerNameAdd function. The function does this by unbinding network transports from the emulated name.
NetServerDiskEnum	The NetServerDiskEnum function retrieves a list of disk drives on a server. The function returns an array of three-character strings (a drive letter, a colon, and a terminating null character).
NetServerEnum	The NetServerEnum function lists all servers of the specified type that are visible in a domain.
NetServerGetInfo	The NetServerGetInfo function retrieves current configuration information for the specified server.
NetServerSetInfo	The NetServerSetInfo function sets a server's operating parameters; it can set them individually or collectively. The information is stored in a way that allows it to remain in effect after the system has been reinitialized.
NetServerTransportAdd	The NetServerTransportAdd function binds the server to the transport protocol.
NetServerTransportAddEx	The NetServerTransportAddEx function binds the specified server to the transport protocol.
NetServerTransportDel	The NetServerTransportDel function unbinds (or disconnects) the transport protocol from the server. Effectively, the server can no longer communicate with clients using the specified transport protocol (such as TCP or XNS).
NetServerTransportEnum	The NetServerTransportEnum function supplies information about transport protocols that are managed by the server.
NetServiceControl	The NetServiceControl function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceEnum	The NetServiceEnum function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceGetInfo	The NetServiceGetInfo function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceInstall	The NetServiceInstall function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetSetPrimaryComputerName	Sets the primary computer name for the specified computer.

TITLE	DESCRIPTION
NetUnjoinDomain	The NetUnjoinDomain function unjoins a computer from a workgroup or a domain.
NetUseAdd	The NetUseAdd function establishes a connection between the local computer and a remote server.
NetUseDel	The NetUseDel function ends a connection to a shared resource.
NetUseEnum	The NetUseEnum function lists all current connections between the local computer and resources on remote servers.
NetUseGetInfo	The NetUseGetInfo function retrieves information about a connection to a shared resource.
NetUserAdd	The NetUserAdd function adds a user account and assigns a password and privilege level.
NetUserChangePassword	The NetUserChangePassword function changes a user's password for a specified network server or domain.
NetUserDel	The NetUserDel function deletes a user account from a server.
NetUserEnum	The NetUserEnum function retrieves information about all user accounts on a server.
NetUserGetGroups	The NetUserGetGroups function retrieves a list of global groups to which a specified user belongs.
NetUserGetInfo	The NetUserGetInfo function retrieves information about a particular user account on a server.
NetUserGetLocalGroups	The NetUserGetLocalGroups function retrieves a list of local groups to which a specified user belongs.
NetUserModalsGet	The NetUserModalsGet function retrieves global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetUserModalsSet	The NetUserModalsSet function sets global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetUserSetGroups	The NetUserSetGroups function sets global group memberships for a specified user account.
NetUserSetInfo	The NetUserSetInfo function sets the parameters of a user account.

TITLE	DESCRIPTION
NetValidateName	The NetValidateName function verifies that a name is valid for name type specified(computer name, workgroup name, domain name, or DNS computer name).
NetValidatePasswordPolicy	The NetValidatePasswordPolicy function allows an application to check password compliance against an application-provided account database and verify that passwords meet the complexity, aging, minimum length, and history reuse requirements of a password policy.
NetValidatePasswordPolicyFree	The NetValidatePasswordPolicyFree function frees the memory that the NetValidatePasswordPolicy function allocates for the OutputArg parameter, which is a NET_VALIDATE_OUTPUT_ARG structure.
NetWkstaGetInfo	The NetWkstaGetInfo function returns information about the configuration of a workstation.
NetWkstaSetInfo	The NetWkstaSetInfo function configures a workstation with information that remains in effect after the system has been reinitialized.
NetWkstaTransportAdd	Not supported.
NetWkstaTransportDel	Not supported.
NetWkstaTransportEnum	The NetWkstaTransportEnum function supplies information about transport protocols that are managed by the redirector, which is the software on the client computer that generates file requests to the server computer.
NetWkstaUserEnum	The NetWkstaUserEnum function lists information about all users currently logged on to the workstation. This list includes interactive, service and batch logons.
NetWkstaUserGetInfo	The NetWkstaUserGetInfo function returns information about the currently logged-on user. This function must be called in the context of the logged-on user.
NetWkstaUserSetInfo	The NetWkstaUserSetInfo function sets the user-specific information about the configuration elements for a workstation.
SetNetScheduleAccountInformation	The SetNetScheduleAccountInformation function sets the AT Service account name and password. The AT Service account name and password are used as the credentials for scheduled jobs created with NetScheduleJobAdd.

Structures

TITLE	DESCRIPTION

TITLE	DESCRIPTION
ADMIN_OTHER_INFO	The ADMIN_OTHER_INFO structure contains error message information. The NetAlertRaise and NetAlertRaiseEx functions use the ADMIN_OTHER_INFO structure to specify information when raising an administrator's interrupting message.
AT_ENUM	The AT_ENUM structure contains information about a submitted job. The NetScheduleJobEnum function uses this structure to enumerate and return information about an entire queue of submitted jobs.
AT_INFO	The AT_INFO structure contains information about a job.
DSREG_JOIN_INFO	Contains information about how a device is joined to Microsoft Azure Active Directory.
DSREG_USER_INFO	Contains information about a user account that is used to join a device to Microsoft Azure Active Directory.
ERRLOG_OTHER_INFO	The ERRLOG_OTHER_INFO structure contains error log information. The NetAlertRaise and NetAlertRaiseEx functions use the ERRLOG_OTHER_INFO structure to specify information when adding a new entry to the error log.
GROUP_INFO_0	The GROUP_INFO_0 structure contains the name of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
GROUP_INFO_1	The GROUP_INFO_1 structure contains a global group name and a comment to associate with the group.
GROUP_INFO_1002	The GROUP_INFO_1002 structure contains a comment to associate with a global group.
GROUP_INFO_1005	The GROUP_INFO_1005 structure contains the resource attributes associated with a global group.
GROUP_INFO_2	The GROUP_INFO_2 structure contains information about a global group, including name, identifier, and resource attributes.
GROUP_INFO_3	The GROUP_INFO_3 structure contains information about a global group, including name, security identifier (SID), and resource attributes.
GROUP_USERS_INFO_0	The GROUP_USERS_INFO_0 structure contains global group member information.
GROUP_USERS_INFO_1	The GROUP_USERS_INFO_1 structure contains global group member information.
LOCALGROUP_INFO_0	The LOCALGROUP_INFO_0 structure contains a local group name.

TITLE	DESCRIPTION
LOCALGROUP_INFO_1	The LOCALGROUP_INFO_1 structure contains a local group name and a comment describing the local group.
LOCALGROUP_INFO_1002	The LOCALGROUP_INFO_1002 structure contains a comment describing a local group.
LOCALGROUP_MEMBERS_INFO_0	The LOCALGROUP_MEMBERS_INFO_0 structure contains the security identifier (SID) associated with a local group member. The member can be a user account or a global group account.
LOCALGROUP_MEMBERS_INFO_1	The LOCALGROUP_MEMBERS_INFO_1 structure contains the security identifier (SID) and account information associated with the member of a local group.
LOCALGROUP_MEMBERS_INFO_2	The LOCALGROUP_MEMBERS_INFO_2 structure contains the security identifier (SID) and account information associated with a local group member.
LOCALGROUP_MEMBERS_INFO_3	The LOCALGROUP_MEMBERS_INFO_3 structure contains the account name and domain name associated with a local group member.
LOCALGROUP_USERS_INFO_0	The LOCALGROUP_USERS_INFO_0 structure contains local group member information.
MSG_INFO_0	The MSG_INFO_0 structure specifies a message alias.
MSG_INFO_1	The MSG_INFO_1 structure specifies a message alias. This structure exists only for compatibility. Message forwarding is not supported.
NET_DISPLAY_GROUP	The NET_DISPLAY_GROUP structure contains information that an account manager can access to determine information about group accounts.
NET_DISPLAY_MACHINE	The NET_DISPLAY_MACHINE structure contains information that an account manager can access to determine information about computers and their attributes.
NET_DISPLAY_USER	The NET_DISPLAY_USER structure contains information that an account manager can access to determine information about user accounts.
NET_VALIDATE_AUTHENTICATION_INPUT_ARG	A client application passes the NET_VALIDATE_AUTHENTICATION_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests an authentication validation.
NET_VALIDATE_OUTPUT_ARG	The NET_VALIDATE_OUTPUT_ARG structure contains information about persistent password-related data that has changed since the user's last logon as well as the result of the function's password validation check.

TITLE	DESCRIPTION
NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG	A client application passes the NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests a password change validation.
NET_VALIDATE_PASSWORD_HASH	The NET_VALIDATE_PASSWORD_HASH structure contains a password hash.
NET_VALIDATE_PASSWORD_RESET_INPUT_ARG	A client application passes the NET_VALIDATE_PASSWORD_RESET_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests a password reset validation.
NET_VALIDATE_PERSISTED_FIELDS	The NET_VALIDATE_PERSISTED_FIELDS structure contains information about a user's password properties.
NETSETUP_PROVISIONING_PARAMS	The NETSETUP_PROVISIONING_PARAMS structure contains information that is used when creating a provisioning package using the NetCreateProvisionPackage function.
PRINT_OTHER_INFO	Contains information about a print job.
SERVER_INFO_100	The SERVER_INFO_100 structure contains information about the specified server, including the name and platform.
SERVER_INFO_1005	The SERVER_INFO_1005 structure contains a comment that describes the specified server.
SERVER_INFO_101	The SERVER_INFO_101 structure contains information about the specified server, including name, platform, type of server, and associated software.
SERVER_INFO_1010	The SERVER_INFO_1010 structure contains the auto-disconnect time associated with the specified server.
SERVER_INFO_1016	The SERVER_INFO_1016 structure contains information about whether the server is visible to other computers in the same network domain.
SERVER_INFO_1017	The SERVER_INFO_1017 structure contains the network announce rate associated with the specified server.
SERVER_INFO_1018	The SERVER_INFO_1018 structure contains information about how much the announce rate can vary for the specified server.
SERVER_INFO_102	Contains information about the specified server, including name, platform, type of server, attributes, and associated software.
SERVER_INFO_1107	The SERVER_INFO_1107 structure specifies the number of users that can simultaneously log on to the specified server.
SERVER_INFO_1501	The SERVER_INFO_1501 structure specifies the number of files that can be open in one session on the specified server.

TITLE	DESCRIPTION
SERVER_INFO_1502	The SERVER_INFO_1502 structure specifies the maximum number of virtual circuits per client for the specified server.
SERVER_INFO_1503	The SERVER_INFO_1503 structure specifies the number of search operations that can be carried out simultaneously.
SERVER_INFO_1506	The SERVER_INFO_1506 structure contains information about the maximum number of work items the specified server can allocate.
SERVER_INFO_1509	The SERVER_INFO_1509 structure specifies the maximum raw mode buffer size.
SERVER_INFO_1510	The SERVER_INFO_1510 structure specifies the maximum number of users that can be logged on to the specified server using a single virtual circuit.
SERVER_INFO_1511	The SERVER_INFO_1511 structure specifies the maximum number of tree connections that users can make with a single virtual circuit.
SERVER_INFO_1512	The SERVER_INFO_1512 structure contains the maximum size of nonpaged memory that the specified server can allocate at a particular time.
SERVER_INFO_1513	The SERVER_INFO_1513 structure contains the maximum size of pageable memory that the specified server can allocate at a particular time.
SERVER_INFO_1515	The SERVER_INFO_1515 structure specifies whether the server should force a client to disconnect once the client's logon time has expired.
SERVER_INFO_1516	The SERVER_INFO_1516 structure specifies whether the server is a reliable time source.
SERVER_INFO_1518	The SERVER_INFO_1518 structure specifies whether the server is visible to LAN Manager 2.x clients.
SERVER_INFO_1523	The SERVER_INFO_1523 structure specifies the length of time the server retains information about incomplete search operations.
SERVER_INFO_1528	The SERVER_INFO_1528 structure specifies the period of time that the scavenger remains idle before waking up to service requests.
SERVER_INFO_1529	The SERVER_INFO_1529 structure specifies the minimum number of free receive work items the server requires before it begins allocating more items.

TITLE	DESCRIPTION
SERVER_INFO_1530	The SERVER_INFO_1530 structure specifies the minimum number of available receive work items the server requires to begin processing a server message block.
SERVER_INFO_1533	The SERVER_INFO_1533 structure specifies the maximum number of outstanding requests a client can send to the server.
SERVER_INFO_1536	The SERVER_INFO_1536 structure specifies whether the server allows clients to use opportunistic locks (oplocks) on files.
SERVER_INFO_1538	The SERVER_INFO_1538 structure specifies whether several MS-DOS File Control Blocks (FCBs) are placed in a single location.
SERVER_INFO_1539	The SERVER_INFO_1539 structure specifies whether the server processes raw Server Message Blocks (SMBs).
SERVER_INFO_1540	The SERVER_INFO_1540 structure specifies whether the server allows redirected server drives to be shared.
SERVER_INFO_1541	The SERVER_INFO_1541 structure specifies the minimum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.
SERVER_INFO_1542	The SERVER_INFO_1542 structure specifies the maximum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.
SERVER_INFO_1544	The SERVER_INFO_1544 structure specifies the initial number of tree connections to be allocated in the connection table.
SERVER_INFO_1550	The SERVER_INFO_1550 structure specifies the percentage of free disk space remaining before an alert message is sent.
SERVER_INFO_1552	The SERVER_INFO_1552 structure specifies the maximum time allowed for a link delay.
SERVER_INFO_402	Contains information about a specified server.
SERVER_INFO_403	The SERVER_INFO_403 structure contains information about a specified server.
SERVER_INFO_502	The SERVER_INFO_502 structure is obsolete. The structure contains information about a specified server.
SERVER_INFO_503	The SERVER_INFO_503 structure is obsolete. The structure contains information about the specified server.
SERVER_TRANSPORT_INFO_0	The SERVER_TRANSPORT_INFO_0 structure contains information about the specified transport protocol, including name, address, and location on the network.

TITLE	DESCRIPTION
SERVER_TRANSPORT_INFO_1	The SERVER_TRANSPORT_INFO_1 structure contains information about the specified transport protocol, including name and address. This information level is valid only for the NetServerTransportAddEx function.
SERVER_TRANSPORT_INFO_2	The SERVER_TRANSPORT_INFO_2 structure contains information about the specified transport protocol, including the transport name and address. This information level is valid only for the NetServerTransportAddEx function.
SERVER_TRANSPORT_INFO_3	The SERVER_TRANSPORT_INFO_3 structure contains information about the specified transport protocol, including name, address and password (credentials). This information level is valid only for the NetServerTransportAddEx function.
STD_ALERT	The STD_ALERT structure contains the time and date when a significant event occurred.
TIME_OF_DAY_INFO	The TIME_OF_DAY_INFO structure contains information about the time of day from a remote server.
USE_INFO_0	The USE_INFO_0 structure contains the name of a shared resource and the local device redirected to it.
USE_INFO_1	Contains information about the connection between a local device and a shared resource.
USE_INFO_2	The USE_INFO_2 structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, and domain name.
USE_INFO_3	The USE_INFO_3 structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, domain name, and specific flags that describe connection behavior.
USER_INFO_0	The USER_INFO_0 structure contains a user account name.
USER_INFO_1	The USER_INFO_1 structure contains information about a user account, including account name, password data, privilege level, and the path to the user's home directory.
USER_INFO_10	The USER_INFO_10 structure contains information about a user account, including the account name, comments associated with the account, and the user's full name.
USER_INFO_1003	The USER_INFO_1003 structure contains a user password. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1005	The USER_INFO_1005 structure contains a privilege level to assign to a user network account. This information level is valid only when you call the NetUserSetInfo function.

TITLE	DESCRIPTION
USER_INFO_1006	The USER_INFO_1006 structure contains the user's home directory path. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1007	The USER_INFO_1007 structure contains a comment associated with a user network account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1008	The USER_INFO_1008 structure contains a set of bit flags defining several user network account parameters. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1009	The USER_INFO_1009 structure contains the path for a user's logon script file. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1010	The USER_INFO_1010 structure contains a set of bit flags defining the operator privileges assigned to a user network account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1011	The USER_INFO_1011 structure contains the full name of a network user. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1012	The USER_INFO_1012 structure contains a user comment. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1013	The USER_INFO_1013 structure contains reserved information for network accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1014	The USER_INFO_1014 structure contains the names of workstations from which the user can log on. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1017	The USER_INFO_1017 structure contains expiration information for network user accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1018	The USER_INFO_1018 structure contains the maximum amount of disk space available to a network user account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1020	The USER_INFO_1020 structure contains the times during which a user can log on to the network. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1023	The USER_INFO_1023 structure contains the name of the server to which network logon requests should be sent. This information level is valid only when you call the NetUserSetInfo function.

TITLE	DESCRIPTION
USER_INFO_1024	The USER_INFO_1024 structure contains the country/region code for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1025	The USER_INFO_1025 structure contains the code page for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1051	The USER_INFO_1051 structure contains the relative ID (RID) associated with the user account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1052	The USER_INFO_1052 structure contains the path to a network user's profile. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1053	The USER_INFO_1053 structure contains user information for network accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_11	The USER_INFO_11 structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, and other user-related network statistics.
USER_INFO_2	The USER_INFO_2 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, and other user-related network statistics.
USER_INFO_20	Contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's relative ID (RID).
USER_INFO_21	The USER_INFO_21 structure contains a one-way encrypted LAN Manager 2.x-compatible password.
USER_INFO_22	The USER_INFO_22 structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, a one-way encrypted LAN Manager 2.x-compatible password, and other user-related network statistics.
USER_INFO_23	Contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's security identifier (SID).
USER_INFO_24	Contains user account information on an account which is connected to an Internet identity. This information includes the Internet provider name for the user, the user's Internet name, and the user's security identifier (SID).

TITLE	DESCRIPTION
USER_INFO_3	The USER_INFO_3 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, relative identifiers (RIDs), and other user-related network statistics.
USER_INFO_4	The USER_INFO_4 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, security identifier (SID), and other user-related network statistics.
USER_MODALS_INFO_0	The USER_MODALS_INFO_0 structure contains global password information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1	The USER_MODALS_INFO_1 structure contains logon server and domain controller information.
USER_MODALS_INFO_1001	The USER_MODALS_INFO_1001 structure contains the minimum length for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1002	The USER_MODALS_INFO_1002 structure contains the maximum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1003	The USER_MODALS_INFO_1003 structure contains the minimum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1004	The USER_MODALS_INFO_1004 structure contains forced logoff information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1005	The USER_MODALS_INFO_1005 structure contains password history information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1006	The USER_MODALS_INFO_1006 structure contains logon server information.
USER_MODALS_INFO_1007	The USER_MODALS_INFO_1007 structure contains domain controller information.
USER_MODALS_INFO_2	The USER_MODALS_INFO_2 structure contains the Security Account Manager (SAM) domain name and identifier.

TITLE	DESCRIPTION
USER_MODALS_INFO_3	The USER_MODALS_INFO_3 structure contains lockout information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_OTHER_INFO	The USER_OTHER_INFO structure contains user error code information. The NetAlertRaise and NetAlertRaiseEx functions use the USER_OTHER_INFO structure to specify information about an event or condition of interest to a user.
WKSTA_INFO_100	Contains information about a workstation environment, including platform-specific information, the names of the domain and the local computer, and information concerning the operating system.
WKSTA_INFO_101	Contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.
WKSTA_INFO_102	Contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.
WKSTA_INFO_502	The WKSTA_INFO_502 structure is obsolete. The structure contains information about a workstation environment.
WKSTA_TRANSPORT_INFO_0	The WKSTA_TRANSPORT_INFO_0 structure contains information about the workstation transport protocol, such as Wide Area Network (WAN) or NetBIOS.
WKSTA_USER_INFO_0	The WKSTA_USER_INFO_0 structure contains the name of the user on a specified workstation.
WKSTA_USER_INFO_1	The WKSTA_USER_INFO_1 structure contains user information as it pertains to a specific workstation. The information includes the name of the current user and the domains accessed by the workstation.
WKSTA_USER_INFO_1101	The WKSTA_USER_INFO_1101 structure contains information about the domains accessed by a workstation.

atacct.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management atacct.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
GetNetScheduleAccountInformation	The GetNetScheduleAccountInformation function retrieves the AT Service account name.
SetNetScheduleAccountInformation	The SetNetScheduleAccountInformation function sets the AT Service account name and password. The AT Service account name and password are used as the credentials for scheduled jobs created with NetScheduleJobAdd.

GetNetScheduleAccountInformation function (atacct.h)

2/1/2021 • 2 minutes to read • Edit Online

[**GetNetScheduleAccountInformation** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **GetNetScheduleAccountInformation** function retrieves the AT Service account name.

Syntax

```
HRESULT GetNetScheduleAccountInformation(
  LPCWSTR  pwszServerName,
  DWORD    ccAccount,
  WCHAR [] wszAccount
);
```

Parameters

`pwszServerName`

A NULL-terminated wide character string for the name of the computer whose account information is being retrieved.

`ccAccount`

The number of characters, including the NULL terminator, allocated for *wszAccount*. The maximum allowed length for this value is the maximum domain name length plus the maximum user name length plus 2, expressed as DNLEN + UNLEN + 2. (The last two characters are the "" character and the NULL terminator.)

`wszAccount`

An array of wide characters, including the NULL terminator, that receives the account information.

Return value

The return value is an HRESULT. A value of S_OK indicates the function succeeded, and the account information is returned in *wszAccount*. A value of S_FALSE indicates the function succeeded, and the account is the Local System account (no information will be returned in *wszAccount*). Any other return values indicate an error condition.

Remarks

To successfully call the **GetNetScheduleAccountInformation** function, the caller should have read access to the task folder which is usually %windir%\tasks or as defined in the following registry setting:

HKLM\SOFTWARE\Microsoft\SchedulingAgent\TasksFolder

Requirements

Minimum supported client	Windows Vista, Windows XP with SP1 [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	atacct.h
Library	Mstask.lib
DLL	Mstask.dll

See also

SetNetScheduleAccountInformation

SetNetScheduleAccountInformation function (atacct.h)

2/1/2021 • 2 minutes to read • Edit Online

[**SetNetScheduleAccountInformation** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **SetNetScheduleAccountInformation** function sets the AT Service account name and password. The AT Service account name and password are used as the credentials for scheduled jobs created with NetScheduleJobAdd.

Syntax

```
HRESULT SetNetScheduleAccountInformation(
  LPCWSTR pwszServerName,
  LPCWSTR pwszAccount,
  LPCWSTR pwszPassword
);
```

Parameters

`pwszServerName`

A NULL-terminated wide character string for the name of the computer whose account information is being set.

`pwszAccount`

A pointer to a NULL-terminated wide character string for the account. To specify the local system account, set this parameter to **NULL**.

`pwszPassword`

A pointer to a NULL-terminated wide character string for the password. For information about securing password information, see Handling Passwords.

Return value

The return value is an HRESULT. A value of S_OK indicates the account name and password were successfully set. Any other value indicates an error condition.

If the function fails, some of the possible return values are listed below.

RETURN CODE/VALUE	DESCRIPTION
E_ACCESSDENIED 0x080070005	Access was denied. This error is returned if the caller was not a member of the Administrators group. This error is also returned if the *pwszAccount* parameter was not **NULL** indicating a named account not the local system account and the *pwszPassword* parameter was incorrect for the account specified in the *pwszAccount* parameter.

HRESULT_FROM_WIN32(ERROR_INVALID_DATA) 0x08007000d	The data is invalid. This error is returned if the *pwszPassword* parameter was **NULL** or the length of *pwszPassword* parameter string was too long.
SCHED_E_ACCOUNT_NAME_NOT_FOUND 0x80041310	Unable to establish existence of the account specified. This error is returned if the *pwszAccount* parameter was not **NULL** indicating a named account not the local system account and the *pwszAccount* parameter could not be found.

Remarks

The **SetNetScheduleAccountInformation** impersonates the caller. Only members of the local Administrators group on the computer where the schedule account information is being set can successfully execute this function. Note that **NULL** passwords are not allowed.

Requirements

Minimum supported client	Windows Vista, Windows XP with SP1 [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	atacct.h
Library	Mstask.lib
DLL	Mstask.dll

See also

GetNetScheduleAccountInformation

lmaccess.h header

2/1/2021 • 13 minutes to read • Edit Online

This header is used by Developer Notes. For more information, see:

- Developer Notes lmaccess.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
I_NetLogonControl2	Controls various aspects of the Netlogon service.
NetAccessAdd	Not supported.
NetAccessDel	Not supported.
NetAccessEnum	Not supported.
NetAccessGetInfo	Not supported.
NetAccessGetUserPerms	Not supported.
NetAccessSetInfo	Not supported.
NetAddServiceAccount	Creates a standalone managed service account (sMSA) or retrieves the credentials for a group managed service account (gMSA) and stores the account information on the local computer.
NetEnumerateServiceAccounts	Enumerates the standalone managed service accounts (sMSA) on the specified server.
NetGetAnyDCName	The NetGetAnyDCName function returns the name of any domain controller (DC) for a domain that is directly trusted by the specified server.
NetGetDCName	The NetGetDCName function returns the name of the primary domain controller (PDC). It does not return the name of the backup domain controller (BDC) for the specified domain. Also, you cannot remote this function to a non-PDC server.
NetGetDisplayInformationIndex	The NetGetDisplayInformationIndex function returns the index of the first display information entry whose name begins with a specified string or whose name alphabetically follows the string.
NetGroupAdd	The NetGroupAdd function creates a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

TITLE	DESCRIPTION
NetGroupAddUser	The NetGroupAddUser function gives an existing user account membership in an existing global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupDel	The NetGroupDel function deletes a global group from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupDelUser	The NetGroupDelUser function removes a user from a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupEnum	The NetGroupEnum function retrieves information about each global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupGetInfo	The NetGroupGetInfo function retrieves information about a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupGetUsers	The NetGroupGetUsers function retrieves a list of the members in a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupSetInfo	The NetGroupSetInfo function sets the parameters of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetGroupSetUsers	The NetGroupSetUsers function sets the membership for the specified global group.
NetIsServiceAccount	Tests whether the specified standalone managed service account (sMSA) or group managed service account (gMSA) exists in the Netlogon store on the specified server.
NetLocalGroupAdd	The NetLocalGroupAdd function creates a local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupAddMember	The NetLocalGroupAddMember function is obsolete. You should use the NetLocalGroupAddMembers function instead.
NetLocalGroupAddMembers	The NetLocalGroupAddMembers function adds membership of one or more existing user accounts or global group accounts to an existing local group.

TITLE	DESCRIPTION
NetLocalGroupDel	The NetLocalGroupDel function deletes a local group account and all its members from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupDelMember	The NetLocalGroupDelMember function is obsolete. You should use the NetLocalGroupDelMembers function instead.
NetLocalGroupDelMembers	The NetLocalGroupDelMembers function removes one or more members from an existing local group. Local group members can be users or global groups.
NetLocalGroupEnum	The NetLocalGroupEnum function returns information about each local group account on the specified server.
NetLocalGroupGetInfo	The NetLocalGroupGetInfo function retrieves information about a particular local group account on a server.
NetLocalGroupGetMembers	The NetLocalGroupGetMembers function retrieves a list of the members of a particular local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetLocalGroupSetInfo	The NetLocalGroupSetInfo function changes the name of an existing local group. The function also associates a comment with a local group.
NetLocalGroupSetMembers	The NetLocalGroupSetMembers function sets the membership for the specified local group.
NetQueryDisplayInformation	The NetQueryDisplayInformation function returns user account, computer, or group account information. Call this function to quickly enumerate account information for display in user interfaces.
NetQueryServiceAccount	Gets information about the specified managed service account.
NetRemoveServiceAccount	Deletes the specified service account from the Active Directory database if the account is a standalone managed service account (sMSA).
NetUserAdd	The NetUserAdd function adds a user account and assigns a password and privilege level.
NetUserChangePassword	The NetUserChangePassword function changes a user's password for a specified network server or domain.
NetUserDel	The NetUserDel function deletes a user account from a server.
NetUserEnum	The NetUserEnum function retrieves information about all user accounts on a server.

TITLE	DESCRIPTION
NetUserGetGroups	The NetUserGetGroups function retrieves a list of global groups to which a specified user belongs.
NetUserGetInfo	The NetUserGetInfo function retrieves information about a particular user account on a server.
NetUserGetLocalGroups	The NetUserGetLocalGroups function retrieves a list of local groups to which a specified user belongs.
NetUserModalsGet	The NetUserModalsGet function retrieves global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetUserModalsSet	The NetUserModalsSet function sets global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
NetUserSetGroups	The NetUserSetGroups function sets global group memberships for a specified user account.
NetUserSetInfo	The NetUserSetInfo function sets the parameters of a user account.
NetValidatePasswordPolicy	The NetValidatePasswordPolicy function allows an application to check password compliance against an application-provided account database and verify that passwords meet the complexity, aging, minimum length, and history reuse requirements of a password policy.
NetValidatePasswordPolicyFree	The NetValidatePasswordPolicyFree function frees the memory that the NetValidatePasswordPolicy function allocates for the OutputArg parameter, which is a NET_VALIDATE_OUTPUT_ARG structure.

Structures

TITLE	DESCRIPTION
GROUP_INFO_0	The GROUP_INFO_0 structure contains the name of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
GROUP_INFO_1	The GROUP_INFO_1 structure contains a global group name and a comment to associate with the group.
GROUP_INFO_1002	The GROUP_INFO_1002 structure contains a comment to associate with a global group.
GROUP_INFO_1005	The GROUP_INFO_1005 structure contains the resource attributes associated with a global group.

TITLE	DESCRIPTION
GROUP_INFO_2	The GROUP_INFO_2 structure contains information about a global group, including name, identifier, and resource attributes.
GROUP_INFO_3	The GROUP_INFO_3 structure contains information about a global group, including name, security identifier (SID), and resource attributes.
GROUP_USERS_INFO_0	The GROUP_USERS_INFO_0 structure contains global group member information.
GROUP_USERS_INFO_1	The GROUP_USERS_INFO_1 structure contains global group member information.
LOCALGROUP_INFO_0	The LOCALGROUP_INFO_0 structure contains a local group name.
LOCALGROUP_INFO_1	The LOCALGROUP_INFO_1 structure contains a local group name and a comment describing the local group.
LOCALGROUP_INFO_1002	The LOCALGROUP_INFO_1002 structure contains a comment describing a local group.
LOCALGROUP_MEMBERS_INFO_0	The LOCALGROUP_MEMBERS_INFO_0 structure contains the security identifier (SID) associated with a local group member. The member can be a user account or a global group account.
LOCALGROUP_MEMBERS_INFO_1	The LOCALGROUP_MEMBERS_INFO_1 structure contains the security identifier (SID) and account information associated with the member of a local group.
LOCALGROUP_MEMBERS_INFO_2	The LOCALGROUP_MEMBERS_INFO_2 structure contains the security identifier (SID) and account information associated with a local group member.
LOCALGROUP_MEMBERS_INFO_3	The LOCALGROUP_MEMBERS_INFO_3 structure contains the account name and domain name associated with a local group member.
LOCALGROUP_USERS_INFO_0	The LOCALGROUP_USERS_INFO_0 structure contains local group member information.
MSA_INFO_0	Specifies information about a managed service account.
NET_DISPLAY_GROUP	The NET_DISPLAY_GROUP structure contains information that an account manager can access to determine information about group accounts.
NET_DISPLAY_MACHINE	The NET_DISPLAY_MACHINE structure contains information that an account manager can access to determine information about computers and their attributes.

TITLE	DESCRIPTION
NET_DISPLAY_USER	The NET_DISPLAY_USER structure contains information that an account manager can access to determine information about user accounts.
NET_VALIDATE_AUTHENTICATION_INPUT_ARG	A client application passes the NET_VALIDATE_AUTHENTICATION_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests an authentication validation.
NET_VALIDATE_OUTPUT_ARG	The NET_VALIDATE_OUTPUT_ARG structure contains information about persistent password-related data that has changed since the user's last logon as well as the result of the function's password validation check.
NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG	A client application passes the NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests a password change validation.
NET_VALIDATE_PASSWORD_HASH	The NET_VALIDATE_PASSWORD_HASH structure contains a password hash.
NET_VALIDATE_PASSWORD_RESET_INPUT_ARG	A client application passes the NET_VALIDATE_PASSWORD_RESET_INPUT_ARG structure to the NetValidatePasswordPolicy function when the application requests a password reset validation.
NET_VALIDATE_PERSISTED_FIELDS	The NET_VALIDATE_PERSISTED_FIELDS structure contains information about a user's password properties.
NETLOGON_INFO_1	Defines a level-1 control query response from a domain controller.
NETLOGON_INFO_2	Defines a level-2 control query response from a domain controller.
NETLOGON_INFO_3	Defines a level-3 control query response from a domain controller.
NETLOGON_INFO_4	Defines a level-4 control query response from a domain controller.
USER_INFO_0	The USER_INFO_0 structure contains a user account name.
USER_INFO_1	The USER_INFO_1 structure contains information about a user account, including account name, password data, privilege level, and the path to the user's home directory.
USER_INFO_10	The USER_INFO_10 structure contains information about a user account, including the account name, comments associated with the account, and the user's full name.
USER_INFO_1003	The USER_INFO_1003 structure contains a user password. This information level is valid only when you call the NetUserSetInfo function.

TITLE	DESCRIPTION
USER_INFO_1005	The USER_INFO_1005 structure contains a privilege level to assign to a user network account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1006	The USER_INFO_1006 structure contains the user's home directory path. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1007	The USER_INFO_1007 structure contains a comment associated with a user network account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1008	The USER_INFO_1008 structure contains a set of bit flags defining several user network account parameters. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1009	The USER_INFO_1009 structure contains the path for a user's logon script file. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1010	The USER_INFO_1010 structure contains a set of bit flags defining the operator privileges assigned to a user network account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1011	The USER_INFO_1011 structure contains the full name of a network user. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1012	The USER_INFO_1012 structure contains a user comment. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1013	The USER_INFO_1013 structure contains reserved information for network accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1014	The USER_INFO_1014 structure contains the names of workstations from which the user can log on. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1017	The USER_INFO_1017 structure contains expiration information for network user accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1018	The USER_INFO_1018 structure contains the maximum amount of disk space available to a network user account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1020	The USER_INFO_1020 structure contains the times during which a user can log on to the network. This information level is valid only when you call the NetUserSetInfo function.

TITLE	DESCRIPTION
USER_INFO_1023	The USER_INFO_1023 structure contains the name of the server to which network logon requests should be sent. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1024	The USER_INFO_1024 structure contains the country/region code for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1025	The USER_INFO_1025 structure contains the code page for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1051	The USER_INFO_1051 structure contains the relative ID (RID) associated with the user account. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1052	The USER_INFO_1052 structure contains the path to a network user's profile. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_1053	The USER_INFO_1053 structure contains user information for network accounts. This information level is valid only when you call the NetUserSetInfo function.
USER_INFO_11	The USER_INFO_11 structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, and other user-related network statistics.
USER_INFO_2	The USER_INFO_2 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, and other user-related network statistics.
USER_INFO_20	Contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's relative ID (RID).
USER_INFO_21	The USER_INFO_21 structure contains a one-way encrypted LAN Manager 2.x-compatible password.
USER_INFO_22	The USER_INFO_22 structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, a one-way encrypted LAN Manager 2.x-compatible password, and other user-related network statistics.
USER_INFO_23	Contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's security identifier (SID).

TITLE	DESCRIPTION
USER_INFO_24	Contains user account information on an account which is connected to an Internet identity. This information includes the Internet provider name for the user, the user's Internet name, and the user's security identifier (SID).
USER_INFO_3	The USER_INFO_3 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, relative identifiers (RIDs), and other user-related network statistics.
USER_INFO_4	The USER_INFO_4 structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, security identifier (SID), and other user-related network statistics.
USER_MODALS_INFO_0	The USER_MODALS_INFO_0 structure contains global password information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1	The USER_MODALS_INFO_1 structure contains logon server and domain controller information.
USER_MODALS_INFO_1001	The USER_MODALS_INFO_1001 structure contains the minimum length for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1002	The USER_MODALS_INFO_1002 structure contains the maximum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1003	The USER_MODALS_INFO_1003 structure contains the minimum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1004	The USER_MODALS_INFO_1004 structure contains forced logoff information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1005	The USER_MODALS_INFO_1005 structure contains password history information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.
USER_MODALS_INFO_1006	The USER_MODALS_INFO_1006 structure contains logon server information.
USER_MODALS_INFO_1007	The USER_MODALS_INFO_1007 structure contains domain controller information.

TITLE	DESCRIPTION
USER_MODALS_INFO_2	The USER_MODALS_INFO_2 structure contains the Security Account Manager (SAM) domain name and identifier.
USER_MODALS_INFO_3	The USER_MODALS_INFO_3 structure contains lockout information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Enumerations

TITLE	DESCRIPTION
MSA_INFO_LEVEL	Indicates the level of a managed service account.
MSA_INFO_STATE	Indicates the state of a managed service account.

GROUP_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_0** structure contains the name of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _GROUP_INFO_0 {
  LPWSTR grpi0_name;
} GROUP_INFO_0, *PGROUP_INFO_0, *LPGROUP_INFO_0;
```

Members

`grpi0_name`

Pointer to a null-terminated Unicode character string that specifies the name of the global group. For more information, see the following Remarks section.

When you call the NetGroupSetInfo function this member specifies the new name of the global group.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupAdd

NetGroupEnum

NetGroupGetInfo

NetGroupSetInfo

Network Management Overview

GROUP_INFO_1 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_1** structure contains a global group name and a comment to associate with the group.

Syntax

```
typedef struct _GROUP_INFO_1 {
  LPWSTR grpi1_name;
  LPWSTR grpi1_comment;
} GROUP_INFO_1, *PGROUP_INFO_1, *LPGROUP_INFO_1;
```

Members

`grpi1_name`

Pointer to a null-terminated Unicode character string that specifies the name of the global group. For more information, see the following Remarks section.

When you call the NetGroupSetInfo function this member is ignored.

`grpi1_comment`

Pointer to a null-terminated Unicode character string that specifies a remark associated with the global group. This member can be a null string. The comment can contain MAXCOMMENTSZ characters.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupAdd

NetGroupEnum

NetGroupGetInfo

NetGroupSetInfo

Network Management Overview

Network Management Structures

GROUP_INFO_1002 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_1002** structure contains a comment to associate with a global group.

Syntax

```
typedef struct _GROUP_INFO_1002 {
  LPWSTR grpi1002_comment;
} GROUP_INFO_1002, *PGROUP_INFO_1002, *LPGROUP_INFO_1002;
```

Members

`grpi1002_comment`

Pointer to a null-terminated Unicode character string that contains a remark to associate with the global group. This member can be a null string. The comment can contain MAXCOMMENTSZ characters.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupSetInfo

Network Management Overview

Network Management Structures

GROUP_INFO_1005 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_1005** structure contains the resource attributes associated with a global group.

Syntax

```
typedef struct _GROUP_INFO_1005 {
  DWORD grpi1005_attributes;
} GROUP_INFO_1005, *PGROUP_INFO_1005, *LPGROUP_INFO_1005;
```

Members

`grpi1005_attributes`

These attributes are hard-coded to SE_GROUP_MANDATORY, SE_GROUP_ENABLED, and SE_GROUP_ENABLED_BY_DEFAULT. For more information, see TOKEN_GROUPS.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupSetInfo

Network Management Overview

Network Management Structures

TOKEN_GROUPS

GROUP_INFO_2 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_2** structure contains information about a global group, including name, identifier, and resource attributes.

It is recommended that you use the GROUP_INFO_3 structure instead.

Syntax

```
typedef struct _GROUP_INFO_2 {
  LPWSTR grpi2_name;
  LPWSTR grpi2_comment;
  DWORD  grpi2_group_id;
  DWORD  grpi2_attributes;
} GROUP_INFO_2, *PGROUP_INFO_2;
```

Members

`grpi2_name`

Pointer to a null-terminated Unicode character string that specifies the name of the global group. For more information, see the following Remarks section.

When you call the NetGroupSetInfo function this member is ignored.

`grpi2_comment`

Pointer to a null-terminated Unicode character string that contains a remark associated with the global group. This member can be a null string. The comment can contain MAXCOMMENTSZ characters.

`grpi2_group_id`

The relative identifier (RID) of the global group. The NetUserAdd and NetUserSetInfo functions ignore this member. For more information about RIDs, see SID Components.

`grpi2_attributes`

These attributes are hard-coded to SE_GROUP_MANDATORY, SE_GROUP_ENABLED, and SE_GROUP_ENABLED_BY_DEFAULT. For more information, see TOKEN_GROUPS.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupAdd

NetGroupEnum

NetGroupGetInfo

NetGroupSetInfo

Network Management Overview

Network Management Structures

TOKEN_GROUPS

GROUP_INFO_3 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_INFO_3** structure contains information about a global group, including name, security identifier (SID), and resource attributes.

Syntax

```
typedef struct _GROUP_INFO_3 {
  LPWSTR grpi3_name;
  LPWSTR grpi3_comment;
  PSID   grpi3_group_sid;
  DWORD  grpi3_attributes;
} GROUP_INFO_3, *PGROUP_INFO_3;
```

Members

`grpi3_name`

Pointer to a null-terminated Unicode character string that specifies the name of the global group.

When you call the NetGroupSetInfo function this member is ignored.

`grpi3_comment`

Pointer to a null-terminated Unicode character string that contains a remark associated with the global group. This member can be a null string. The comment can contain MAXCOMMENTSZ characters.

`grpi3_group_sid`

Pointer to a SID structure that contains the security identifier (SID) that uniquely identifies the global group. The NetUserAdd and NetUserSetInfo functions ignore this member.

`grpi3_attributes`

These attributes are hard-coded to SE_GROUP_MANDATORY, SE_GROUP_ENABLED, and SE_GROUP_ENABLED_BY_DEFAULT. For more information, see TOKEN_GROUPS.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Group Functions

NetGroupAdd

NetGroupEnum

NetGroupGetInfo

NetGroupSetInfo

Network Management Overview

Network Management Structures

SID

TOKEN_GROUPS

GROUP_USERS_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_USERS_INFO_0** structure contains global group member information.

Syntax

```
typedef struct _GROUP_USERS_INFO_0 {
  LPWSTR grui0_name;
} GROUP_USERS_INFO_0, *PGROUP_USERS_INFO_0, *LPGROUP_USERS_INFO_0;
```

Members

`grui0_name`

A pointer to a null-terminated Unicode character string that specifies a name. For more information, see the Remarks section.

Remarks

If you are calling the NetGroupGetUsers function or the NetGroupSetUsers function, the **grui0_name** member contains the name of a user that is a member of the specified group.

If you are calling the NetUserGetGroups function or the NetUserSetGroups function, the **grui0_name** member contains the name of a global group to which the specified user belongs.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

GROUP_USERS_INFO_1

Group Functions

NetGroupGetUsers

NetGroupSetUsers

NetUserGetGroups

NetUserGetInfo

NetUserSetGroups

Network Management Overview

Network Management Structures

GROUP_USERS_INFO_1 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **GROUP_USERS_INFO_1** structure contains global group member information.

Syntax

```
typedef struct _GROUP_USERS_INFO_1 {
  LPWSTR grui1_name;
  DWORD  grui1_attributes;
} GROUP_USERS_INFO_1, *PGROUP_USERS_INFO_1, *LPGROUP_USERS_INFO_1;
```

Members

`grui1_name`

Type: **LPWSTR**

A pointer to a null-terminated Unicode character string that specifies a name. For more information, see the Remarks section.

`grui1_attributes`

Type: **DWORD**

A set of attributes for this entry. This member can be a combination of the security group attributes defined in the *Winnt.h* header file.

VALUE	MEANING
SE_GROUP_MANDATORY 0x00000001	The group is mandatory.
SE_GROUP_ENABLED_BY_DEFAULT 0x00000002	The group is enabled for access checks by default.
SE_GROUP_ENABLED 0x00000004	The group is enabled for access checks.
SE_GROUP_OWNER 0x00000008	The group identifies a group account for which the user of the token is the owner of the group.
SE_GROUP_USE_FOR_DENY_ONLY 0x00000010	The group is used for deny only purposes. When this attribute is set, the SE_GROUP_ENABLED attribute must not be set.

SE_GROUP_INTEGRITY 0x00000020	The group is used for integrity. This attribute is available on Windows Vista and later.
SE_GROUP_INTEGRITY_ENABLED 0x00000040	The group is enabled for integrity level. This attribute is available on Windows Vista and later.
SE_GROUP_LOGON_ID 0xC0000000	The group is used to identify a logon session associated with an access token.
SE_GROUP_RESOURCE 0x20000000	The group identifies a domain-local group.

Remarks

If you are calling the NetGroupGetUsers function or the NetGroupSetUsers function, the **grui1_name** member contains the name of a user that is a member of the specified group.

If you are calling the NetUserGetGroups function or the NetUserSetGroups function, the **grui1_name** member contains the name of a global group to which the specified user belongs.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Windows Vista and later include an addition to the access control security mechanism of Windows that labels processes and other securable objects with an integrity level. Internet-facing programs are at higher risk for exploits than other programs because they download untrustworthy content from unknown sources. Running these programs with fewer permissions, or at a lower integrity level, than other programs reduces the ability of an exploit to modify the system or harm user data files. The SE_GROUP_INTEGRITY and SE_GROUP_INTEGRITY_ENABLED attributes of the **grui1_attributes** member are used for this purpose.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

GROUP_USERS_INFO_0

Group Functions

NetGroupGetUsers

NetGroupSetUsers

NetUserGetGroups

NetUserGetInfo

NetUserSetGroups

Network Management Overview

Network Management Structures

LOCALGROUP_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_INFO_0** structure contains a local group name.

Syntax

```
typedef struct _LOCALGROUP_INFO_0 {
  LPWSTR lgrpi0_name;
} LOCALGROUP_INFO_0, *PLOCALGROUP_INFO_0, *LPLOCALGROUP_INFO_0;
```

Members

`lgrpi0_name`

Pointer to a Unicode string that specifies a local group name. For more information, see the following Remarks section.

Remarks

When you call the NetLocalGroupAdd function, this member specifies the name of a new local group. When you call the NetLocalGroupSetInfo function, this member specifies the new name of an existing local group.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Local Group Functions

NetLocalGroupAdd

NetLocalGroupEnum

NetLocalGroupSetInfo

Network Management Overview

Network Management Structures

LOCALGROUP_INFO_1 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_INFO_1** structure contains a local group name and a comment describing the local group.

Syntax

```
typedef struct _LOCALGROUP_INFO_1 {
  LPWSTR lgrpi1_name;
  LPWSTR lgrpi1_comment;
} LOCALGROUP_INFO_1, *PLOCALGROUP_INFO_1, *LPLOCALGROUP_INFO_1;
```

Members

`lgrpi1_name`

Pointer to a Unicode string that specifies a local group name. For more information, see the following Remarks section.

This member is ignored when you call the NetLocalGroupSetInfo function.

`lgrpi1_comment`

Pointer to a Unicode string that contains a remark associated with the local group. This member can be a null string. The comment can have as many as MAXCOMMENTSZ characters.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Local Group Functions

NetLocalGroupAdd

NetLocalGroupEnum

NetLocalGroupGetInfo

NetLocalGroupSetInfo

Network Management Overview

Network Management Structures

LOCALGROUP_INFO_1002 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_INFO_1002** structure contains a comment describing a local group.

Syntax

```
typedef struct _LOCALGROUP_INFO_1002 {
  LPWSTR lgrpi1002_comment;
} LOCALGROUP_INFO_1002, *PLOCALGROUP_INFO_1002, *LPLOCALGROUP_INFO_1002;
```

Members

`lgrpi1002_comment`

Pointer to a Unicode string that specifies a remark associated with the local group. This member can be a null string. The comment can have as many as MAXCOMMENTSZ characters.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Local Group Functions

NetLocalGroupSetInfo

Network Management Overview

Network Management Structures

LOCALGROUP_MEMBERS_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_MEMBERS_INFO_0** structure contains the security identifier (SID) associated with a local group member. The member can be a user account or a global group account.

Syntax

```
typedef struct _LOCALGROUP_MEMBERS_INFO_0 {
  PSID lgrmi0_sid;
} LOCALGROUP_MEMBERS_INFO_0, *PLOCALGROUP_MEMBERS_INFO_0, *LPLOCALGROUP_MEMBERS_INFO_0;
```

Members

`lgrmi0_sid`

Pointer to a SID structure that contains the security identifier (SID) of the local group member.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

LOCALGROUP_MEMBERS_INFO_1

LOCALGROUP_MEMBERS_INFO_2

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupAddMembers

NetLocalGroupDelMembers

NetLocalGroupGetMembers

NetLocalGroupSetMembers

Network Management Overview

Network Management Structures

LOCALGROUP_MEMBERS_INFO_1 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_MEMBERS_INFO_1** structure contains the security identifier (SID) and account information associated with the member of a local group.

Syntax

```
typedef struct _LOCALGROUP_MEMBERS_INFO_1 {
  PSID         lgrmi1_sid;
  SID_NAME_USE lgrmi1_sidusage;
  LPWSTR       lgrmi1_name;
} LOCALGROUP_MEMBERS_INFO_1, *PLOCALGROUP_MEMBERS_INFO_1, *LPLOCALGROUP_MEMBERS_INFO_1;
```

Members

`lgrmi1_sid`

Type: **PSID**

A pointer to a SID structure that contains the security identifier (SID) of an account that is a member of this local group member. The account can be a user account or a global group account.

`lgrmi1_sidusage`

Type: **SID_NAME_USE**

The account type associated with the security identifier specified in the **lgrmi1_sid** member. The following values are valid.

VALUE	MEANING
SidTypeUser	The account is a user account.
SidTypeGroup	The account is a global group account.
SidTypeWellKnownGroup	The account is a well-known group account (such as Everyone). For more information, see Well-Known SIDs.
SidTypeDeletedAccount	The account has been deleted.
SidTypeUnknown	The account type cannot be determined.

`lgrmi1_name`

Type: **LPWSTR**

A pointer to the account name of the local group member identified by the **lgrmi1_sid** member. The **lgrmi1_name** member does not include the domain name. For more information, see the following Remarks section.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_2

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupGetMembers

Network Management Overview

Network Management Structures

LOCALGROUP_MEMBERS_INFO_2 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_MEMBERS_INFO_2** structure contains the security identifier (SID) and account information associated with a local group member.

Syntax

```
typedef struct _LOCALGROUP_MEMBERS_INFO_2 {
  PSID         lgrmi2_sid;
  SID_NAME_USE lgrmi2_sidusage;
  LPWSTR       lgrmi2_domainandname;
} LOCALGROUP_MEMBERS_INFO_2, *PLOCALGROUP_MEMBERS_INFO_2, *LPLOCALGROUP_MEMBERS_INFO_2;
```

Members

`lgrmi2_sid`

Type: **PSID**

A pointer to a SID structure that contains the security identifier (SID) of a local group member. The local group member can be a user account or a global group account.

`lgrmi2_sidusage`

Type: **SID_NAME_USE**

The account type associated with the security identifier specified in the **lgrmi2_sid** member. The following values are valid.

VALUE	MEANING
SidTypeUser	The account is a user account.
SidTypeGroup	The account is a global group account.
SidTypeWellKnownGroup	The account is a well-known group account (such as Everyone). For more information, see Well-Known SIDs.
SidTypeDeletedAccount	The account has been deleted.
SidTypeUnknown	The account type cannot be determined.

`lgrmi2_domainandname`

Type: **LPWSTR**

A pointer to the account name of the local group member identified by **lgrmi2_sid**. The **lgrmi2_domainandname** member includes the domain name and has the form:

```
<DomainName>\<AccountName>
```

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_1

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupGetMembers

Network Management Overview

Network Management Structures

LOCALGROUP_MEMBERS_INFO_3 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_MEMBERS_INFO_3** structure contains the account name and domain name associated with a local group member.

Syntax

```
typedef struct _LOCALGROUP_MEMBERS_INFO_3 {
  LPWSTR lgrmi3_domainandname;
} LOCALGROUP_MEMBERS_INFO_3, *PLOCALGROUP_MEMBERS_INFO_3, *LPLOCALGROUP_MEMBERS_INFO_3;
```

Members

`lgrmi3_domainandname`

Pointer to a null-terminated Unicode string specifying the account name of the local group member prefixed by the domain name and the "" separator character. For example:

```
<DomainName>\<AccountName>
```

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_1

LOCALGROUP_MEMBERS_INFO_2

Local Group Functions

NetLocalGroupAddMembers

NetLocalGroupDelMembers

NetLocalGroupGetMembers

NetLocalGroupSetMembers

Network Management Overview

Network Management Structures

LOCALGROUP_USERS_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **LOCALGROUP_USERS_INFO_0** structure contains local group member information.

Syntax

```
typedef struct _LOCALGROUP_USERS_INFO_0 {
  LPWSTR lgrui0_name;
} LOCALGROUP_USERS_INFO_0, *PLOCALGROUP_USERS_INFO_0, *LPLOCALGROUP_USERS_INFO_0;
```

Members

`lgrui0_name`

Pointer to a Unicode string specifying the name of a local group to which the user belongs.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Local Group Functions

NetUserGetLocalGroups

Network Management Overview

Network Management Structures

NET_DISPLAY_GROUP structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NET_DISPLAY_GROUP** structure contains information that an account manager can access to determine information about group accounts.

Syntax

```
typedef struct _NET_DISPLAY_GROUP {
  LPWSTR grpi3_name;
  LPWSTR grpi3_comment;
  DWORD  grpi3_group_id;
  DWORD  grpi3_attributes;
  DWORD  grpi3_next_index;
} NET_DISPLAY_GROUP, *PNET_DISPLAY_GROUP;
```

Members

`grpi3_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the group.

`grpi3_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the group. This string can be a null string, or it can have any number of characters before the terminating null character.

`grpi3_group_id`

Type: **DWORD**

The relative identifier (RID) of the group. The relative identifier is determined by the accounts database when the group is created. It uniquely identifies the group to the account manager within the domain. The NetUserAdd and NetUserSetInfo functions ignore this member. For more information about RIDs, see SID Components.

`grpi3_attributes`

Type: **DWORD**

These attributes are hard-coded to SE_GROUP_MANDATORY, SE_GROUP_ENABLED, and SE_GROUP_ENABLED_BY_DEFAULT. For more information, see TOKEN_GROUPS.

`grpi3_next_index`

Type: **DWORD**

The index of the last entry returned by the NetQueryDisplayInformation function. Pass this value as the *Index* parameter to **NetQueryDisplayInformation** to return the next logical entry. Note that you should not use the value of this member for any purpose except to retrieve more data with additional calls to **NetQueryDisplayInformation**.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Get Functions

NetQueryDisplayInformation

NetUserAdd

NetUserSetInfo

Network Management Overview

Network Management Structures

TOKEN_GROUPS

NET_DISPLAY_MACHINE structure (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NET_DISPLAY_MACHINE** structure contains information that an account manager can access to determine information about computers and their attributes.

Syntax

```
typedef struct _NET_DISPLAY_MACHINE {
  LPWSTR usri2_name;
  LPWSTR usri2_comment;
  DWORD  usri2_flags;
  DWORD  usri2_user_id;
  DWORD  usri2_next_index;
} NET_DISPLAY_MACHINE, *PNET_DISPLAY_MACHINE;
```

Members

`usri2_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the computer to access.

`usri2_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the computer. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri2_flags`

Type: **DWORD**

A set of flags that contains values that determine several features. This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_PASSWD_NOTREQD	No password is required.

UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out (blocked). For the NetUserSetInfo function, this value can be cleared to unlock a previously locked account. This value cannot be used to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	Represents the password, which will never expire on the account.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING

UF_NORMAL_ACCOUNT	A default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	An account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	A computer account for a workstation or a server that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	A computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	A permit to trust account for a domain that trusts other domains.

`usri2_user_id`

Type: **DWORD**

The relative identifier (RID) of the computer. The relative identifier is determined by the accounts database when the computer is defined. For more information about RIDS, see SID Components.

`usri2_next_index`

Type: **DWORD**

The index of the last entry returned by the NetQueryDisplayInformation function. Pass this value as the *Index* parameter to **NetQueryDisplayInformation** to return the next logical entry. Note that you should not use the value of this member for any purpose except to retrieve more data with additional calls to **NetQueryDisplayInformation**.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Get Functions

NetQueryDisplayInformation

NetUserAdd

NetUserSetInfo

Network Management Overview

Network Management Structures

NET_DISPLAY_USER structure (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NET_DISPLAY_USER** structure contains information that an account manager can access to determine information about user accounts.

Syntax

```
typedef struct _NET_DISPLAY_USER {
  LPWSTR usri1_name;
  LPWSTR usri1_comment;
  DWORD  usri1_flags;
  LPWSTR usri1_full_name;
  DWORD  usri1_user_id;
  DWORD  usri1_next_index;
} NET_DISPLAY_USER, *PNET_DISPLAY_USER;
```

Members

`usri1_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account.

`usri1_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the user. This string can be a null string, or it can have any number of characters before the terminating null character (MAXCOMMENTSZ).

`usri1_flags`

Type: **DWORD**

A set of user account flags. This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_PASSWD_NOTREQD	No password is required.

UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out (blocked). For the NetUserSetInfo function, this value can be cleared to unlock a previously locked account. This value cannot be used to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password will never expire on the account.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	The account is marked as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	The user is required to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	This principal is restricted to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.

UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a workstation or a server that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri1_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri1_user_id`

Type: **DWORD**

The relative identifier (RID) of the user. The relative identifier is determined by the accounts database when the user is created. It uniquely defines this user account to the account manager within the domain. For more information about relative identifiers, see SID Components.

`usri1_next_index`

Type: **DWORD**

The index of the last entry returned by the NetQueryDisplayInformation function. Pass this value as the *Index* parameter to **NetQueryDisplayInformation** to return the next logical entry. Note that you should not use the value of this member for any purpose except to retrieve more data with additional calls to **NetQueryDisplayInformation**.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

Get Functions

NetQueryDisplayInformation

NetUserAdd

NetUserSetInfo

Network Management Overview

Network Management Structures

NET_VALIDATE_AUTHENTICATION_INPUT_ARG structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

A client application passes the **NET_VALIDATE_AUTHENTICATION_INPUT_ARG** structure to the NetValidatePasswordPolicy function when the application requests an authentication validation.

Syntax

```
typedef struct _NET_VALIDATE_AUTHENTICATION_INPUT_ARG {
  NET_VALIDATE_PERSISTED_FIELDS InputPersistedFields;
  BOOLEAN                       PasswordMatched;
} NET_VALIDATE_AUTHENTICATION_INPUT_ARG, *PNET_VALIDATE_AUTHENTICATION_INPUT_ARG;
```

Members

`InputPersistedFields`

Specifies a NET_VALIDATE_PERSISTED_FIELDS structure that contains persistent password-related information about the account being logged on.

`PasswordMatched`

BOOLEAN value that indicates the result of the client application's authentication of the password supplied by the user. If this parameter is **FALSE**, the password has not been authenticated.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NET_VALIDATE_OUTPUT_ARG structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NET_VALIDATE_OUTPUT_ARG** structure contains information about persistent password-related data that has changed since the user's last logon as well as the result of the function's password validation check.

Syntax

```
typedef struct _NET_VALIDATE_OUTPUT_ARG {
  NET_VALIDATE_PERSISTED_FIELDS ChangedPersistedFields;
  NET_API_STATUS                ValidationStatus;
} NET_VALIDATE_OUTPUT_ARG, *PNET_VALIDATE_OUTPUT_ARG;
```

Members

`ChangedPersistedFields`

A structure that contains changes to persistent information about the account being logged on. For more information, see the following Remarks section.

`ValidationStatus`

The result of the password validation check performed by the NetValidatePasswordPolicy function. The status depends on the value specified in the *ValidationType* parameter to that function.

Authentication. When you call NetValidatePasswordPolicy and specify the *ValidationType* parameter as NetValidateAuthentication, this member can be one of the following values.

VALUE	MEANING
NERR_AccountLockedOut	Validation failed. The account is locked out.
NERR_PasswordMustChange	Validation failed. The password must change at the next logon.
NERR_PasswordExpired	Validation failed. The password has expired.
NERR_BadPassword	Validation failed. The password is invalid.
NERR_Success	The password passes the validation check.

Password change. When you call NetValidatePasswordPolicy and specify the *ValidationType* parameter as NetValidatePasswordChange, this member can be one of the following values.

VALUE	MEANING
NERR_AccountLockedOut	Validation failed. The account is locked out.

NERR_PasswordTooRecent	Validation failed. The password for the user is too recent to change.
NERR_BadPassword	Validation failed. The password is invalid.
NERR_PasswordHistConflict	Validation failed. The password cannot be used at this time.
NERR_PasswordTooShort	Validation failed. The password does not meet policy requirements because it is too short.
NERR_PasswordTooLong	Validation failed. The password does not meet policy requirements because it is too long.
NERR_PasswordNotComplexEnough	Validation failed. The password does not meet policy requirements because it is not complex enough.
NERR_PasswordFilterError	Validation failed. The password does not meet the requirements of the password filter DLL.
NERR_Success	The password passes the validation check.

Password reset. When you call **NetValidatePasswordPolicy** and specify the *ValidationType* parameter as NetValidatePasswordReset, this member can be one of the following values.

VALUE	MEANING
NERR_PasswordTooShort	Validation failed. The password does not meet policy requirements because it is too short.
NERR_PasswordTooLong	Validation failed. The password does not meet policy requirements because it is too long.
NERR_PasswordNotComplexEnough	Validation failed. The password does not meet policy requirements because it is not complex enough.
NERR_PasswordFilterError	Validation failed. The password does not meet the requirements of the password filter DLL.
NERR_Success	The password passes the validation check.

Remarks

The NetValidatePasswordPolicy function outputs the **NET_VALIDATE_OUTPUT_ARG** structure.

Note that it is the application's responsibility to save all the data in the **ChangedPersistedFields** member of the **NET_VALIDATE_OUTPUT_ARG** structure as well as any User object information. The next time the application calls NetValidatePasswordPolicy on the same instance of the User object, the application must provide the required fields from the persistent information.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

A client application passes the **NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG** structure to the NetValidatePasswordPolicy function when the application requests a password change validation.

Syntax

```
typedef struct _NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG {
  NET_VALIDATE_PERSISTED_FIELDS InputPersistedFields;
  LPWSTR                        ClearPassword;
  LPWSTR                        UserAccountName;
  NET_VALIDATE_PASSWORD_HASH    HashedPassword;
  BOOLEAN                       PasswordMatch;
} NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG, *PNET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG;
```

Members

`InputPersistedFields`

Specifies a NET_VALIDATE_PERSISTED_FIELDS structure that contains persistent password-related information about the account being logged on.

`ClearPassword`

Pointer to a Unicode string specifying the new password, in plaintext format.

`UserAccountName`

Pointer to a Unicode string specifying the name of the user account.

`HashedPassword`

Specifies a NET_VALIDATE_PASSWORD_HASH structure that contains a hash of the new password.

`PasswordMatch`

BOOLEAN value that indicates the result of the application's attempt to validate the old password supplied by the user. If this parameter is **FALSE**, the password was not validated.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NET_VALIDATE_PASSWORD_HASH structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NET_VALIDATE_PASSWORD_HASH** structure contains a password hash.

Syntax

```
typedef struct _NET_VALIDATE_PASSWORD_HASH {
  ULONG  Length;
  LPBYTE Hash;
} NET_VALIDATE_PASSWORD_HASH, *PNET_VALIDATE_PASSWORD_HASH;
```

Members

`Length`

Specifies the length of this structure.

`Hash`

Password hash.

Remarks

The NET_VALIDATE_PASSWORD_RESET_INPUT_ARG and NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG structures contain a **NET_VALIDATE_PASSWORD_HASH** structure. The NET_VALIDATE_PERSISTED_FIELDS structure contains a pointer to this structure.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NET_VALIDATE_PASSWORD_RESET_INPUT_ARG structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

A client application passes the **NET_VALIDATE_PASSWORD_RESET_INPUT_ARG** structure to the NetValidatePasswordPolicy function when the application requests a password reset validation.

Syntax

```
typedef struct _NET_VALIDATE_PASSWORD_RESET_INPUT_ARG {
  NET_VALIDATE_PERSISTED_FIELDS InputPersistedFields;
  LPWSTR                        ClearPassword;
  LPWSTR                        UserAccountName;
  NET_VALIDATE_PASSWORD_HASH    HashedPassword;
  BOOLEAN                       PasswordMustChangeAtNextLogon;
  BOOLEAN                       ClearLockout;
} NET_VALIDATE_PASSWORD_RESET_INPUT_ARG, *PNET_VALIDATE_PASSWORD_RESET_INPUT_ARG;
```

Members

`InputPersistedFields`

Specifies a NET_VALIDATE_PERSISTED_FIELDS structure that contains persistent password-related information about the account being logged on.

`ClearPassword`

Pointer to a Unicode string specifying the new password, in plaintext format.

`UserAccountName`

Pointer to a Unicode string specifying the name of the user account.

`HashedPassword`

Specifies a NET_VALIDATE_PASSWORD_HASH structure that contains a hash of the new password.

`PasswordMustChangeAtNextLogon`

BOOLEAN value that indicates whether the user must change his or her password at the next logon. If this parameter is **TRUE**, the user must change the password at the next logon.

`ClearLockout`

BOOLEAN value that can reset the "lockout state" of the user account. If this member is **TRUE**, the account will no longer be locked out. Note that an application cannot directly lock out an account. An account can be locked out only as a result of exceeding the maximum number of invalid password authentications allowed for the account.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NET_VALIDATE_PERSISTED_FIELDS structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NET_VALIDATE_PERSISTED_FIELDS** structure contains information about a user's password properties. Input to and output from the NetValidatePasswordPolicy function contain persistent password-related data. When the function outputs this structure, it identifies the persistent data that has changed in this call.

Syntax

```
typedef struct _NET_VALIDATE_PERSISTED_FIELDS {
  ULONG                       PresentFields;
  FILETIME                    PasswordLastSet;
  FILETIME                    BadPasswordTime;
  FILETIME                    LockoutTime;
  ULONG                       BadPasswordCount;
  ULONG                       PasswordHistoryLength;
  PNET_VALIDATE_PASSWORD_HASH PasswordHistory;
} NET_VALIDATE_PERSISTED_FIELDS, *PNET_VALIDATE_PERSISTED_FIELDS;
```

Members

`PresentFields`

Type: **ULONG**

A set of bit flags identifying the persistent password-related data that has changed. This member is valid only when this structure is output from the **NetValidatePasswordPolicy** function. This member is ignored when this structure is input to the function. For more information, see the following Remarks section.

VALUE	MEANING
NET_VALIDATE_PASSWORD_LAST_SET	The **PasswordLastSet** member contains a new value.
NET_VALIDATE_BAD_PASSWORD_TIME	The **BadPasswordTime** member contains a new value.
NET_VALIDATE_LOCKOUT_TIME	The **LockoutTime** member contains a new value.
NET_VALIDATE_BAD_PASSWORD_COUNT	The **BadPasswordCount** member contains a new value.
NET_VALIDATE_PASSWORD_HISTORY_LENGTH	The **PasswordHistoryLength** member contains a new value.
NET_VALIDATE_PASSWORD_HISTORY	The **PasswordHistory** member contains a new value.

`PasswordLastSet`

Type: **FILETIME**

The date and time (in GMT) when the password for the account was set or last changed.

`BadPasswordTime`

Type: **FILETIME**

The date and time (in GMT) when the user tried to log on to the account using an incorrect password.

`LockoutTime`

Type: **FILETIME**

The date and time (in GMT) when the account was last locked out. If the account has not been locked out, this member is zero. A lockout occurs when the number of bad password logins exceeds the number allowed.

`BadPasswordCount`

Type: **ULONG**

The number of times the user tried to log on to the account using an incorrect password.

`PasswordHistoryLength`

Type: **ULONG**

The number of previous passwords saved in the history list for the account. The user cannot reuse a password in the history list.

`PasswordHistory`

Type: **PNET_VALIDATE_PASSWORD_HASH**

A pointer to a NET_VALIDATE_PASSWORD_HASH structure that contains the password hashes in the history list.

Remarks

Note that it is the application's responsibility to save all changed persistent data as well as any user object information. The next time the application calls NetValidatePasswordPolicy on the same instance of the user object, the application must provide the required fields from the persistent information.

The NET_VALIDATE_AUTHENTICATION_INPUT_ARG, NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG, NET_VALIDATE_PASSWORD_RESET_INPUT_ARG, and NET_VALIDATE_OUTPUT_ARG structures contain a **NET_VALIDATE_PERSISTED_FIELDS** structure.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetValidatePasswordPolicy

Network Management Overview

Network Management Structures

NetAccessAdd function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessAdd** function creates a new access control list (ACL) for a resource.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be the following value.

VALUE	MEANING
1	The *pbBuffer* parameter points to an **access_info_1** structure.

`buf`

Pointer to the buffer that contains the access information structure.

`parm_err`

Specifies the size, in bytes, of the buffer pointed to by the *pbBuffer* parameter.

Return value

If the function succeeds, the return value is **NERR_Success**.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires User level security to be enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Authorization Functions

NetAccessDel function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessDel** function deletes the access control list (ACL) for a resource.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessDel(
  LPCWSTR servername,
  LPCWSTR resource
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`resource`

Pointer to a string that contains the name of the network resource for which to remove the access control list.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires Admin privilege to successfully execute on a computer that has local security enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib

DLL	Netapi32.dll

See also

Authorization Functions

NetAccessEnum function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessEnum** function retrieves information about each access permission record.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessEnum(
  LPCWSTR servername,
  LPCWSTR BasePath,
  DWORD   Recursive,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`BasePath`

Pointer to a string that contains a base pathname for the resource. A **NULL** pointer or **NULL** string means no base path is to be used. The path can be specified as a universal naming convention (UNC) pathname.

`Recursive`

Specifies a flag that enables or disables recursive searching.

If this parameter is equal to zero, the **NetAccessEnum** function returns entries for the resource named as the base path by the *pszBasePath* parameter, and for the resources directly below that base path.

If this parameter is nonzero, the function returns entries for all access control lists (ACLs) that have *pszBasePath* at the beginning of the resource name.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	The *pbBuffer* parameter points to an **access_info_0** structure.

1	The *pbBuffer* parameter points to an **access_info_1** structure.

`bufptr`

Pointer to the buffer that receives the access information structure. The format of this data depends on the value of the *sLevel* parameter.

`prefmaxlen`

Specifies the size, in bytes, of the buffer pointed to by the *pbBuffer* parameter.

`entriesread`

Pointer to an unsigned short integer that receives the count of elements actually enumerated. The count is valid only if the **NetAccessEnum** function returns **NERR_Success** or **ERROR_MORE_DATA**.

`totalentries`

Pointer to an unsigned short integer that receives the total number of entries that could have been enumerated. The count is valid only if the **NetAccessEnum** function returns **NERR_Success** or **ERROR_MORE_DATA**.

`resume_handle`

TBD

Return value

If the function succeeds, the return value is **NERR_Success**.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires Admin privilege to successfully execute on a computer that has local security enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Authorization functions

NetAccessGetInfo function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessGetInfo** function retrieves the access control list (ACL) for a specified resource.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessGetInfo(
  LPCWSTR servername,
  LPCWSTR resource,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`resource`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	The p parameter points to an **access_info_0** structure.
1	The *pbBuffer* parameter points to an **access_info_1** structure.

`level`

Pointer to the buffer that receives the access information structure. The format of this data depends on the value of the *sLevel* parameter.

`bufptr`

Specifies the size, in bytes, of the buffer pointed to by the *pbBuffer* parameter.

Return value

If the function succeeds, the return value is **NERR_Success**.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires Admin privilege to successfully execute on a computer that has local security enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Authorization Functions

NetAccessGetUserPerms function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessGetUserPerms** function returns a specified user's or group's access permissions for a particular resource.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessGetUserPerms(
  LPCWSTR servername,
  LPCWSTR UGname,
  LPCWSTR resource,
  LPDWORD Perms
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`UGname`

Pointer to a string that specifies the name of the user or group to query.

`resource`

Pointer to a string that contains the name of the network resource to query.

`Perms`

Pointer to an unsigned short integer that receives the user permissions for the specified resource.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires Admin privilege to successfully execute on a computer that has local security enabled. When users request their own access permissions, no special privilege is required.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Authorization Functions

NetAccessSetInfo function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. For a list of alternate functions, see Authorization Functions.]

Not supported.

The **NetAccessSetInfo** function changes the access control list (ACL) for a resource.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAccessSetInfo(
  LPCWSTR servername,
  LPCWSTR resource,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`resource`

Pointer to a string that contains the name of the network resource to modify.

`level`

Specifies the information level of the data. This parameter can be the following value.

VALUE	MEANING
1	The *pbBuffer* parameter points to an **access_info_1** structure.

`buf`

Pointer to the buffer that contains the access information structure. The format of this data depends on the value of the *sLevel* parameter.

`parm_err`

TBD

Return value

If the function succeeds, the return value is **NERR_Success**.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

This function requires Admin privilege to successfully execute on a computer that has local security enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h, Lmaccess.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Authorization functions

NetGetAnyDCName function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGetAnyDCName** function returns the name of any domain controller (DC) for a domain that is directly trusted by the specified server.

Applications that support DNS-style names should call the DsGetDcName function. This function can locate any DC in any domain, whether or not the domain is directly trusted by the specified server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGetAnyDCName(
  LPCWSTR ServerName,
  LPCWSTR DomainName,
  LPBYTE  *Buffer
);
```

Parameters

`ServerName`

`DomainName`

`Buffer`

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_NO_LOGON_SERVERS	No domain controllers could be found.
ERROR_NO_SUCH_DOMAIN	The specified domain is not a trusted domain.
ERROR_NO_TRUST_LSA_SECRET	The client side of the trust relationship is broken.
ERROR_NO_TRUST_SAM_ACCOUNT	The server side of the trust relationship is broken or the password is broken.
ERROR_DOMAIN_TRUST_INCONSISTENT	The server that responded is not a proper domain controller of the specified domain.

Remarks

No special group membership is required to successfully execute the **NetGetAnyDCName** function.

If *servername* specifies a stand-alone workstation or a stand-alone server, no *domainname* is valid.

If *servername* specifies a workstation that is a member of a domain, or a server that is a member of a domain, the *domainname* must be in the same domain as *servername*.

If *servername* specifies a domain controller, the *domainname* must be one of the domains trusted by the domain for which the server is a controller. The domain controller that this call finds has been operational at least once during this call.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

DsGetDcName

Get Functions

NetGetDCName

Network Management Functions

Network Management Overview

NetGetDCName function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGetDCName** function returns the name of the primary domain controller (PDC). It does not return the name of the backup domain controller (BDC) for the specified domain. Also, you cannot remote this function to a non-PDC server.

Applications that support DNS-style names should call the DsGetDcName function. Domain controllers in this type of environment have a multi-master directory replication relationship. Therefore, it may be advantageous for your application to use a DC that is not the PDC. You can call the **DsGetDcName** function to locate any DC in the domain; **NetGetDCName** returns only the name of the PDC.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGetDCName(
  LPCWSTR ServerName,
  LPCWSTR DomainName,
  LPBYTE  *Buffer
);
```

Parameters

`ServerName`

`DomainName`

`Buffer`

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
NERR_DCNotFound	Could not find the domain controller for the domain specified in the *domainname* parameter.
ERROR_BAD_NETPATH	The network path was not found. This error is returned if the computer specified in the *servername* parameter could not be found.
ERROR_INVALID_NAME	The name syntax is incorrect. This error is returned if the name specified in the *servername* parameter contains illegal characters.
ERROR_NOT_SUPPORTED	The request is not supported.

Remarks

No special group membership is required to successfully execute the **NetGetDCName** function.

Examples

The following code sample demonstrates how to retrieve the primary domain controller using the **NetGetDCName** function. The sample calls **NetGetDCName** specifying the servername and domainname parameters. If the call succeeds, the code prints information out the name of the primary domain controller. Finally, the sample frees the memory allocated for the buffer where the domain controller name was returned.

```
#ifndef UNICODE
#define UNICODE
#endif

#include <stdio.h>
#include <stdlib.h>  // for _wtoi function
#include <assert.h>
#include <windows.h>
#include <lm.h>

// Need to link with netapi32.lib
#pragma comment(lib, "netapi32.lib")

int wmain(int argc, wchar_t * argv[])
{

    NET_API_STATUS nStatus;

    LPCWSTR lpServer = NULL;
    LPCWSTR lpDomain = NULL;

    LPCWSTR lpDcName = NULL;

    if (argc != 3 ) {
        wprintf(L"Usage: %ws <ServerName> <DomainName>\n",
                argv[0]);
        wprintf(L"     %ws Myserver Domain\n", argv[0]);
        exit(1);
    }

    lpServer = argv[1];
    lpDomain = argv[2];

    wprintf(L"Calling NetGetDCName with parameters\n");
    wprintf(L"    lpServer = %ws\n", lpServer);
    wprintf(L"    lpDomain = %ws\n", lpDomain);

    //
    // Call the NetGetDCName function
    //
    nStatus = NetGetDCName(lpServer, lpDomain, (LPBYTE *) &lpDcName);
    //
    // If the call succeeds,
    //
    if (nStatus == NERR_Success) {
        wprintf(L"NetGetDCName was successful\n", nStatus);
        wprintf(L"DC Name = %ws\n", lpDcName);
        // Need to free the returned buffer
        nStatus = NetApiBufferFree( (LPVOID) lpDcName);
        if (nStatus != NERR_Success)
            wprintf(L"NetApiBufferFree failed with error: %lu (0x%lx)\n",
                nStatus, nStatus);
    } else {
        wprintf(L"NetGetDCName failed with error: %lu (0x%lx)\n", nStatus,
                nStatus);
        wprintf(L"   Error = ");
```

```
        switch (nStatus) {
        case ERROR_INVALID_PARAMETER:
            wprintf(L"ERROR_INVALID_PARAMETER\n");
            break;
        case ERROR_NO_SUCH_DOMAIN:
            wprintf(L"ERROR_NO_SUCH_DOMAIN\n");
            break;
        case ERROR_NOT_SUPPORTED:
            wprintf(L"ERROR_NOT_SUPPORTED\n");
            break;
        case ERROR_BAD_NETPATH:
            wprintf(L"ERROR_BAD_NETPATH\n");
            break;
        case ERROR_INVALID_COMPUTERNAME:
            wprintf(L"ERROR_INVALID_COMPUTERNAME\n");
            break;
        case DNS_ERROR_INVALID_NAME_CHAR:
            wprintf(L"DNS_ERROR_INVALID_NAME_CHAR\n");
            break;
        case DNS_ERROR_NON_RFC_NAME:
            wprintf(L"DNS_ERROR_NON_RFC_NAME\n");
            break;
        case ERROR_INVALID_NAME:
            wprintf(L"ERROR_INVALID_NAME\n");
            break;
        case NERR_DCNotFound:
            wprintf(L"NERR_DCNotFound\n");
            break;
        case NERR_WkstaNotStarted:
            wprintf(L"NERR_WkstaNotStarted\n");
            break;
        case RPC_S_SERVER_UNAVAILABLE:
            wprintf(L"RPC_S_SERVER_UNAVAILABLE\n");
            break;
        case RPC_E_REMOTE_DISABLED:
            wprintf(L"RPC_E_REMOTE_DISABLED\n");
            break;
        default:
            wprintf(L"Other error, see Winerror.h or lmerr.h)\n");
            break;
        }
    }

    return nStatus;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

DsGetDcName

Get Functions

NetGetAnyDCName

Network Management Functions

Network Management Overview

NetGetDisplayInformationIndex function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGetDisplayInformationIndex** function returns the index of the first display information entry whose name begins with a specified string or whose name alphabetically follows the string. You can use this function to determine a starting index for subsequent calls to the NetQueryDisplayInformation function.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGetDisplayInformationIndex(
  LPCWSTR ServerName,
  DWORD   Level,
  LPCWSTR Prefix,
  LPDWORD Index
);
```

Parameters

`ServerName`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`Level`

Specifies the level of accounts to query. This parameter can be one of the following values.

VALUE	MEANING
1	Query all local and global (normal) user accounts.
2	Query all workstation and server user accounts.
3	Query all global groups.

`Prefix`

Pointer to a string that specifies the prefix for which to search.

`Index`

Pointer to a value that receives the index of the requested entry.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The value specified for the *Level* parameter is invalid.
ERROR_NO_MORE_ITEMS	There were no more items on which to operate.
NERR_InvalidComputer	The computer name is invalid.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The function only returns information to which the caller has Read access. The caller must have List Contents access to the Domain object, and Enumerate Entire SAM Domain access on the SAM Server object located in the System container.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Get Functions

NetQueryDisplayInformation

Network Management Functions

NetGroupAdd function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetGroupAdd** function creates a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies a global group name. The *buf* parameter contains a pointer to a GROUP_INFO_0 structure.
1	Specifies a global group name and a comment. The *buf* parameter contains a pointer to a GROUP_INFO_1 structure.
2	Specifies detailed information about the global group. The *buf* parameter contains a pointer to a GROUP_INFO_2 structure. Note that on Windows XP and later, it is recommended that you use GROUP_INFO_3 instead.
3	Specifies detailed information about the global group. The *buf* parameter contains a pointer to a GROUP_INFO_3 structure. **Windows 2000:** This level is not supported.

`buf`

Pointer to a buffer that contains the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first member of the global group information structure in error

when ERROR_INVALID_PARAMETER is returned. If this parameter is **NULL**, the index is not returned on error. For more information, see the NetGroupSetInfo function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_GroupExists	The global group already exists.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the user container is used to perform the access check for this function. The caller must be able to create child objects of the group class. Typically, callers must also have write access to the entire object for calls to this function to succeed.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_INFO_0

GROUP_INFO_1

GROUP_INFO_3

Group Functions

NetGroupAddUser

NetGroupDel

NetGroupDelUser

NetGroupSetInfo

Network Management Functions

Network Management Overview

NetGroupAddUser function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGroupAddUser** function gives an existing user account membership in an existing global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupAddUser(
  LPCWSTR servername,
  LPCWSTR GroupName,
  LPCWSTR username
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`GroupName`

Pointer to a constant string that specifies the name of the global group in which the user is to be given membership. For more information, see the following Remarks section.

`username`

Pointer to a constant string that specifies the name of the user to be given membership in the global group. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.

NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.
NERR_UserNotFound	The user name could not be found.
NERR_GroupNotFound	The global group name could not be found.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Group Functions

NetGroupAdd

NetGroupDel

NetGroupDelUser

Network Management Functions

Network Management Overview

NetGroupDel function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGroupDel** function deletes a global group from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupDel(
  LPCWSTR servername,
  LPCWSTR groupname
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the global group account to delete. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.
NERR_GroupNotFound	The global group name could not be found.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Group Functions

NetGroupAdd

NetGroupAddUser

NetGroupDelUser

Network Management Functions

Network Management Overview

NetGroupDelUser function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGroupDelUser** function removes a user from a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupDelUser(
  LPCWSTR servername,
  LPCWSTR GroupName,
  LPCWSTR Username
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`GroupName`

Pointer to a constant string that specifies the name of the global group from which the user's membership should be removed. For more information, see the following Remarks section.

`Username`

Pointer to a constant string that specifies the name of the user to remove from the global group. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.

NERR_UserNotFound	The user name could not be found.
NERR_GroupNotFound	The global group name could not be found.
NERR_UserNotInGroup	The user does not belong to this global group.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Group Functions

NetGroupAdd

NetGroupAddUser

NetGroupDel

Network Management Functions

Network Management Overview

NetGroupEnum function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetGroupEnum** function retrieves information about each global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

The NetQueryDisplayInformation function provides an efficient mechanism for enumerating global groups. When possible, it is recommended that you use **NetQueryDisplayInformation** instead of the **NetGroupEnum** function.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupEnum(
  LPCWSTR    servername,
  DWORD      level,
  LPBYTE     *bufptr,
  DWORD      prefmaxlen,
  LPDWORD    entriesread,
  LPDWORD    totalentries,
  PDWORD_PTR resume_handle
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the global group name. The *bufptr* parameter points to an array of GROUP_INFO_0 structures.
1	Return the global group name and a comment. The *bufptr* parameter points to an array of GROUP_INFO_1 structures.
2	Return detailed information about the global group. The *bufptr* parameter points to an array of GROUP_INFO_2 structures. Note that on Windows XP and later, it is recommended that you use GROUP_INFO_3 instead.
3	Return detailed information about the global group. The *bufptr* parameter points to an array of GROUP_INFO_3 structures. **Windows 2000:** This level is not supported.

`bufptr`

Pointer to the buffer to receive the global group information structure. The format of this data depends on the value of the *level* parameter.

The system allocates the memory for this buffer. You must call the NetApiBufferFree function to deallocate the memory. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of the returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required to hold the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. The total number of entries is only a hint. For more information about determining the exact number of entries, see the following Remarks section.

`resume_handle`

Pointer to a variable that contains a resume handle that is used to continue the global group enumeration. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information

about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The function only returns information to which the caller has Read access. The caller must have List Contents access to the Domain object, and Enumerate Entire SAM Domain access on the SAM Server object located in the System container.

To determine the exact total number of groups, you must enumerate the entire tree, which can be a costly operation. To enumerate the entire tree, use the *resume_handle* parameter to continue the enumeration for consecutive calls, and use the *entriesread* parameter to accumulate the total number of groups. If your application is communicating with a domain controller, you should consider using the ADSI LDAP Provider to retrieve this type of data more efficiently. The ADSI LDAP Provider implements a set of ADSI objects that support various ADSI interfaces. For more information, see ADSI Service Providers.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_INFO_0

GROUP_INFO_1

GROUP_INFO_3

Group Functions

NetApiBufferFree

NetGroupGetInfo

NetGroupGetUsers

NetQueryDisplayInformation

Network Management Functions

Network Management Overview

NetGroupGetInfo function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGroupGetInfo** function retrieves information about a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupGetInfo(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the global group for which to retrieve information. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the global group name. The *bufptr* parameter points to a GROUP_INFO_0 structure.
1	Return the global group name and a comment. The *bufptr* parameter points to a GROUP_INFO_1 structure.
2	Return detailed information about the global group. The *bufptr* parameter points to a GROUP_INFO_2 structure. Note that on Windows XP and later, it is recommended that you use GROUP_INFO_3 instead.
3	Return detailed information about the global group. The *bufptr* parameter points to a GROUP_INFO_3 structure. **Windows 2000:** This level is not supported.

`bufptr`

Pointer to the address of the buffer that receives the global group information structure. The format of this data

depends on the value of the *level* parameter. The system allocates the memory for this buffer. You must call the NetApiBufferFree function to deallocate the memory. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_GroupNotFound	The global group name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)

Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_INFO_0

GROUP_INFO_1

GROUP_INFO_3

Group Functions

NetApiBufferFree

NetGroupSetInfo

Network Management Functions

Network Management Overview

NetGroupGetUsers function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetGroupGetUsers** function retrieves a list of the members in a particular global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupGetUsers(
  LPCWSTR    servername,
  LPCWSTR    groupname,
  DWORD      level,
  LPBYTE     *bufptr,
  DWORD      prefmaxlen,
  LPDWORD    entriesread,
  LPDWORD    totalentries,
  PDWORD_PTR ResumeHandle
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

A pointer to a constant string that specifies the name of the global group whose members are to be listed. For more information, see the following Remarks section.

`level`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
0	Return the global group's member names. The *bufptr* parameter points to an array of GROUP_USERS_INFO_0 structures.
1	Return the global group's member names and attributes. The *bufptr* parameter points to an array of GROUP_USERS_INFO_1 structures.

`bufptr`

A pointer to the address of the buffer that receives the information structure. The system allocates the memory for this buffer. You must call the NetApiBufferFree function to deallocate the memory. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

The preferred maximum length of the returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required to hold the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position.

`ResumeHandle`

A pointer to a variable that contains a resume handle that is used to continue an existing user enumeration. The handle should be zero on the first call and left unchanged for subsequent calls. If *ResumeHandle* parameter is **NULL**, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter was specified as a value other than 0 or 1.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to complete the operation.
NERR_InvalidComputer	The computer name is invalid.
NERR_GroupNotFound	The global group name in the structure pointed to by *bufptr* parameter could not be found.
NERR_InternalError	An internal error occurred.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users

and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

To grant one user membership in an existing global group, you can call the NetGroupAddUser function. To remove a user from a global group, call the NetGroupDelUser function. For information about replacing the membership of a global group, see NetGroupSetUsers.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_USERS_INFO_0

GROUP_USERS_INFO_1

Group Functions

NetApiBufferFree

NetGroupAddUser

NetGroupDelUser

NetGroupSetUsers

NetQueryDisplayInformation

NetUserGetGroups

Network Management Functions

NetGroupSetInfo function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetGroupSetInfo** function sets the parameters of a global group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupSetInfo(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the global group for which to set information. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies a global group name. The *buf* parameter points to a GROUP_INFO_0 structure.
1	Specifies a global group name and a comment. The *buf* parameter points to a GROUP_INFO_1 structure.
2	Specifies detailed information about the global group. The *buf* parameter points to a GROUP_INFO_2 structure. Note that on Windows XP and later, it is recommended that you use GROUP_INFO_3 instead.
3	Specifies detailed information about the global group. The *buf* parameter points to a GROUP_INFO_3 structure. **Windows 2000:** This level is not supported.
1002	Specifies a comment only about the global group. The *buf* parameter points to a GROUP_INFO_1002 structure.

1005	Specifies global group attributes. The *buf* parameter points to a GROUP_INFO_1005 structure.

For more information, see the following Remarks section.

`buf`

Pointer to a buffer that contains the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first member of the group information structure in error following an ERROR_INVALID_PARAMETER error code. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid. For more information, see the following Remarks section.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_GroupNotFound	The global group name could not be found.
NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management

Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function. Typically, callers must have write access to the entire object for calls to this function to succeed.

The correct way to set the new name of a global group is to call the **NetGroupSetInfo** function, using a GROUP_INFO_0 structure. Specify the new value in the **grpi0_name** member. If you use a GROUP_INFO_1 structure and specify the value in the **grpi1_name** member, the new name value is ignored.

If the **NetGroupSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the first member of the group information structure that is invalid. (A group information structure begins with GROUP_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix grpi*_ indicates that the member can begin with multiple prefixes, for example, grpi1_ or grpi2_.)

VALUE	MEMBER
GROUP_NAME_PARMNUM	grpi*_name
GROUP_COMMENT_PARMNUM	grpi*_comment
GROUP_ATTRIBUTES_PARMNUM	grpi*_attributes

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_INFO_0

GROUP_INFO_1

GROUP_INFO_1002

GROUP_INFO_1005

GROUP_INFO_3

Group Functions

NetGroupGetInfo

Network Management Functions

Network Management Overview

NetGroupSetUsers function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetGroupSetUsers** function sets the membership for the specified global group. Each user you specify is enrolled as a member of the global group. Users you do not specify, but who are currently members of the global group, will have their membership revoked.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGroupSetUsers(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  DWORD   totalentries
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

A pointer to a constant string that specifies the name of the global group of interest. For more information, see the Remarks section.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	The *buf* parameter points to an array of GROUP_USERS_INFO_0 structures that specify user names.
1	The *buf* parameter points to an array of GROUP_USERS_INFO_1 structures that specifies user names and the attributes of the group.

`buf`

A pointer to the buffer that contains the data. For more information, see Network Management Function Buffers.

`totalentries`

The number of entries in the buffer pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter was specified as a value other than 0 or 1.
ERROR_INVALID_PARAMETER	A parameter passed was not valid. This error is returned if the *totalentries* parameter was not valid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to complete the operation.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_GroupNotFound	The global group name could not be found.
NERR_InternalError	An internal error occurred.
NERR_SpeGroupOp	The operation is not allowed on certain special groups. These groups include user groups, admin groups, local groups, and guest groups.
NERR_UserNotFound	The user name could not be found.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Group object is used to perform the access check for this function.

You can replace the global group membership with an entirely new list of members by calling the **NetGroupSetUsers** function. The typical sequence of steps to perform this follows.

To replace the global group membership

1. Call the NetGroupGetUsers function to retrieve the current membership list.
2. Modify the returned membership list to reflect the new membership.
3. Call the **NetGroupSetUsers** function to replace the old membership list with the new membership list.

To grant one user membership in an existing global group, you can call the NetGroupAddUser function. To remove a user from a global group, call the NetGroupDelUser function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_USERS_INFO_0

GROUP_USERS_INFO_1

Group Functions

NetGroupAddUser

NetGroupDelUser

NetGroupGetUsers

NetUserGetGroups

NetUserSetGroups

Network Management Functions

Network Management Overview

NetLocalGroupAdd function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetLocalGroupAdd** function creates a local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

A pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	A local group name. The *buf* parameter points to a LOCALGROUP_INFO_0 structure.
1	A local group name and a comment to associate with the group. The *buf* parameter points to a LOCALGROUP_INFO_1 structure.

`buf`

A pointer to a buffer that contains the local group information structure. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

A pointer to a value that receives the index of the first member of the local group information structure to cause the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error. For more information, see the Remarks section in the NetLocalGroupSetInfo topic.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.
ERROR_ALIAS_EXISTS	The specified local group already exists. This error is returned if the group name member in the structure pointed to by the *buf* parameter is already in use as an alias.
ERROR_INVALID_LEVEL	A *level* parameter is invalid.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if one or more of the members in the structure pointed to by the *buf* parameter is invalid.
NERR_GroupExists	The group name exists. This error is returned if the group name member in the structure pointed to by the *buf* parameter is already in use as a group name.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_UserExists	The user name exists. This error is returned if the group name member in the structure pointed to by the *buf* parameter is already in use as a user name.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the user container is used to perform the access check for this function. The caller must be able to create child objects of the group class.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If the **NetLocalGroupAdd** function returns **ERROR_INVALID_PARAMETER** and a **NULL** pointer was not passed in *parm_err* parameter, on return the *parm_err* parameter indicates the first member of the local group information structure that is invalid. The format of the local group information structure is specified in the *level* parameter. A pointer to the local group information structure is passed in *buf* parameter. The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error.

VALUE	MEMBER

LOCALGROUP_NAME_PARMNUM	If the *level* parameter was 0, the **lgrpi0_name** member of the LOCALGROUP_INFO_0 structure was invalid. If the *level* parameter was 1, the **lgrpi1_name** member of the LOCALGROUP_INFO_1 structure was invalid.
LOCALGROUP_COMMENT_PARMNUM	If the *level* parameter was 1, the **lgrpi1_comment** member of the LOCALGROUP_INFO_1 structure was invalid.

When making requests to a domain controller and Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same results as the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_INFO_0

LOCALGROUP_INFO_1

Local Group Functions

NetLocalGroupAddMembers

NetLocalGroupDel

NetLocalGroupSetInfo

Network Management Functions

Network Management Overview

Network Management Function Buffers

NetLocalGroupAddMember function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupAddMember** function is obsolete. You should use the NetLocalGroupAddMembers function instead.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupAddMember(
  LPCWSTR servername,
  LPCWSTR groupname,
  PSID    membersid
);
```

Parameters

`servername`

TBD

`groupname`

TBD

`membersid`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmaccess.h

NetLocalGroupAddMembers function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupAddMembers** function adds membership of one or more existing user accounts or global group accounts to an existing local group. The function does not change the membership status of users or global groups that are currently members of the local group.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupAddMembers(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  DWORD   totalentries
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group to which the specified users or global groups will be added. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies the security identifier (SID) of the new local group member. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_0 structures.
3	Specifies the domain and name of the new local group member. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_3 structures.

`buf`

Pointer to a buffer that contains the data for the new local group members. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`totalentries`

Specifies the number of entries in the buffer pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
NERR_GroupNotFound	The local group specified by the *groupname* parameter does not exist.
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_NO_SUCH_MEMBER	One or more of the members specified do not exist. Therefore, no new members were added.
ERROR_MEMBER_IN_ALIAS	One or more of the members specified were already members of the local group. No new members were added.
ERROR_INVALID_MEMBER	One or more of the members cannot be added because their account type is invalid. No new members were added.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib

DLL	Netapi32.dll

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupAdd

NetLocalGroupDel

NetLocalGroupDelMembers

NetLocalGroupGetMembers

Network Management Functions

Network Management Overview

NetLocalGroupDel function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupDel** function deletes a local group account and all its members from the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupDel(
  LPCWSTR servername,
  LPCWSTR groupname
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group account to delete. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_GroupNotFound	The local group specified by the *groupname* parameter does not exist.
ERROR_NO_SUCH_ALIAS	The specified local group does not exist.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Local Group Functions

NetLocalGroupAdd

NetLocalGroupDelMembers

Network Management Functions

Network Management Overview

NetLocalGroupDelMember function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupDelMember** function is obsolete. You should use the NetLocalGroupDelMembers function instead.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupDelMember(
  LPCWSTR servername,
  LPCWSTR groupname,
  PSID    membersid
);
```

Parameters

`servername`

TBD

`groupname`

TBD

`membersid`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmaccess.h

NetLocalGroupDelMembers function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupDelMembers** function removes one or more members from an existing local group. Local group members can be users or global groups.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupDelMembers(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  DWORD   totalentries
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group from which the specified users or global groups will be removed. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies the security identifier (SID) of a local group member to remove. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_0 structures.
3	Specifies the domain and name of a local group member to remove. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_3 structures.

`buf`

Pointer to a buffer that specifies the members to be removed. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`totalentries`

Specifies the number of entries in the array pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_GroupNotFound	The local group specified by the *groupname* parameter does not exist.
ERROR_NO_SUCH_MEMBER	One or more of the specified members do not exist. No members were deleted.
ERROR_MEMBER_NOT_IN_ALIAS	One or more of the members specified were not members of the local group. No members were deleted.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupAddMembers

NetLocalGroupDel

NetLocalGroupGetMembers

Network Management Functions

Network Management Overview

NetLocalGroupEnum function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetLocalGroupEnum** function returns information about each local group account on the specified server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupEnum(
  LPCWSTR    servername,
  DWORD      level,
  LPBYTE     *bufptr,
  DWORD      prefmaxlen,
  LPDWORD    entriesread,
  LPDWORD    totalentries,
  PDWORD_PTR resumehandle
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return local group names. The *bufptr* parameter points to an array of LOCALGROUP_INFO_0 structures.
1	Return local group names and the comment associated with each group. The *bufptr* parameter points to an array of LOCALGROUP_INFO_1 structures.

`bufptr`

Pointer to the address of the buffer that receives the information structure. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the approximate total number of entries that could have been enumerated from the current resume position. The total number of entries is only a hint. For more information about determining the exact number of entries, see the following Remarks section.

`resumehandle`

Pointer to a value that contains a resume handle that is used to continue an existing local group search. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, then no resume handle is stored. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
NERR_InvalidComputer	The computer name is invalid.
NERR_BufTooSmall	The return buffer is too small.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The function only returns information to which the caller has Read access. The caller must have List Contents access to the Domain object, and Enumerate Entire SAM Domain access on the SAM Server object located in the System container.

To determine the exact total number of local groups, you must enumerate the entire tree, which can be a costly operation. To enumerate the entire tree, use the *resumehandle* parameter to continue the enumeration for consecutive calls, and use the *entriesread* parameter to accumulate the total number of local groups. If your application is communicating with a domain controller, you should consider using the ADSI LDAP Provider to retrieve this type of data more efficiently. The ADSI LDAP Provider implements a set of ADSI objects that support various ADSI interfaces. For more information, see ADSI Service Providers.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_INFO_0

LOCALGROUP_INFO_1

Local Group Functions

NetLocalGroupGetInfo

NetLocalGroupGetMembers

NetQueryDisplayInformation

Network Management Functions

Network Management Overview

NetLocalGroupGetInfo function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetLocalGroupGetInfo** function retrieves information about a particular local group account on a server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupGetInfo(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group account for which the information will be retrieved. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be the following value.

VALUE	MEANING
1	Return the comment associated with the local group. The *bufptr* parameter points to a LOCALGROUP_INFO_1 structure.

`bufptr`

Pointer to the address of the buffer that receives the return information structure. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.

NERR_InvalidComputer	The computer name is invalid.
NERR_GroupNotFound	The specified local group does not exist.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_INFO_1

Local Group Functions

NetLocalGroupEnum

NetLocalGroupGetMembers

NetLocalGroupSetInfo

NetQueryDisplayInformation

Network Management Functions

Network Management Overview

NetLocalGroupGetMembers function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetLocalGroupGetMembers** function retrieves a list of the members of a particular local group in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory. Local group members can be users or global groups.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupGetMembers(
  LPCWSTR    servername,
  LPCWSTR    localgroupname,
  DWORD      level,
  LPBYTE     *bufptr,
  DWORD      prefmaxlen,
  LPDWORD    entriesread,
  LPDWORD    totalentries,
  PDWORD_PTR resumehandle
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`localgroupname`

Pointer to a constant string that specifies the name of the local group whose members are to be listed. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the security identifier (SID) associated with the local group member. The *bufptr* parameter points to an array of LOCALGROUP_MEMBERS_INFO_0 structures.
1	Return the SID and account information associated with the local group member. The *bufptr* parameter points to an array of LOCALGROUP_MEMBERS_INFO_1 structures.
2	Return the SID, account information, and the domain name associated with the local group member. The *bufptr* parameter points to an array of LOCALGROUP_MEMBERS_INFO_2 structures.

3	Return the account and domain names of the local group member. The *bufptr* parameter points to an array of LOCALGROUP_MEMBERS_INFO_3 structures.

`bufptr`

Pointer to the address that receives the return information structure. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the total number of entries that could have been enumerated from the current resume position.

`resumehandle`

Pointer to a value that contains a resume handle which is used to continue an existing group member search. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, then no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NO_SUCH_ALIAS	The specified local group does not exist.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied

based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

If this function returns **ERROR_MORE_DATA**, then it must be repeatedly called until **ERROR_SUCCESS** or **NERR_success** is returned. Failure to do so can result in an RPC connection leak.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_1

LOCALGROUP_MEMBERS_INFO_2

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupEnum

NetLocalGroupGetInfo

NetLocalGroupSetMembers

Network Management Functions

Network Management Overview

NetLocalGroupSetInfo function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetLocalGroupSetInfo** function changes the name of an existing local group. The function also associates a comment with a local group.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupSetInfo(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group account to modify. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies the local group name. The *buf* parameter points to a LOCALGROUP_INFO_0 structure. Use this level to change the name of an existing local group.
1	Specifies the local group name and a comment to associate with the group. The *buf* parameter points to a LOCALGROUP_INFO_1 structure.
1002	Specifies a comment to associate with the local group. The *buf* parameter points to a LOCALGROUP_INFO_1002 structure.

`buf`

Pointer to a buffer that contains the local group information. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first member of the local group information structure that

caused the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid. For more information, see the following Remarks section.
ERROR_NO_SUCH_ALIAS	The specified local group does not exist.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_InvalidComputer	The computer name is invalid.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function. Typically, callers must have write access to the entire object for calls to this function to succeed.

To specify the new name of an existing local group, call **NetLocalGroupSetInfo** with LOCALGROUP_INFO_0 and specify a value using the **lgrpi0_name** member. If you call the **NetLocalGroupSetInfo** function with LOCALGROUP_INFO_1 and specify a new value using the **lgrpi1_name** member, that value will be ignored.

If the **NetLocalGroupSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the first member of the local group information structure that is invalid. (A local group information structure begins with LOCALGROUP_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix lgrpi*_ indicates that the member can begin with multiple prefixes, for example, lgrpi0_ or lgrpi1_.)

VALUE	MEMBER
LOCALGROUP_NAME_PARMNUM	lgrpi*_name
LOCALGROUP_COMMENT_PARMNUM	lgrpi*_comment

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_INFO_0

LOCALGROUP_INFO_1

LOCALGROUP_INFO_1002

Local Group Functions

NetLocalGroupGetInfo

Network Management Functions

Network Management Overview

NetLocalGroupSetMembers function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetLocalGroupSetMembers** function sets the membership for the specified local group. Each user or global group specified is made a member of the local group. Users or global groups that are not specified but who are currently members of the local group will have their membership revoked.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetLocalGroupSetMembers(
  LPCWSTR servername,
  LPCWSTR groupname,
  DWORD   level,
  LPBYTE  buf,
  DWORD   totalentries
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`groupname`

Pointer to a constant string that specifies the name of the local group in which the specified users or global groups should be granted membership. For more information, see the following Remarks section.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies the security identifier (SID) associated with a local group member. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_0 structures.
3	Specifies the account and domain names of the local group member. The *buf* parameter points to an array of LOCALGROUP_MEMBERS_INFO_3 structures.

`buf`

Pointer to the buffer that contains the member information. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`totalentries`

Specifies a value that contains the total number of entries in the buffer pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
NERR_GroupNotFound	The group specified by the *groupname* parameter does not exist.
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_NO_SUCH_MEMBER	One or more of the members doesn't exist. The local group membership was not changed.
ERROR_INVALID_MEMBER	One or more of the members cannot be added because it has an invalid account type. The local group membership was not changed.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the LocalGroup object is used to perform the access check for this function.

You can replace the local group membership with an entirely new list of members by calling the **NetLocalGroupSetMembers** function. The typical sequence of steps to perform this follows.

To replace the local group membership

1. Call the NetLocalGroupGetMembers function to retrieve the current membership list.
2. Modify the returned membership list to reflect the new membership.
3. Call the **NetLocalGroupSetMembers** function to replace the old membership list with the new membership list.

To add one or more existing user accounts or global group accounts to an existing local group, you can call the NetLocalGroupAddMembers function. To remove one or more members from an existing local group, call the NetLocalGroupDelMembers function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management local group functions. For more information, see IADsGroup.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_MEMBERS_INFO_0

LOCALGROUP_MEMBERS_INFO_3

Local Group Functions

NetLocalGroupAddMembers

NetLocalGroupDelMembers

NetLocalGroupGetMembers

Network Management Functions

Network Management Overview

NetQueryDisplayInformation function (lmaccess.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetQueryDisplayInformation** function returns user account, computer, or group account information. Call this function to quickly enumerate account information for display in user interfaces.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetQueryDisplayInformation(
  LPCWSTR ServerName,
  DWORD   Level,
  DWORD   Index,
  DWORD   EntriesRequested,
  DWORD   PreferredMaximumLength,
  LPDWORD ReturnedEntryCount,
  PVOID   *SortedBuffer
);
```

Parameters

`ServerName`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`Level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
1	Return user account information. The *SortedBuffer* parameter points to an array of NET_DISPLAY_USER structures.
2	Return individual computer information. The *SortedBuffer* parameter points to an array of NET_DISPLAY_MACHINE structures.
3	Return group account information. The *SortedBuffer* parameter points to an array of NET_DISPLAY_GROUP structures.

`Index`

Specifies the index of the first entry for which to retrieve information. Specify zero to retrieve account information beginning with the first display information entry. For more information, see the following Remarks section.

`EntriesRequested`

Specifies the maximum number of entries for which to retrieve information. On Windows 2000 and later, each call to **NetQueryDisplayInformation** returns a maximum of 100 objects.

`PreferredMaximumLength`

Specifies the preferred maximum size, in bytes, of the system-allocated buffer returned in the *SortedBuffer* parameter. It is recommended that you set this parameter to MAX_PREFERRED_LENGTH.

`ReturnedEntryCount`

Pointer to a value that receives the number of entries in the buffer returned in the *SortedBuffer* parameter. If this parameter is zero, there are no entries with an index as large as that specified. Entries may be returned when the function's return value is either NERR_Success or ERROR_MORE_DATA.

`SortedBuffer`

Pointer to a buffer that receives a pointer to a system-allocated buffer that specifies a sorted list of the requested information. The format of this data depends on the value of the *Level* parameter. Because this buffer is allocated by the system, it must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA. For more information, see the following Return Values section, and the topics Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The *Level* parameter specifies an invalid value.
ERROR_MORE_DATA	More entries are available. That is, the last entry returned in the *SortedBuffer* parameter is not the last entry available. To retrieve additional entries, call **NetQueryDisplayInformation** again with the *Index* parameter set to the value returned in the **next_index** member of the last entry in *SortedBuffer*. Note that you should not use the value of the **next_index** member for any purpose except to retrieve more data with additional calls to **NetQueryDisplayInformation**.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The **NetQueryDisplayInformation** function only returns information to which the caller has Read access. The caller must have List Contents access to the Domain object, and Enumerate Entire SAM Domain access on the SAM Server object located in the System container.

The **NetQueryDisplayInformation** and NetGetDisplayInformationIndex functions provide an efficient mechanism for enumerating user and group accounts. When possible, use these functions instead of the NetUserEnum function or the NetGroupEnum function.

To enumerate trusting domains or member computer accounts, call NetUserEnum, specifying the appropriate filter value to obtain the account information you require. To enumerate trusted domains, call the LsaEnumerateTrustedDomains or LsaEnumerateTrustedDomainsEx function.

The number of entries returned by this function depends on the security descriptor located on the root domain object. The API will return either the first 100 entries or the entire set of entries in the domain, depending on the access privileges of the user. The ACE used to control this behavior is "SAM-Enumerate-Entire-Domain", and is granted to Authenticated Users by default. Administrators can modify this setting to allow users to enumerate the entire domain.

Each call to **NetQueryDisplayInformation** returns a maximum of 100 objects. Calling the **NetQueryDisplayInformation** function to enumerate domain account information can be costly in terms of performance. If you are programming for Active Directory, you may be able to use methods on the IDirectorySearch interface to make paged queries against the domain. For more information, see IDirectorySearch::SetSearchPreference and IDirectorySearch::ExecuteSearch. To enumerate trusted domains, call the LsaEnumerateTrustedDomainsEx function.

Examples

The following code sample demonstrates how to return group account information using a call to the **NetQueryDisplayInformation** function. If the user specifies a server name, the sample first calls the MultiByteToWideChar function to convert the name to Unicode. The sample calls **NetQueryDisplayInformation**, specifying information level 3 (NET_DISPLAY_GROUP) to retrieve group account information. If there are entries to return, the sample returns the data and prints the group information. Finally, the code sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif

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

#pragma comment(lib, "netapi32.lib")

void main( int argc, char *argv[ ] )
{
   PNET_DISPLAY_GROUP pBuff, p;
   DWORD res, dwRec, i = 0;
   //
   // You can pass a NULL or empty string
   //  to retrieve the local information.
   //
   TCHAR szServer[255]=TEXT("");

   if(argc > 1)
      //
      // Check to see if a server name was passed;
      //  if so, convert it to Unicode.
      //
      MultiByteToWideChar(CP_ACP, 0, argv[1], -1, szServer, 255);

   do // begin do
   {
      //
      // Call the NetQueryDisplayInformation function;
      //   specify information level 3 (group account information).
      //
```

```
      //
      res = NetQueryDisplayInformation(szServer, 3, i, 1000, MAX_PREFERRED_LENGTH, &dwRec, (PVOID*) &pBuff);
      //
      // If the call succeeds,
      //
      if((res==ERROR_SUCCESS) || (res==ERROR_MORE_DATA))
      {
         p = pBuff;
         for(;dwRec>0;dwRec--)
         {
            //
            // Print the retrieved group information.
            //
            printf("Name:      %S\n"
                  "Comment:   %S\n"
                  "Group ID:  %u\n"
                  "Attributes: %u\n"
                  "--------------------------------\n",
                  p->grpi3_name,
                  p->grpi3_comment,
                  p->grpi3_group_id,
                  p->grpi3_attributes);
            //
            // If there is more data, set the index.
            //
            i = p->grpi3_next_index;
            p++;
         }
         //
         // Free the allocated memory.
         //
         NetApiBufferFree(pBuff);
      }
      else
         printf("Error: %u\n", res);
   //
   // Continue while there is more data.
   //
   } while (res==ERROR_MORE_DATA); // end do
   return;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Get Functions

LsaEnumerateTrustedDomains

LsaEnumerateTrustedDomainsEx

NET_DISPLAY_GROUP

NET_DISPLAY_MACHINE

NET_DISPLAY_USER

NetGetDisplayInformationIndex

NetGroupEnum

NetUserEnum

Network Management Functions

Network Management Overview

NetUserAdd function (lmaccess.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetUserAdd** function adds a user account and assigns a password and privilege level.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
1	Specifies information about the user account. The *buf* parameter points to a USER_INFO_1 structure. When you specify this level, the call initializes certain attributes to their default values. For more information, see the following Remarks section.
2	Specifies level one information and additional attributes about the user account. The *buf* parameter points to a USER_INFO_2 structure.
3	Specifies level two information and additional attributes about the user account. This level is valid only on servers. The *buf* parameter points to a USER_INFO_3 structure. Note that it is recommended that you use USER_INFO_4 instead.
4	Specifies level two information and additional attributes about the user account. This level is valid only on servers. The *buf* parameter points to a USER_INFO_4 structure. **Windows 2000:** This level is not supported.

`buf`

Pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter.

For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first member of the user information structure that causes ERROR_INVALID_PARAMETER. If this parameter is **NULL**, the index is not returned on error. For more information, see the NetUserSetInfo function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_GroupExists	The group already exists.
NERR_UserExists	The user account already exists.
NERR_PasswordTooShort	The password is shorter than required. (The password could also be too long, be too recent in its change history, not have enough unique characters, or not meet another password policy requirement.)

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the user container is used to perform the access check for this function. The caller must be able to create child objects of the user class.

Server users must use a system in which the server creates a system account for the new user. The creation of this account is controlled by several parameters in the server's LanMan.ini file.

If the newly added user already exists as a system user, the **usri1_home_dir** member of the USER_INFO_1 structure is ignored.

When you call the **NetUserAdd** function and specify information level 1, the call initializes the additional members in the USER_INFO_2, USER_INFO_3, and USER_INFO_4 structures to their default values. You can change the default values by making subsequent calls to the NetUserSetInfo function. The default values supplied are listed following. (The prefix usriX indicates that the member can begin with multiple prefixes, for example, usri2_ or usri4_.)

MEMBER	DEFAULT VALUE
usriX_auth_flags	None (0)
usriX_full_name	None (null string)
usriX_usr_comment	None (null string)
usriX_parms	None (null string)
usriX_workstations	All (null string)
usriX_acct_expires	Never (TIMEQ_FOREVER)
usriX_max_storage	Unlimited (USER_MAXSTORAGE_UNLIMITED)
usriX_logon_hours	Logon allowed at any time (each element 0xFF; all bits set to 1)
usriX_logon_server	Any domain controller (*)
usriX_country_code	0
usriX_code_page	0

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Examples

The following code sample demonstrates how to add a user account and assign a privilege level using a call to the **NetUserAdd** function. The code sample fills in the members of the USER_INFO_1 structure and calls **NetUserAdd**, specifying information level 1.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   USER_INFO_1 ui;
   DWORD dwLevel = 1;
   DWORD dwError = 0;
   NET_API_STATUS nStatus;

   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
      exit(1);
   }
   //
   // Set up the USER_INFO_1 structure.
   //  USER_PRIV_USER: name identifies a user,
   //    rather than an administrator or a guest.
   //  UF_SCRIPT: required
   //
   ui.usri1_name = argv[2];
   ui.usri1_password = argv[2];
   ui.usri1_priv = USER_PRIV_USER;
   ui.usri1_home_dir = NULL;
   ui.usri1_comment = NULL;
   ui.usri1_flags = UF_SCRIPT;
   ui.usri1_script_path = NULL;
   //
   // Call the NetUserAdd function, specifying level 1.
   //
   nStatus = NetUserAdd(argv[1],
                        dwLevel,
                        (LPBYTE)&ui,
                        &dwError);
   //
   // If the call succeeds, inform the user.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"User %s has been successfully added on %s\n",
               argv[2], argv[1]);
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserDel

NetUserEnum

NetUserSetInfo

Network Management Functions

Network Management Overview

USER_INFO_1

USER_INFO_2

USER_INFO_4

User Functions

NetUserChangePassword function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetUserChangePassword** function changes a user's password for a specified network server or domain.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserChangePassword(
  LPCWSTR domainname,
  LPCWSTR username,
  LPCWSTR oldpassword,
  LPCWSTR newpassword
);
```

Parameters

`domainname`

A pointer to a constant string that specifies the DNS or NetBIOS name of a remote server or domain on which the function is to execute. If this parameter is **NULL**, the logon domain of the caller is used.

`username`

A pointer to a constant string that specifies a user name. The **NetUserChangePassword** function changes the password for the specified user.

If this parameter is **NULL**, the logon name of the caller is used. For more information, see the following Remarks section.

`oldpassword`

A pointer to a constant string that specifies the user's old password.

`newpassword`

A pointer to a constant string that specifies the user's new password.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PASSWORD	The user has entered an invalid password.

NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_UserNotFound	The user name could not be found.
NERR_PasswordTooShort	The password is shorter than required. (The password could also be too long, be too recent in its change history, not have enough unique characters, or not meet another password policy requirement.)

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same result you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If an application calls the **NetUserChangePassword** function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. A user can change his or her own password. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function. In addition, the caller must have the "Change password" control access right on the User object. This right is granted to Anonymous Logon and Everyone by default.

Note that for the function to succeed, the *oldpassword* parameter must match the password as it currently exists.

In some cases, the process that calls the **NetUserChangePassword** function must also have the SE_CHANGE_NOTIFY_NAME privilege enabled; otherwise, **NetUserChangePassword** fails and GetLastError returns ERROR_ACCESS_DENIED. This privilege is not required for the LocalSystem account or for accounts that are members of the administrators group. By default, SE_CHANGE_NOTIFY_NAME is enabled for all users, but some administrators may disable the privilege for everyone. For more information about account privileges, see Privileges and Authorization Constants.

See Forcing a User to Change the Logon Password for a code sample that demonstrates how to force a user to change the logon password on the next logon using the NetUserGetInfo and NetUserSetInfo functions.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

The **NetUserChangePassword** function does not control how the *oldpassword* and *newpassword* parameters are secured when sent over the network to a remote server. Any encryption of these parameters is handled by the Remote Procedure Call (RPC) mechanism supported by the network redirector that provides the network transport. Encryption is also controlled by the security mechanisms supported by the local computer and the security mechanisms supported by remote network server or domain specified in the *domainname* parameter. For more details on security when the Microsoft network redirector is used and the remote network server is

running Microsoft Windows, see the protocol documentation for MS-RPCE, MS-SAMR, MS-SPNG, and MS-NLMP.

Examples

The following code sample demonstrates how to change a user's password with a call to the **NetUserChangePassword** function. All parameters to the function are required.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwError = 0;
   NET_API_STATUS nStatus;
   //
   // All parameters are required.
   //
   if (argc != 5)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName OldPassword NewPassword\n", argv[0]);
      exit(1);
   }
   //
   // Call the NetUserChangePassword function.
   //
   nStatus = NetUserChangePassword(argv[1], argv[2], argv[3], argv[4]);
   //
   // If the call succeeds, inform the user.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"User password has been changed successfully\n");
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib

DLL	Netapi32.dll

See also

NetUserGetInfo

NetUserSetInfo

Network Management Functions

Network Management Overview

User Functions

NetUserDel function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetUserDel** function deletes a user account from a server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserDel(
  LPCWSTR servername,
  LPCWSTR username
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

Pointer to a constant string that specifies the name of the user account to delete. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_UserNotFound	The user name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function.

An account cannot be deleted while a user or application is accessing a server resource. If the user was added to the system with a call to the NetUserAdd function, deleting the user also deletes the user's system account.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Examples

The following code sample demonstrates how to delete a user account with a call to the **NetUserDel** function.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwError = 0;
   NET_API_STATUS nStatus;
   //
   // All parameters are required.
   //
   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
      exit(1);
   }
   //
   // Call the NetUserDel function to delete the share.
   //
   nStatus = NetUserDel(argv[1], argv[2]);
   //
   // Display the result of the call.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"User %s has been successfully deleted on %s\n",
               argv[2], argv[1]);
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserAdd

NetUserEnum

NetUserSetInfo

Network Management Functions

Network Management Overview

User Functions

NetUserEnum function (lmaccess.h)

2/1/2021 • 8 minutes to read • Edit Online

The **NetUserEnum** function retrieves information about all user accounts on a server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserEnum(
  LPCWSTR servername,
  DWORD   level,
  DWORD   filter,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  PDWORD  resume_handle
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return user account names. The *bufptr* parameter points to an array of USER_INFO_0 structures.
1	Return detailed information about user accounts. The *bufptr* parameter points to an array of USER_INFO_1 structures.
2	Return detailed information about user accounts, including authorization levels and logon information. The *bufptr* parameter points to an array of USER_INFO_2 structures.
3	Return detailed information about user accounts, including authorization levels, logon information, RIDs for the user and the primary group, and profile information. The *bufptr* parameter points to an array of USER_INFO_3 structures.
10	Return user and account names and comments. The *bufptr* parameter points to an array of USER_INFO_10 structures.
11	Return detailed information about user accounts. The *bufptr* parameter points to an array of USER_INFO_11 structures.

20	Return the user's name and identifier and various account attributes. The *bufptr* parameter points to an array of USER_INFO_20 structures. Note that on Windows XP and later, it is recommended that you use USER_INFO_23 instead.

`filter`

A value that specifies the user account types to be included in the enumeration. A value of zero indicates that all normal user, trust data, and machine account data should be included.

This parameter can also be a combination of the following values.

VALUE	MEANING
FILTER_TEMP_DUPLICATE_ACCOUNT	Enumerates account data for users whose primary account is in another domain. This account type provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
FILTER_NORMAL_ACCOUNT	Enumerates normal user account data. This account type is associated with a typical user.
FILTER_INTERDOMAIN_TRUST_ACCOUNT	Enumerates interdomain trust account data. This account type is associated with a trust account for a domain that trusts other domains.
FILTER_WORKSTATION_TRUST_ACCOUNT	Enumerates workstation or member server trust account data. This account type is associated with a machine account for a computer that is a member of the domain.
FILTER_SERVER_TRUST_ACCOUNT	Enumerates member server machine account data. This account type is associated with a computer account for a backup domain controller that is a member of the domain.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter.

The buffer for this data is allocated by the system and the application must call the NetApiBufferFree function to free the allocated memory when the data returned is no longer needed. Note that you must free the buffer even if the **NetUserEnum** function fails with ERROR_MORE_DATA.

`prefmaxlen`

The preferred maximum length, in bytes, of the returned data. If you specify MAX_PREFERRED_LENGTH, the **NetUserEnum** function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint. If your application is communicating with a Windows 2000 or later domain controller, you should consider using the ADSI LDAP Provider to retrieve this type of data more efficiently. The ADSI LDAP Provider implements a set of ADSI objects that support various ADSI interfaces. For more information, see ADSI Service Providers.

LAN Manager: If the call is to a computer that is running LAN Manager 2.*x*, the *totalentries* parameter will always reflect the total number of entries in the database no matter where it is in the resume sequence.

`resume_handle`

A pointer to a value that contains a resume handle which is used to continue an existing user search. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, then no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter is set to a value not supported.
NERR_BufTooSmall	The buffer is too small to contain an entry. No information has been written to the buffer.
NERR_InvalidComputer	The computer name is invalid.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.

Remarks

The **NetUserEnum** function retrieves information about all user accounts on a specified remote server or the local computer.

The NetQueryDisplayInformation function can be used to quickly enumerate user, computer, or global group account information for display in user interfaces .

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call the **NetUserEnum** function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security

Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The **NetUserEnum** function only returns information to which the caller has Read access. The caller must have List Contents access to the Domain object, and Enumerate Entire SAM Domain access on the SAM Server object located in the System container.

The LsaEnumerateTrustedDomains or LsaEnumerateTrustedDomainsEx function can be used to retrieve the names and SIDs of domains trusted by a Local Security Authority (LSA) policy object.

The **NetUserEnum** function does not return all system users. It returns only those users who have been added with a call to the NetUserAdd function. There is no guarantee that the list of users will be returned in sorted order.

If you call the **NetUserEnum** function and specify information level 1, 2, or 3, for the *level* parameter, the password member of each structure retrieved is set to **NULL** to maintain password security.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

The **NetUserEnum** function does not support a *level* parameter of 4 and the USER_INFO_4 structure. The NetUserGetInfo function supports a *level* parameter of 4 and the **USER_INFO_4** structure.

Examples

The following code sample demonstrates how to retrieve information about the user accounts on a server with a call to the **NetUserEnum** function. The sample calls **NetUserEnum**, specifying information level 0 (USER_INFO_0) to enumerate only global user accounts. If the call succeeds, the code loops through the entries and prints the name of each user account. Finally, the code sample frees the memory allocated for the information buffer and prints a total of the users enumerated.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t *argv[])
{
   LPUSER_INFO_0 pBuf = NULL;
   LPUSER_INFO_0 pTmpBuf;
   DWORD dwLevel = 0;
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   DWORD dwResumeHandle = 0;
   DWORD i;
   DWORD dwTotalCount = 0;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
```

```
//
if (argc == 2)
   pszServerName =  (LPTSTR) argv[1];
wprintf(L"\nUser account on %s: \n", pszServerName);
//
// Call the NetUserEnum function, specifying level 0;
//   enumerate global user account types only.
//
do // begin do
{
   nStatus = NetUserEnum((LPCWSTR) pszServerName,
                         dwLevel,
                         FILTER_NORMAL_ACCOUNT, // global users
                         (LPBYTE*)&pBuf,
                         dwPrefMaxLen,
                         &dwEntriesRead,
                         &dwTotalEntries,
                         &dwResumeHandle);
   //
   // If the call succeeds,
   //
   if ((nStatus == NERR_Success) || (nStatus == ERROR_MORE_DATA))
   {
      if ((pTmpBuf = pBuf) != NULL)
      {
         //
         // Loop through the entries.
         //
         for (i = 0; (i < dwEntriesRead); i++)
         {
            assert(pTmpBuf != NULL);

            if (pTmpBuf == NULL)
            {
               fprintf(stderr, "An access violation has occurred\n");
               break;
            }
            //
            //  Print the name of the user account.
            //
            wprintf(L"\t-- %s\n", pTmpBuf->usri0_name);

            pTmpBuf++;
            dwTotalCount++;
         }
      }
   }
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated buffer.
   //
   if (pBuf != NULL)
   {
      NetApiBufferFree(pBuf);
      pBuf = NULL;
   }
}
// Continue to call NetUserEnum while
//  there are more entries.
//
while (nStatus == ERROR_MORE_DATA); // end do
//
// Check again for allocated memory.
//
if (pBuf != NULL)
   NetApiBufferFree(pBuf);
```

```
        NetApiBufferFree(pBuf);
    //
    // Print the final count of users enumerated.
    //
    fprintf(stderr, "\nTotal of %d entries enumerated\n", dwTotalCount);

    return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LsaEnumerateTrustedDomains

LsaEnumerateTrustedDomainsEx

NetQueryDisplayInformation

NetUserAdd

NetUserGetGroups

NetUserGetInfo

Network Management Functions

Network Management Overview

USER_INFO_0

USER_INFO_1

USER_INFO_10

USER_INFO_11

USER_INFO_2

USER_INFO_20

USER_INFO_23

USER_INFO_3

NetUserGetGroups function (lmaccess.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetUserGetGroups** function retrieves a list of global groups to which a specified user belongs.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserGetGroups(
  LPCWSTR servername,
  LPCWSTR username,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

A pointer to a constant string that specifies the name of the user to search for in each group account. For more information, see the following Remarks section.

`level`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
0	Return the names of the global groups to which the user belongs. The *bufptr* parameter points to an array of GROUP_USERS_INFO_0 structures.
1	Return the names of the global groups to which the user belongs with attributes. The *bufptr* parameter points to an array of GROUP_USERS_INFO_1 structures.

`bufptr`

A pointer to the buffer that receives the data. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

The preferred maximum length, in bytes, of returned data. If MAX_PREFERRED_LENGTH is specified, the function allocates the amount of memory required for the data. If another value is specified in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the

function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually retrieved.

`totalentries`

A pointer to a value that receives the total number of entries that could have been retrieved.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access rights to the requested information.
ERROR_BAD_NETPATH	The network path was not found. This error is returned if the *servername* parameter could not be found.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter was specified as a value other than 0 or 1.
ERROR_INVALID_NAME	The name syntax is incorrect. This error is returned if the *servername* parameter has leading or trailing blanks or contains an illegal character.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to complete the operation.
NERR_InternalError	An internal error occurred.
NERR_UserNotFound	The user could not be found. This error is returned if the *username* could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information

about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function.

To retrieve a list of the local groups to which a user belongs, you can call the NetUserGetLocalGroups function. Network groups are separate and distinct from Windows NT system groups.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Examples

The following code sample demonstrates how to retrieve a list of global groups to which a user belongs with a call to the **NetUserGetGroups** function. The sample calls **NetUserGetGroups**, specifying information level 0 (GROUP_USERS_INFO_0). The code loops through the entries and prints the name of the global groups in which the user has membership. The sample also prints the total number of entries that are available and the number of entries actually enumerated if they do not match. Finally, the code sample frees the memory allocated for the buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t *argv[])
{
   LPGROUP_USERS_INFO_0 pBuf = NULL;
   DWORD dwLevel = 0;
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   NET_API_STATUS nStatus;

   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
      exit(1);
   }
   //
   // Call the NetUserGetGroups function, specifying level 0.
   //
   nStatus = NetUserGetGroups(argv[1],
                              argv[2],
                              dwLevel,
                              (LPBYTE*)&pBuf,
                              dwPrefMaxLen,
                              &dwEntriesRead,
                              &dwTotalEntries);
   //
   // If the call succeeds,
   //
   if (nStatus == NERR_Success)
   {
      LPGROUP_USERS_INFO_0 pTmpBuf;
      DWORD i;
      DWORD dwTotalCount = 0;
```

```
      if ((pTmpBuf = pBuf) != NULL)
      {
         fprintf(stderr, "\nGlobal group(s):\n");
         //
         // Loop through the entries;
         //  print the name of the global groups
         //  to which the user belongs.
         //
         for (i = 0; i < dwEntriesRead; i++)
         {
            assert(pTmpBuf != NULL);

            if (pTmpBuf == NULL)
            {
               fprintf(stderr, "An access violation has occurred\n");
               break;
            }

            wprintf(L"\t-- %s\n", pTmpBuf->grui0_name);

            pTmpBuf++;
            dwTotalCount++;
         }
      }
      //
      // If all available entries were
      //  not enumerated, print the number actually
      //  enumerated and the total number available.
      //
      if (dwEntriesRead < dwTotalEntries)
         fprintf(stderr, "\nTotal entries: %d", dwTotalEntries);
      //
      // Otherwise, just print the total.
      //
      printf("\nEntries enumerated: %d\n", dwTotalCount);
   }
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated buffer.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib

DLL	Netapi32.dll

See also

GROUP_USERS_INFO_0

GROUP_USERS_INFO_1

NetGroupGetUsers

NetQueryDisplayInformation

NetUserGetInfo

NetUserGetLocalGroups

Network Management Functions

Network Management Overview

User Functions

NetUserGetInfo function (lmaccess.h)

2/1/2021 • 8 minutes to read • Edit Online

The **NetUserGetInfo** function retrieves information about a particular user account on a server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserGetInfo(
  LPCWSTR servername,
  LPCWSTR username,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

A pointer to a constant string that specifies the name of the user account for which to return information. For more information, see the following Remarks section.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the user account name. The *bufptr* parameter points to a USER_INFO_0 structure.
1	Return detailed information about the user account. The *bufptr* parameter points to a USER_INFO_1 structure.
2	Return detailed information and additional attributes about the user account. The *bufptr* parameter points to a USER_INFO_2 structure.
3	Return detailed information and additional attributes about the user account. This level is valid only on servers. The *bufptr* parameter points to a USER_INFO_3 structure. Note that it is recommended that you use USER_INFO_4 instead.

4	Return detailed information and additional attributes about the user account. This level is valid only on servers. The *bufptr* parameter points to a USER_INFO_4 structure. **Note** This level is supported on Windows XP and later.
10	Return user and account names and comments. The *bufptr* parameter points to a USER_INFO_10 structure.
11	Return detailed information about the user account. The *bufptr* parameter points to a USER_INFO_11 structure.
20	Return the user's name and identifier and various account attributes. The *bufptr* parameter points to a USER_INFO_20 structure. Note that on Windows XP and later, it is recommended that you use USER_INFO_23 instead.
23	Return the user's name and identifier and various account attributes. The *bufptr* parameter points to a USER_INFO_23 structure. **Note** This level is supported on Windows XP and later.
24	Return user account information for accounts which are connected to an Internet identity. The *bufptr* parameter points to a USER_INFO_24 structure. **Note** The level is supported on Windows 8 and Windows Server 2012.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.

ERROR_BAD_NETPATH	The network path specified in the *servername* parameter was not found.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
NERR_InvalidComputer	The computer name is invalid.
NERR_UserNotFound	The user name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

If the information level specified in the *level* parameter is set to 24, the *servername* parameter specified must resolve to the local computer. If the *servername* resolves to a remote computer or to a domain controller, the **NetUserGetInfo** function will fail.

Examples

The following code sample demonstrates how to retrieve information about a particular user account with a call to the **NetUserGetInfo** function. The sample calls **NetUserGetInfo**, specifying various information levels . If the call succeeds, the code prints information about the user account. Finally, the sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "advapi32.lib")
#pragma comment(lib, "netapi32.lib")

#include <windows.h>
#include <stdio.h>
#include <assert.h>
#include <lm.h>
#include <sddl.h>               /* for ConvertSidToStringSid function */

int wmain(int argc, wchar_t * argv[])
```

```
{
    DWORD dwLevel = 0;

    LPUSER_INFO_0 pBuf = NULL;
    LPUSER_INFO_1 pBuf1 = NULL;
    LPUSER_INFO_2 pBuf2 = NULL;
    LPUSER_INFO_3 pBuf3 = NULL;
    LPUSER_INFO_4 pBuf4 = NULL;
    LPUSER_INFO_10 pBuf10 = NULL;
    LPUSER_INFO_11 pBuf11 = NULL;
    LPUSER_INFO_20 pBuf20 = NULL;
    LPUSER_INFO_23 pBuf23 = NULL;

    NET_API_STATUS nStatus;

    LPTSTR sStringSid = NULL;

    int i = 0;
    int j = 0;

    if (argc != 3)
    {
        fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
        exit(1);
    }

    while (i < 24)
    {

        //
        // Call the NetUserGetInfo function.
        //
        dwLevel = i;
        wprintf
            (L"\nCalling NetUserGetinfo with Servername=%s Username=%s Level=%d\n",
             argv[1], argv[2], dwLevel);
        nStatus = NetUserGetInfo(argv[1], argv[2], dwLevel, (LPBYTE *) & pBuf);
        //
        // If the call succeeds, print the user information.
        //
        if (nStatus == NERR_Success)
        {
            if (pBuf != NULL)
            {

                switch (i)
                {
                case 0:
                    wprintf(L"\tUser account name: %s\n", pBuf->usri0_name);
                    break;
                case 1:
                    pBuf1 = (LPUSER_INFO_1) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf1->usri1_name);
                    wprintf(L"\tPassword: %s\n", pBuf1->usri1_password);
                    wprintf(L"\tPassword age (seconds): %d\n",
                            pBuf1->usri1_password_age);
                    wprintf(L"\tPrivilege level: %d\n", pBuf1->usri1_priv);
                    wprintf(L"\tHome directory: %s\n", pBuf1->usri1_home_dir);
                    wprintf(L"\tUser comment: %s\n", pBuf1->usri1_comment);
                    wprintf(L"\tFlags (in hex): %x\n", pBuf1->usri1_flags);
                    wprintf(L"\tScript path: %s\n", pBuf1->usri1_script_path);
                    break;
                case 2:
                    pBuf2 = (LPUSER_INFO_2) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf2->usri2_name);
                    wprintf(L"\tPassword: %s\n", pBuf2->usri2_password);
                    wprintf(L"\tPassword age (seconds): %d\n",
                            pBuf2->usri2_password_age);
                    wprintf(L"\tPrivilege level: %d\n", pBuf2->usri2_priv);
```

```
            wprintf(L"\tHome directory: %s\n", pBuf2->usri2_home_dir);
            wprintf(L"\tComment: %s\n", pBuf2->usri2_comment);
            wprintf(L"\tFlags (in hex): %x\n", pBuf2->usri2_flags);
            wprintf(L"\tScript path: %s\n", pBuf2->usri2_script_path);
            wprintf(L"\tAuth flags (in hex): %x\n",
                    pBuf2->usri2_auth_flags);
            wprintf(L"\tFull name: %s\n", pBuf2->usri2_full_name);
            wprintf(L"\tUser comment: %s\n", pBuf2->usri2_usr_comment);
            wprintf(L"\tParameters: %s\n", pBuf2->usri2_parms);
            wprintf(L"\tWorkstations: %s\n", pBuf2->usri2_workstations);
            wprintf
                (L"\tLast logon (seconds since January 1, 1970 GMT): %d\n",
                 pBuf2->usri2_last_logon);
            wprintf
                (L"\tLast logoff (seconds since January 1, 1970 GMT): %d\n",
                 pBuf2->usri2_last_logoff);
            wprintf
                (L"\tAccount expires (seconds since January 1, 1970 GMT): %d\n",
                 pBuf2->usri2_acct_expires);
            wprintf(L"\tMax storage: %d\n", pBuf2->usri2_max_storage);
            wprintf(L"\tUnits per week: %d\n",
                    pBuf2->usri2_units_per_week);
            wprintf(L"\tLogon hours:");
            for (j = 0; j < 21; j++)
            {
                printf(" %x", (BYTE) pBuf2->usri2_logon_hours[j]);
            }
            wprintf(L"\n");
            wprintf(L"\tBad password count: %d\n",
                    pBuf2->usri2_bad_pw_count);
            wprintf(L"\tNumber of logons: %d\n",
                    pBuf2->usri2_num_logons);
            wprintf(L"\tLogon server: %s\n", pBuf2->usri2_logon_server);
            wprintf(L"\tCountry code: %d\n", pBuf2->usri2_country_code);
            wprintf(L"\tCode page: %d\n", pBuf2->usri2_code_page);
            break;
        case 4:
            pBuf4 = (LPUSER_INFO_4) pBuf;
            wprintf(L"\tUser account name: %s\n", pBuf4->usri4_name);
            wprintf(L"\tPassword: %s\n", pBuf4->usri4_password);
            wprintf(L"\tPassword age (seconds): %d\n",
                    pBuf4->usri4_password_age);
            wprintf(L"\tPrivilege level: %d\n", pBuf4->usri4_priv);
            wprintf(L"\tHome directory: %s\n", pBuf4->usri4_home_dir);
            wprintf(L"\tComment: %s\n", pBuf4->usri4_comment);
            wprintf(L"\tFlags (in hex): %x\n", pBuf4->usri4_flags);
            wprintf(L"\tScript path: %s\n", pBuf4->usri4_script_path);
            wprintf(L"\tAuth flags (in hex): %x\n",
                    pBuf4->usri4_auth_flags);
            wprintf(L"\tFull name: %s\n", pBuf4->usri4_full_name);
            wprintf(L"\tUser comment: %s\n", pBuf4->usri4_usr_comment);
            wprintf(L"\tParameters: %s\n", pBuf4->usri4_parms);
            wprintf(L"\tWorkstations: %s\n", pBuf4->usri4_workstations);
            wprintf
                (L"\tLast logon (seconds since January 1, 1970 GMT): %d\n",
                 pBuf4->usri4_last_logon);
            wprintf
                (L"\tLast logoff (seconds since January 1, 1970 GMT): %d\n",
                 pBuf4->usri4_last_logoff);
            wprintf
                (L"\tAccount expires (seconds since January 1, 1970 GMT): %d\n",
                 pBuf4->usri4_acct_expires);
            wprintf(L"\tMax storage: %d\n", pBuf4->usri4_max_storage);
            wprintf(L"\tUnits per week: %d\n",
                    pBuf4->usri4_units_per_week);
            wprintf(L"\tLogon hours:");
            for (j = 0; j < 21; j++)
            {
                printf(" %x", (BYTE) pBuf4->usri4_logon_hours[j]);
```

```
                    }
                    wprintf(L"\n");
                    wprintf(L"\tBad password count: %d\n",
                            pBuf4->usri4_bad_pw_count);
                    wprintf(L"\tNumber of logons: %d\n",
                            pBuf4->usri4_num_logons);
                    wprintf(L"\tLogon server: %s\n", pBuf4->usri4_logon_server);
                    wprintf(L"\tCountry code: %d\n", pBuf4->usri4_country_code);
                    wprintf(L"\tCode page: %d\n", pBuf4->usri4_code_page);
                    if (ConvertSidToStringSid
                        (pBuf4->usri4_user_sid, &sStringSid))
                    {
                        wprintf(L"\tUser SID: %s\n", sStringSid);
                        LocalFree(sStringSid);
                    }
                    else
                        wprintf(L"ConvertSidToSTringSid failed with error %d\n",
                                GetLastError());
                    wprintf(L"\tPrimary group ID: %d\n",
                            pBuf4->usri4_primary_group_id);
                    wprintf(L"\tProfile: %s\n", pBuf4->usri4_profile);
                    wprintf(L"\tHome directory drive letter: %s\n",
                            pBuf4->usri4_home_dir_drive);
                    wprintf(L"\tPassword expired information: %d\n",
                            pBuf4->usri4_password_expired);
                    break;
                case 10:
                    pBuf10 = (LPUSER_INFO_10) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf10->usri10_name);
                    wprintf(L"\tComment: %s\n", pBuf10->usri10_comment);
                    wprintf(L"\tUser comment: %s\n",
                            pBuf10->usri10_usr_comment);
                    wprintf(L"\tFull name: %s\n", pBuf10->usri10_full_name);
                    break;
                case 11:
                    pBuf11 = (LPUSER_INFO_11) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf11->usri11_name);
                    wprintf(L"\tComment: %s\n", pBuf11->usri11_comment);
                    wprintf(L"\tUser comment: %s\n",
                            pBuf11->usri11_usr_comment);
                    wprintf(L"\tFull name: %s\n", pBuf11->usri11_full_name);
                    wprintf(L"\tPrivilege level: %d\n", pBuf11->usri11_priv);
                    wprintf(L"\tAuth flags (in hex): %x\n",
                            pBuf11->usri11_auth_flags);
                    wprintf(L"\tPassword age (seconds): %d\n",
                            pBuf11->usri11_password_age);
                    wprintf(L"\tHome directory: %s\n", pBuf11->usri11_home_dir);
                    wprintf(L"\tParameters: %s\n", pBuf11->usri11_parms);
                    wprintf
                        (L"\tLast logon (seconds since January 1, 1970 GMT): %d\n",
                         pBuf11->usri11_last_logon);
                    wprintf
                        (L"\tLast logoff (seconds since January 1, 1970 GMT): %d\n",
                         pBuf11->usri11_last_logoff);
                    wprintf(L"\tBad password count: %d\n",
                            pBuf11->usri11_bad_pw_count);
                    wprintf(L"\tNumber of logons: %d\n",
                            pBuf11->usri11_num_logons);
                    wprintf(L"\tLogon server: %s\n",
                            pBuf11->usri11_logon_server);
                    wprintf(L"\tCountry code: %d\n",
                            pBuf11->usri11_country_code);
                    wprintf(L"\tWorkstations: %s\n",
                            pBuf11->usri11_workstations);
                    wprintf(L"\tMax storage: %d\n", pBuf11->usri11_max_storage);
                    wprintf(L"\tUnits per week: %d\n",
                            pBuf11->usri11_units_per_week);
                    wprintf(L"\tLogon hours:");
                    for (j = 0; j < 21; j++)
```

```
                    {
                        printf(" %x", (BYTE) pBuf11->usri11_logon_hours[j]);
                    }
                    wprintf(L"\n");
                    wprintf(L"\tCode page: %d\n", pBuf11->usri11_code_page);
                    break;
                case 20:
                    pBuf20 = (LPUSER_INFO_20) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf20->usri20_name);
                    wprintf(L"\tFull name: %s\n", pBuf20->usri20_full_name);
                    wprintf(L"\tComment: %s\n", pBuf20->usri20_comment);
                    wprintf(L"\tFlags (in hex): %x\n", pBuf20->usri20_flags);
                    wprintf(L"\tUser ID: %u\n", pBuf20->usri20_user_id);
                    break;
                case 23:
                    pBuf23 = (LPUSER_INFO_23) pBuf;
                    wprintf(L"\tUser account name: %s\n", pBuf23->usri23_name);
                    wprintf(L"\tFull name: %s\n", pBuf23->usri23_full_name);
                    wprintf(L"\tComment: %s\n", pBuf23->usri23_comment);
                    wprintf(L"\tFlags (in hex): %x\n", pBuf23->usri23_flags);
                    if (ConvertSidToStringSid
                        (pBuf23->usri23_user_sid, &sStringSid))
                    {
                        wprintf(L"\tUser SID: %s\n", sStringSid);
                        LocalFree(sStringSid);
                    }
                    else
                        wprintf(L"ConvertSidToSTringSid failed with error %d\n",
                                GetLastError());
                    break;
                default:
                    break;
                }
            }
        }
        // Otherwise, print the system error.
        //
        else
            fprintf(stderr, "NetUserGetinfo failed with error: %d\n", nStatus);
        //
        // Free the allocated memory.
        //
        if (pBuf != NULL)
            NetApiBufferFree(pBuf);

        switch (i)
        {
        case 0:
        case 1:
        case 10:
            i++;
            break;
        case 2:
            i = 4;
            break;
        case 4:
            i = 10;
            break;
        case 11:
            i = 20;
            break;
        case 20:
            i = 23;
            break;
        default:
            i = 24;
            break;
        }
    }
```

```
    return 0;
}
```

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserEnum

NetUserGetGroups

NetUserSetInfo

Network Management Functions

Network Management Overview

USER_INFO_0

USER_INFO_1

USER_INFO_10

USER_INFO_11

USER_INFO_2

USER_INFO_23

USER_INFO_24

USER_INFO_4

User Functions

NetUserGetLocalGroups function (lmaccess.h)

2/1/2021 • 6 minutes to read • Edit Online

The **NetUserGetLocalGroups** function retrieves a list of local groups to which a specified user belongs.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserGetLocalGroups(
  LPCWSTR servername,
  LPCWSTR username,
  DWORD   level,
  DWORD   flags,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

A pointer to a constant string that specifies the name of the user for which to return local group membership information. If the string is of the form *DomainName**UserName* the user name is expected to be found on that domain. If the string is of the form *UserName*, the user name is expected to be found on the server specified by the *servername* parameter. For more information, see the Remarks section.

`level`

The information level of the data. This parameter can be the following value.

VALUE	MEANING
0	Return the names of the local groups to which the user belongs. The *bufptr* parameter points to an array of LOCALGROUP_USERS_INFO_0 structures.

`flags`

A bitmask of flags that affect the operation. Currently, only the value defined is **LG_INCLUDE_INDIRECT**. If this bit is set, the function also returns the names of the local groups in which the user is indirectly a member (that is, the user has membership in a global group that is itself a member of one or more local groups).

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with **ERROR_MORE_DATA**.

`prefmaxlen`

The preferred maximum length, in bytes, of the returned data. If **MAX_PREFERRED_LENGTH** is specified in this parameter, the function allocates the amount of memory required for the data. If another value is specified in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns **ERROR_MORE_DATA**. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of entries that could have been enumerated.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access rights to the requested information. This error is also returned if the *servername* parameter has a trailing blank.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter was not specified as 0.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *flags* parameter contains a value other than **LG_INCLUDE_INDIRECT**.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to complete the operation.
NERR_DCNotFound	The domain controller could not be found.
NERR_UserNotFound	The user could not be found. This error is returned if the *username* could not be found.
RPC_S_SERVER_UNAVAILABLE	The RPC server is unavailable. This error is returned if the *servername* parameter could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Domain object is used to perform the access check for this function. The caller must have Read Property permission on the Domain object.

To retrieve a list of global groups to which a specified user belongs, you can call the NetUserGetGroups function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Examples

The following code sample demonstrates how to retrieve a list of the local groups to which a user belongs with a call to the **NetUserGetLocalGroups** function. The sample calls **NetUserGetLocalGroups**, specifying information level 0 (LOCALGROUP_USERS_INFO_0). The sample loops through the entries and prints the name of each local group in which the user has membership. If all available entries are not enumerated, it also prints the number of entries actually enumerated and the total number of entries available. Finally, the code sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t *argv[])
{
   LPLOCALGROUP_USERS_INFO_0 pBuf = NULL;
   DWORD dwLevel = 0;
   DWORD dwFlags = LG_INCLUDE_INDIRECT ;
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   NET_API_STATUS nStatus;

   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
      exit(1);
   }
   //
   // Call the NetUserGetLocalGroups function
   //  specifying information level 0.
   //
   //  The LG_INCLUDE_INDIRECT flag specifies that the
   //   function should also return the names of the local
   //   groups in which the user is indirectly a member.
   //
   nStatus = NetUserGetLocalGroups(argv[1],
                                   argv[2],
                                   dwLevel,
                                   dwFlags,
```

```
                                    dwFlags,
                                    (LPBYTE *) &pBuf,
                                    dwPrefMaxLen,
                                    &dwEntriesRead,
                                    &dwTotalEntries);
   //
   // If the call succeeds,
   //
   if (nStatus == NERR_Success)
   {
      LPLOCALGROUP_USERS_INFO_0 pTmpBuf;
      DWORD i;
      DWORD dwTotalCount = 0;

      if ((pTmpBuf = pBuf) != NULL)
      {
         fprintf(stderr, "\nLocal group(s):\n");
         //
         // Loop through the entries and
         //  print the names of the local groups
         //  to which the user belongs.
         //
         for (i = 0; i < dwEntriesRead; i++)
         {
            assert(pTmpBuf != NULL);

            if (pTmpBuf == NULL)
            {
               fprintf(stderr, "An access violation has occurred\n");
               break;
            }

            wprintf(L"\t-- %s\n", pTmpBuf->lgrui0_name);

            pTmpBuf++;
            dwTotalCount++;
         }
      }
         //
         // If all available entries were
         //  not enumerated, print the number actually
         //  enumerated and the total number available.
         //
      if (dwEntriesRead < dwTotalEntries)
         fprintf(stderr, "\nTotal entries: %d", dwTotalEntries);
      //
      // Otherwise, just print the total.
      //
      printf("\nEntries enumerated: %d\n", dwTotalCount);
   }
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LOCALGROUP_USERS_INFO_0

NetUserGetGroups

Network Management Functions

Network Management Overview

User Functions

NetUserModalsGet function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetUserModalsGet** function retrieves global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserModalsGet(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used. For more information, see the following Remarks section.

`level`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
0	Return global password parameters. The *bufptr* parameter points to a USER_MODALS_INFO_0 structure.
1	Return logon server and domain controller information. The *bufptr* parameter points to a USER_MODALS_INFO_1 structure.
2	Return domain name and identifier. The *bufptr* parameter points to a USER_MODALS_INFO_2 structure. For more information, see the following Remarks section.
3	Return lockout information. The *bufptr* parameter points to a USER_MODALS_INFO_3 structure.

A null session logon can call **NetUserModalsGet** anonymously at information levels 0 and 3.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter.

The buffer for this data is allocated by the system and the application must call the NetApiBufferFree function to

free the allocated memory when the data returned is no longer needed. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_BAD_NETPATH	The network path was not found. This error is returned if the *servername* parameter could not be found.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter is not one of the supported values.
ERROR_INVALID_NAME	The file name, directory name, or volume label syntax is incorrect. This error is returned if the *servername* parameter syntax is incorrect.
ERROR_WRONG_TARGET_NAME	The target account name is incorrect. This error is returned for a logon failure to a remote *servername* parameter running on Windows Vista.
NERR_InvalidComputer	The computer name is invalid.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user modal functions. For more information, see IADsDomain.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Domain object is used to perform the access check for this function.

To retrieve the security identifier (SID) of the domain to which the computer belongs, call the **NetUserModalsGet** function specifying a USER_MODALS_INFO_2 structure and **NULL** in the *servername* parameter. If the computer isn't a member of a domain, the function returns a **NULL** pointer.

Examples

The following code sample demonstrates how to retrieve global information for all users and global groups with a call to the **NetUserModalsGet** function. The sample calls **NetUserModalsGet**, specifying information level 0 (USER_MODALS_INFO_0). If the call succeeds, the sample prints global password information. Finally, the code

sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 0;
   USER_MODALS_INFO_0 *pBuf = NULL;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Call the NetUserModalsGet function; specify level 0.
   //
   nStatus = NetUserModalsGet((LPCWSTR) pszServerName,
                              dwLevel,
                              (LPBYTE *)&pBuf);
   //
   // If the call succeeds, print the global information.
   //
   if (nStatus == NERR_Success)
   {
      if (pBuf != NULL)
      {
         printf("\tMinimum password length:  %d\n", pBuf->usrmod0_min_passwd_len);
         printf("\tMaximum password age (d): %d\n", pBuf->usrmod0_max_passwd_age/86400);
         printf("\tMinimum password age (d): %d\n", pBuf->usrmod0_min_passwd_age/86400);
         printf("\tForced log off time (s):  %d\n", pBuf->usrmod0_force_logoff);
         printf("\tPassword history length:  %d\n", pBuf->usrmod0_password_hist_len);
      }
   }
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserModalsSet

Network Management Functions

Network Management Overview

USER_MODALS_INFO_0

USER_MODALS_INFO_1

USER_MODALS_INFO_2

USER_MODALS_INFO_3

User Modals Functions

NetUserModalsSet function (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetUserModalsSet** function sets global information for all users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserModalsSet(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies global password parameters. The *buf* parameter points to a USER_MODALS_INFO_0 structure.
1	Specifies logon server and domain controller information. The *buf* parameter points to a USER_MODALS_INFO_1 structure.
2	Specifies the domain name and identifier. The *buf* parameter points to a USER_MODALS_INFO_2 structure.
3	Specifies lockout information. The *buf* parameter points to a USER_MODALS_INFO_3 structure.
1001	Specifies the minimum allowable password length. The *buf* parameter points to a USER_MODALS_INFO_1001 structure.
1002	Specifies the maximum allowable password age. The *buf* parameter points to a USER_MODALS_INFO_1002 structure.
1003	Specifies the minimum allowable password age. The *buf* parameter points to a USER_MODALS_INFO_1003 structure.

1004	Specifies forced logoff information. The *buf* parameter points to a USER_MODALS_INFO_1004 structure.
1005	Specifies the length of the password history. The *buf* parameter points to a USER_MODALS_INFO_1005 structure.
1006	Specifies the role of the logon server. The *buf* parameter points to a USER_MODALS_INFO_1006 structure.
1007	Specifies domain controller information. The *buf* parameter points to a USER_MODALS_INFO_1007 structure.

`buf`

Pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first member of the information structure that causes ERROR_INVALID_PARAMETER. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	The specified parameter is invalid. For more information, see the following Remarks section.
NERR_InvalidComputer	The computer name is invalid.
NERR_UserNotFound	The user name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user modal functions. For more information, see IADsDomain.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management

Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Domain object is used to perform the access check for this function. Typically, callers must have write access to the entire object for calls to this function to succeed.

If the **NetUserModalsSet** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the first member of the information structure that is invalid. (The information structure begins with USER_MODALS_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix usrmod*_ indicates that the member can begin with multiple prefixes, for example, usrmod2_ or usrmod1002_.)

VALUE	MEMBER
MODALS_MIN_PASSWD_LEN_PARMNUM	usrmod*_min_passwd_len
MODALS_MAX_PASSWD_AGE_PARMNUM	usrmod*_max_passwd_age
MODALS_MIN_PASSWD_AGE_PARMNUM	usrmod*_min_passwd_age
MODALS_FORCE_LOGOFF_PARMNUM	usrmod*_force_logoff
MODALS_PASSWD_HIST_LEN_PARMNUM	usrmod*_password_hist_len
MODALS_ROLE_PARMNUM	usrmod*_role
MODALS_PRIMARY_PARMNUM	usrmod*_primary
MODALS_DOMAIN_NAME_PARMNUM	usrmod*_domain_name
MODALS_DOMAIN_ID_PARMNUM	usrmod*_domain_id
MODALS_LOCKOUT_DURATION_PARMNUM	usrmod*_lockout_duration
MODALS_LOCKOUT_OBSERVATION_WINDOW_PARMNUM	usrmod*_lockout_observation_window
MODALS_LOCKOUT_THRESHOLD_PARMNUM	usrmod*_lockout_threshold

Examples

The following code sample demonstrates how to set the global information for all users and global groups with a call to the **NetUserModalsSet** function. The sample fills in the members of the USER_MODALS_INFO_0 structure and calls **NetUserModalsSet**, specifying information level 0.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 0;
   USER_MODALS_INFO_0 ui;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Fill in the USER_MODALS_INFO_0 structure.
   //
   ui.usrmod0_min_passwd_len = 0;
   ui.usrmod0_max_passwd_age = (86400 * 30);
   ui.usrmod0_min_passwd_age = 0;
   ui.usrmod0_force_logoff = TIMEQ_FOREVER; // never force logoff
   ui.usrmod0_password_hist_len = 0;
   //
   // Call the NetUserModalsSet function; specify level 0.
   //
   nStatus = NetUserModalsSet((LPCWSTR) pszServerName,
                              dwLevel,
                              (LPBYTE)&ui,
                              NULL);
   //
   // If the call succeeds, inform the user.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"Modals information set successfully on %s\n", argv[1]);
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserModalsGet

Network Management Functions

Network Management Overview

USER_MODALS_INFO_0

USER_MODALS_INFO_1

USER_MODALS_INFO_1001

USER_MODALS_INFO_1002

USER_MODALS_INFO_1003

USER_MODALS_INFO_1004

USER_MODALS_INFO_1005

USER_MODALS_INFO_1006

USER_MODALS_INFO_1007

USER_MODALS_INFO_2

USER_MODALS_INFO_3

User Modals Functions

NetUserSetGroups function (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetUserSetGroups** function sets global group memberships for a specified user account.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserSetGroups(
  LPCWSTR servername,
  LPCWSTR username,
  DWORD   level,
  LPBYTE  buf,
  DWORD   num_entries
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

A pointer to a constant string that specifies the name of the user for which to set global group memberships. For more information, see the Remarks section.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	The *buf* parameter points to an array of GROUP_USERS_INFO_0 structures that specifies global group names.
1	The *buf* parameter points to an array of GROUP_USERS_INFO_1 structures that specifies global group names with attributes.

`buf`

A pointer to the buffer that specifies the data. For more information, see Network Management Function Buffers.

`num_entries`

The number of entries contained in the array pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The system call level is not correct. This error is returned if the *level* parameter was specified as a value other than 0 or 1.
ERROR_INVALID_PARAMETER	A parameter passed was not valid. This error is returned if the *num_entries* parameter was not valid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to complete the operation.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_GroupNotFound	The group group name specified by the **grui0_name** in the GROUP_USERS_INFO_0 structure or **grui1_name** member in the GROUP_USERS_INFO_1 structure pointed to by the *buf* parameter does not exist.
NERR_InternalError	An internal error occurred.
NERR_UserNotFound	The user name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function.

To grant a user membership in one existing global group, you can call the NetGroupAddUser function.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Examples

The following code sample demonstrates how to set global group memberships for a user account with a call to the **NetUserSetGroups** function. The code sample fills in the **grui0_name** member of the GROUP_USERS_INFO_0 structure and calls **NetUserSetGroups**, specifying information level 0.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 0;
   GROUP_USERS_INFO_0 gi;
   NET_API_STATUS nStatus;

   if (argc != 4)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName GroupName\n", argv[0]);
      exit(1);
   }
   //
   // Fill in the GROUP_USERS_INFO_0 structure member.
   //
   gi.grui0_name = argv[3];
   //
   // Call the NetUserSetGroups function; specify level 0.
   //
   nStatus = NetUserSetGroups(argv[1],
                              argv[2],
                              dwLevel,
                              (LPBYTE)&gi,
                              1);
   //
   // If the call succeeds, inform the user.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"Group membership has been successful for %s\n", argv[2]);
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows

Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

GROUP_USERS_INFO_0

GROUP_USERS_INFO_1

NetGroupAddUser

NetUserGetGroups

Network Management Functions

Network Management Overview

User Functions

NetUserSetInfo function (lmaccess.h)

2/1/2021 • 9 minutes to read • Edit Online

The **NetUserSetInfo** function sets the parameters of a user account.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUserSetInfo(
  LPCWSTR servername,
  LPCWSTR username,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`username`

A pointer to a constant string that specifies the name of the user account for which to set information. For more information, see the following Remarks section.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies the user account name. The *buf* parameter points to a USER_INFO_0 structure. Use this structure to specify a new group name. For more information, see the following Remarks section.
1	Specifies detailed information about the user account. The *buf* parameter points to a USER_INFO_1 structure.
2	Specifies level one information and additional attributes about the user account. The *buf* parameter points to a USER_INFO_2 structure.
3	Specifies level two information and additional attributes about the user account. This level is valid only on servers. The *buf* parameter points to a USER_INFO_3 structure. Note that it is recommended that you use USER_INFO_4 instead.
4	Specifies level two information and additional attributes about the user account. This level is valid only on servers. The *buf* parameter points to a USER_INFO_4 structure.

21	Specifies a one-way encrypted LAN Manager 2.*x*-compatible password. The *buf* parameter points to a USER_INFO_21 structure.
22	Specifies detailed information about the user account. The *buf* parameter points to a USER_INFO_22 structure.
1003	Specifies a user password. The *buf* parameter points to a USER_INFO_1003 structure.
1005	Specifies a user privilege level. The *buf* parameter points to a USER_INFO_1005 structure.
1006	Specifies the path of the home directory for the user. The *buf* parameter points to a USER_INFO_1006 structure.
1007	Specifies a comment to associate with the user account. The *buf* parameter points to a USER_INFO_1007 structure.
1008	Specifies user account attributes. The *buf* parameter points to a USER_INFO_1008 structure.
1009	Specifies the path for the user's logon script file. The *buf* parameter points to a USER_INFO_1009 structure.
1010	Specifies the user's operator privileges. The *buf* parameter points to a USER_INFO_1010 structure.
1011	Specifies the full name of the user. The *buf* parameter points to a USER_INFO_1011 structure.
1012	Specifies a comment to associate with the user. The *buf* parameter points to a USER_INFO_1012 structure.
1014	Specifies the names of workstations from which the user can log on. The *buf* parameter points to a USER_INFO_1014 structure.
1017	Specifies when the user account expires. The *buf* parameter points to a USER_INFO_1017 structure.
1020	Specifies the times during which the user can log on. The *buf* parameter points to a USER_INFO_1020 structure.
1024	Specifies the user's country/region code. The *buf* parameter points to a USER_INFO_1024 structure.
1051	Specifies the relative identifier of a global group that represents the enrolled user. The *buf* parameter points to a USER_INFO_1051 structure.

1052	Specifies the path to a network user's profile. The *buf* parameter points to a USER_INFO_1052 structure.
1053	Specifies the drive letter assigned to the user's home directory. The *buf* parameter points to a USER_INFO_1053 structure.

`buf`

A pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

A pointer to a value that receives the index of the first member of the user information structure that causes ERROR_INVALID_PARAMETER. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid. For more information, see the following Remarks section.
NERR_InvalidComputer	The computer name is invalid.
NERR_NotPrimary	The operation is allowed only on the primary domain controller of the domain.
NERR_SpeGroupOp	The operation is not allowed on specified special groups, which are user groups, admin groups, local groups, or guest groups.
NERR_LastAdmin	The operation is not allowed on the last administrative account.
NERR_BadPassword	The share name or password is invalid.
NERR_PasswordTooShort	The password is shorter than required. (The password could also be too long, be too recent in its change history, not have enough unique characters, or not meet another password policy requirement.)

NERR_UserNotFound	The user name could not be found.

Remarks

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management user functions. For more information, see IADsUser and IADsComputer.

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Power Users can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the User object is used to perform the access check for this function.

Only users or applications having administrative privileges can call the **NetUserSetInfo** function to change a user's password. When an administrator calls **NetUserSetInfo**, the only restriction applied is that the new password length must be consistent with system modals. A user or application that knows a user's current password can call the NetUserChangePassword function to change the password. For more information about calling functions that require administrator privileges, see Running with Special Privileges.

Members of the Administrators local group can set any modifiable user account elements. All users can set the **usri2_country_code** member of the USER_INFO_2 structure (and the **usri1024_country_code** member of the USER_INFO_1024 structure) for their own accounts.

A member of the Account Operator's local group cannot set details for an Administrators class account, give an existing account Administrator privilege, or change the operator privilege of any account. If you attempt to change the privilege level or disable the last account with Administrator privilege in the security database, (the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory), the **NetUserSetInfo** function fails and returns NERR_LastAdmin.

To set the following user account control flags, the following privileges and control access rights are required.

ACCOUNT CONTROL FLAG	PRIVILEGE OR RIGHT REQUIRED
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	SeEnableDelegationPrivilege privilege, which is granted to Administrators by default.
UF_TRUSTED_FOR_DELEGATION	SeEnableDelegationPrivilege.
UF_PASSWD_NOTREQD	"Update password not required" control access right on the Domain object, which is granted to authenticated users by default.
UF_DONT_EXPIRE_PASSWD	"Unexpire password" control access right on the Domain object, which is granted to authenticated users by default.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	"Enable per user reversibly encrypted password" control access right on the Domain object, which is granted to authenticated users by default.

UF_SERVER_TRUST_ACCOUNT	"Add/remove replica in domain" control access right on the Domain object, which is granted to Administrators by default.

For a list of privilege constants, see Authorization Constants.

The correct way to specify the new name for an account is to call **NetUserSetInfo** with USER_INFO_0 and to specify the new value using the **usri0_name** member. If you call **NetUserSetInfo** with other information levels and specify a value using a **usriX_name** member, the value is ignored.

Note that calls to **NetUserSetInfo** can change the home directory only for user accounts that the network server creates.

If the **NetUserSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the first member of the user information structure that is invalid. (A user information structure begins with USER_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix usri*_ indicates that the member can begin with multiple prefixes, for example, usri10_ or usri1003_.)

VALUE	MEMBER
USER_NAME_PARMNUM	usri*_name
USER_PASSWORD_PARMNUM	usri*_password
USER_PASSWORD_AGE_PARMNUM	usri*_password_age
USER_PRIV_PARMNUM	usri*_priv
USER_HOME_DIR_PARMNUM	usri*_home_dir
USER_COMMENT_PARMNUM	usri*_comment
USER_FLAGS_PARMNUM	usri*_flags
USER_SCRIPT_PATH_PARMNUM	usri*_script_path
USER_AUTH_FLAGS_PARMNUM	usri*_auth_flags
USER_FULL_NAME_PARMNUM	usri*_full_name
USER_USR_COMMENT_PARMNUM	usri*_usr_comment
USER_PARMS_PARMNUM	usri*_parms
USER_WORKSTATIONS_PARMNUM	usri*_workstations
USER_LAST_LOGON_PARMNUM	usri*_last_logon
USER_LAST_LOGOFF_PARMNUM	usri*_last_logoff
USER_ACCT_EXPIRES_PARMNUM	usri*_acct_expires

USER_MAX_STORAGE_PARMNUM	usri*_max_storage
USER_UNITS_PER_WEEK_PARMNUM	usri*_units_per_week
USER_LOGON_HOURS_PARMNUM	usri*_logon_hours
USER_PAD_PW_COUNT_PARMNUM	usri*_bad_pw_count
USER_NUM_LOGONS_PARMNUM	usri*_num_logons
USER_LOGON_SERVER_PARMNUM	usri*_logon_server
USER_COUNTRY_CODE_PARMNUM	usri*_country_code
USER_CODE_PAGE_PARMNUM	usri*_code_page
USER_PRIMARY_GROUP_PARMNUM	usri*_primary_group_id
USER_PROFILE_PARMNUM	usri*_profile
USER_HOME_DIR_DRIVE_PARMNUM	usri*_home_dir_drive

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

The **NetUserSetInfo** function does not control how the password parameters are secured when sent over the network to a remote server to change a user password. Any encryption of these parameters is handled by the Remote Procedure Call (RPC) mechanism supported by the network redirector that provides the network transport. Encryption is also controlled by the security mechanisms supported by the local computer and the security mechanisms supported by remote network server specified in the *servername* parameter. For more details on security when the Microsoft network redirector is used and the remote network server is running Microsoft Windows, see the protocol documentation for MS-RPCE and MS-SAMR.

Examples

The following code sample demonstrates how to disable a user account with a call to the **NetUserSetInfo** function. The code sample fills in the **usri1008_flags** member of the USER_INFO_1008 structure, specifying the value UF_ACCOUNTDISABLE. Then the sample calls **NetUserSetInfo**, specifying information level 0.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 1008;
   USER_INFO_1008 ui;
   NET_API_STATUS nStatus;

   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName UserName\n", argv[0]);
      exit(1);
   }
   // Fill in the USER_INFO_1008 structure member.
   // UF_SCRIPT: required.
   //
   ui.usri1008_flags = UF_SCRIPT | UF_ACCOUNTDISABLE;
   //
   // Call the NetUserSetInfo function
   //  to disable the account, specifying level 1008.
   //
   nStatus = NetUserSetInfo(argv[1],
                            argv[2],
                            dwLevel,
                            (LPBYTE)&ui,
                            NULL);
   //
   // Display the result of the call.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"User account %s has been disabled\n", argv[2]);
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUserGetInfo

Network Management Functions

Network Management Overview

USER_INFO_0

USER_INFO_1

USER_INFO_1003

USER_INFO_1005

USER_INFO_1006

USER_INFO_1007

USER_INFO_1008

USER_INFO_1009

USER_INFO_1010

USER_INFO_1011

USER_INFO_1012

USER_INFO_1013

USER_INFO_1014

USER_INFO_1017

USER_INFO_1020

USER_INFO_1024

USER_INFO_1051

USER_INFO_1052

USER_INFO_1053

USER_INFO_2

USER_INFO_21

USER_INFO_22

USER_INFO_4

User Functions

NetValidatePasswordPolicy function (lmaccess.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetValidatePasswordPolicy** function allows an application to check password compliance against an application-provided account database and verify that passwords meet the complexity, aging, minimum length, and history reuse requirements of a password policy.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetValidatePasswordPolicy(
  LPCWSTR                     ServerName,
  LPVOID                      Qualifier,
  NET_VALIDATE_PASSWORD_TYPE ValidationType,
  LPVOID                      InputArg,
  LPVOID                      *OutputArg
);
```

Parameters

`ServerName`

A pointer to a constant Unicode string specifying the name of the remote server on which the function is to execute. This string must begin with \ followed by the remote server name. If this parameter is **NULL**, the local computer is used.

`Qualifier`

Reserved for future use. This parameter must be **NULL**.

`ValidationType`

The type of password validation to perform. This parameter must be one of the following enumerated constant values.

```
typedef enum _NET_VALIDATE_PASSWORD_TYPE {

    NetValidateAuthentication = 1,
    NetValidatePasswordChange,
    NetValidatePasswordReset,

} NET_VALIDATE_PASSWORD_TYPE, *PNET_VALIDATE_PASSWORD_TYPE;
```

These values have the following meanings.

VALUE	MEANING
NetValidateAuthentication	The application is requesting password validation during authentication. The *InputArg* parameter points to a NET_VALIDATE_AUTHENTICATION_INPUT_ARG structure. This type of validation enforces password expiration and account lockout policy.

NetValidatePasswordChange	The application is requesting password validation during a password change operation. The *InputArg* parameter points to a NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG structure.
NetValidatePasswordReset	The application is requesting password validation during a password reset operation. The *InputArg* parameter points to a NET_VALIDATE_PASSWORD_RESET_INPUT_ARG structure. You can also reset the "lockout state" of a user account by specifying this structure.

`InputArg`

A pointer to a structure that depends on the type of password validation to perform. The type of structure depends on the value of the *ValidationType* parameter. For more information, see the description of the *ValidationType* parameter.

`OutputArg`

If the **NetValidatePasswordPolicy** function succeeds (the return value is **Nerr_Success**), then the function allocates an buffer that contains the results of the operation. The *OutputArg* parameter contains a pointer to a NET_VALIDATE_OUTPUT_ARG structure. The application must examine **ValidationStatus** member in the **NET_VALIDATE_OUTPUT_ARG** structure pointed to by the *OutputArg* parameter to determine the results of the password policy validation check. The **NET_VALIDATE_OUTPUT_ARG** structure contains a NET_VALIDATE_PERSISTED_FIELDS structure with changes to persistent password-related information, and the results of the password validation. The application must plan to persist all persisted the fields in the **NET_VALIDATE_PERSISTED_FIELDS** structure aside from the **ValidationStatus**member as information along with the user object information and provide the required fields from the persisted information when calling this function in the future on the same user object.

If the **NetValidatePasswordPolicy** function fails (the return value is nonzero), then *OutputArg* parameter is set to a **NULL** pointer and password policy could not be examined.

For more information, see the Return Values and Remarks sections.

Return value

If the function succeeds, and the password is authenticated, changed, or reset, the return value is NERR_Success and the function allocates an *OutputArg* parameter.

If the function fails, the *OutputArg* parameter is **NULL** and the return value is a system error code that can be one of the following error codes. For a list of all possible error codes, see System Error Codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *InputArg* or *OutputArg* parameters are **NULL**. This error is also returned if the *Qualifier* parameter is not **NULL** or if the *ValidationType* parameter is not one of the allowed values.
ERROR_NOT_ENOUGH_MEMORY	Not enough memory is available to complete the operation.

Remarks

The **NetValidatePasswordPolicy** function is designed to allow applications to validate passwords for users

that are in an account database provided by the application. This function can also be used to verify that passwords meet the complexity, aging, minimum length, and history reuse requirements of a password policy. This function also provides the means for an application to implement an account-lockout mechanism.

The **NetValidatePasswordPolicy** function does not validate passwords in Active Directory accounts and cannot be used for this purpose. The only policy that this function checks a password against in Active Directory accounts is the password complexity (the password strength).

A typical scenario for the use of the **NetValidatePasswordPolicy** function would be enforcing the choice of strong passwords by users for web applications and applications that allow password-protected documents. Another use of this function could be checking password complexity in a situation in which a password is attached to a functional operation rather than to a user account; for example, passwords that are used with Secure Multipurpose Internet Mail Extensions (S/MIME) certificate-based public keys.

If the **NetValidatePasswordPolicy** function is called on a domain controller that is running Active Directory, access is allowed or denied based on the ACL for the securable object. The default ACL permits all authenticated users and members of the "Pre-Windows 2000 compatible access" group to view the information. If you call this function on a member server or workstation, all authenticated users can view the information. For information about anonymous access and restricting anonymous access on these platforms, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

The security descriptor of the Domain object is used to perform the access check for the **NetValidatePasswordPolicy** function.

To call **NetValidatePasswordPolicy** in a security context that is not the default, first call the LogonUser function, specifying LOGON32_LOGON_NEW_CREDENTIALS in the *dwLogonType* parameter, and then call **NetValidatePasswordPolicy** under impersonation. For more information about impersonation, see Client Impersonation.

If the return code of the **NetValidatePasswordPolicy** function is **Nerr_Success** then the function allocates a buffer pointed to by the *OutputArg* parameter that contains a NET_VALIDATE_OUTPUT_ARG structure with the results of the operation. The application must examine **ValidationStatus** member in the **NET_VALIDATE_OUTPUT_ARG** structure to determine the results of the password policy validation check. For more information, see **NET_VALIDATE_OUTPUT_ARG**.

Note that it is the application's responsibility to save all the data in the **ChangedPersistedFields** member of the **NET_VALIDATE_OUTPUT_ARG** structure as well as any User object information. The next time the application calls **NetValidatePasswordPolicy** on the same instance of the User object, the application must provide the required fields from the persistent information.

When you call **NetValidatePasswordPolicy** and specify NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG or NET_VALIDATE_PASSWORD_RESET_INPUT_ARG in *InputArg* parameter, the call also validates the password by passing it through the password filter DLL that the computer is configured to use. For more information about password filters, see Using Password Filters.

If the return value from the **NetValidatePasswordPolicy** function is nonzero then *OutputArg* parameter is set to **NULL** and password policy could not be examined.

The NetValidatePasswordPolicyFree function should be called after calling **NetValidatePasswordPolicy** to free the memory allocated for the *OutputArg* parameter that is returned by the call to the **NetValidatePasswordPolicy** function.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

LogonUser

NET_VALIDATE_AUTHENTICATION_INPUT_ARG

NET_VALIDATE_OUTPUT_ARG

NET_VALIDATE_PASSWORD_CHANGE_INPUT_ARG

NET_VALIDATE_PASSWORD_RESET_INPUT_ARG

NET_VALIDATE_PERSISTED_FIELDS

NetValidatePasswordPolicyFree

Network Management Functions

Network Management Overview

NetValidatePasswordPolicyFree function (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetValidatePasswordPolicyFree** function frees the memory that the NetValidatePasswordPolicy function allocates for the *OutputArg* parameter, which is a NET_VALIDATE_OUTPUT_ARG structure.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetValidatePasswordPolicyFree(
  LPVOID *OutputArg
);
```

Parameters

`OutputArg`

Pointer to the memory allocated for the *OutputArg* parameter by a call to the **NetValidatePasswordPolicy** function.

Return value

If the function frees the memory, or if there is no memory to free from a previous call to **NetValidatePasswordPolicy**, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to successfully execute this function.

Requirements

Minimum supported client	None supported
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	lmaccess.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetValidatePasswordPolicy

Network Management Functions

Network Management Overview

USER_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_0** structure contains a user account name.

Syntax

```
typedef struct _USER_INFO_0 {
  LPWSTR usri0_name;
} USER_INFO_0, *PUSER_INFO_0, *LPUSER_INFO_0;
```

Members

`usri0_name`

Pointer to a Unicode string that specifies the name of the user account. For the NetUserSetInfo function, this member specifies the name of the user.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUseDel

NetUserEnum

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1 structure (lmaccess.h)

2/1/2021 • 4 minutes to read • Edit Online

The **USER_INFO_1** structure contains information about a user account, including account name, password data, privilege level, and the path to the user's home directory.

Syntax

```
typedef struct _USER_INFO_1 {
  LPWSTR usri1_name;
  LPWSTR usri1_password;
  DWORD  usri1_password_age;
  DWORD  usri1_priv;
  LPWSTR usri1_home_dir;
  LPWSTR usri1_comment;
  DWORD  usri1_flags;
  LPWSTR usri1_script_path;
} USER_INFO_1, *PUSER_INFO_1, *LPUSER_INFO_1;
```

Members

`usri1_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. For the NetUserSetInfo function, this member is ignored. For more information, see the following Remarks section.

`usri1_password`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the password of the user indicated by the **usri1_name** member. The length cannot exceed PWLEN bytes. The NetUserEnum and NetUserGetInfo functions return a **NULL** pointer to maintain password security.

By convention, the length of passwords is limited to LM20_PWLEN characters.

`usri1_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri1_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri1_priv`

Type: **DWORD**

The level of privilege assigned to the **usri1_name** member. When you call the NetUserAdd function, this member must be USER_PRIV_USER. When you call the NetUserSetInfo function, this member must be the value returned by the **NetUserGetInfo** function or the **NetUserEnum** function. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING

USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri1_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory for the user specified in the **usri1_name** member. The string can be **NULL**.

`usri1_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment to associate with the user account. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri1_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function and clear this value to unlock a previously locked account. You cannot use this value to lock a previously unlocked account.

UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows 2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.

UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri1_script_path`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be **NULL**.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_10 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_10** structure contains information about a user account, including the account name, comments associated with the account, and the user's full name.

Syntax

```
typedef struct _USER_INFO_10 {
  LPWSTR usri10_name;
  LPWSTR usri10_comment;
  LPWSTR usri10_usr_comment;
  LPWSTR usri10_full_name;
} USER_INFO_10, *PUSER_INFO_10, *LPUSER_INFO_10;
```

Members

`usri10_name`

Pointer to a Unicode string that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member. For more information, see the following Remarks section.

`usri10_comment`

Pointer to a Unicode string that contains a comment associated with the user account. The string can be a null string, or can have any number of characters before the terminating null character.

`usri10_usr_comment`

Pointer to a Unicode string that contains a user comment. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri10_full_name`

Pointer to a Unicode string that contains the full name of the user. This string can be a null string, or it can have any number of characters before the terminating null character.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmaccess.h (include Lm.h)

See also

NetUserDel

NetUserEnum

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1003 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1003** structure contains a user password. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1003 {
  LPWSTR usri1003_password;
} USER_INFO_1003, *PUSER_INFO_1003, *LPUSER_INFO_1003;
```

Members

`usri1003_password`

Specifies a Unicode string that contains the password for the user account specified in the *username* parameter to the **NetUserSetInfo** function. The length cannot exceed PWLEN bytes.

Remarks

By convention, the length of passwords is limited to LM20_PWLEN characters.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1005 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1005** structure contains a privilege level to assign to a user network account. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1005 {
  DWORD usri1005_priv;
} USER_INFO_1005, *PUSER_INFO_1005, *LPUSER_INFO_1005;
```

Members

`usri1005_priv`

Specifies a **DWORD** value that indicates the level of privilege to assign for the user account specified in the *username* parameter to the **NetUserSetInfo** function. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

USER_INFO_1006 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1006** structure contains the user's home directory path. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1006 {
  LPWSTR usri1006_home_dir;
} USER_INFO_1006, *PUSER_INFO_1006, *LPUSER_INFO_1006;
```

Members

`usri1006_home_dir`

Pointer to a Unicode string specifying the path of the home directory for the user account specified in the *username* parameter to the **NetUserSetInfo** function. The string can be null.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1007 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1007** structure contains a comment associated with a user network account. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1007 {
  LPWSTR usri1007_comment;
} USER_INFO_1007, *PUSER_INFO_1007, *LPUSER_INFO_1007;
```

Members

`usri1007_comment`

Pointer to a Unicode string that contains a comment to associate with the user account specified in the *username* parameter to the **NetUserSetInfo** function. This string can be a null string, or it can have any number of characters before the terminating null character.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1008 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1008** structure contains a set of bit flags defining several user network account parameters. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1008 {
  DWORD usri1008_flags;
} USER_INFO_1008, *PUSER_INFO_1008, *LPUSER_INFO_1008;
```

Members

`usri1008_flags`

The features to associate with the user account specified in the *username* parameter to the **NetUserSetInfo** function. This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.

UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.

UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1009 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1009** structure contains the path for a user's logon script file. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1009 {
  LPWSTR usri1009_script_path;
} USER_INFO_1009, *PUSER_INFO_1009, *LPUSER_INFO_1009;
```

Members

`usri1009_script_path`

Pointer to a Unicode string specifying the path for the user's logon script file. The user is specified in the *username* parameter to the **NetUserSetInfo** function. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be null.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1010 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1010** structure contains a set of bit flags defining the operator privileges assigned to a user network account. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1010 {
  DWORD usri1010_auth_flags;
} USER_INFO_1010, *PUSER_INFO_1010, *LPUSER_INFO_1010;
```

Members

`usri1010_auth_flags`

Specifies a **DWORD** value that contains a set of bit flags that specify the user's operator privileges. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.
AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

Remarks

For more information about controlling access to securable objects, see Access Control, Privileges, and Securable Objects.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1011 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1011** structure contains the full name of a network user. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1011 {
  LPWSTR usri1011_full_name;
} USER_INFO_1011, *PUSER_INFO_1011, *LPUSER_INFO_1011;
```

Members

`usri1011_full_name`

Pointer to a Unicode string that contains the full name of the user. The user is specified in the *username* parameter to the **NetUserSetInfo** function. This string can be a null string, or it can have any number of characters before the terminating null character.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1012 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1012** structure contains a user comment. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1012 {
  LPWSTR usri1012_usr_comment;
} USER_INFO_1012, *PUSER_INFO_1012, *LPUSER_INFO_1012;
```

Members

`usri1012_usr_comment`

Pointer to a Unicode string that contains a user comment. The user is specified in the *username* parameter to the **NetUserSetInfo** function. This string can be a null string, or it can have any number of characters before the terminating null character.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1013 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1013** structure contains reserved information for network accounts. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1013 {
  LPWSTR usri1013_parms;
} USER_INFO_1013, *PUSER_INFO_1013, *LPUSER_INFO_1013;
```

Members

`usri1013_parms`

Pointer to a Unicode string that is reserved for use by applications. The string can be a null string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

The system components that use this member are services for Macintosh, file and print services for NetWare, and the Remote Access Server (RAS).

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1014 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1014** structure contains the names of workstations from which the user can log on. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1014 {
  LPWSTR usri1014_workstations;
} USER_INFO_1014, *PUSER_INFO_1014, *LPUSER_INFO_1014;
```

Members

`usri1014_workstations`

> **IMPORTANT**
>
> You should no longer use **usri1014_workstations**. Instead, you can control sign-in access to workstations by configuring the User Rights Assignment settings (**Allow log on locally** and **Deny log on locally**, or **Allow log on through Remote Desktop Services** and **Deny log on through Remote Desktop Services**).

Pointer to a Unicode string that contains the names of workstations from which the user can log on. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

As many as eight workstations can be specified; the names must be separated by commas. A null string indicates that there is no restriction.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1017 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1017** structure contains expiration information for network user accounts. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1017 {
  DWORD usri1017_acct_expires;
} USER_INFO_1017, *PUSER_INFO_1017, *LPUSER_INFO_1017;
```

Members

`usri1017_acct_expires`

Specifies a **DWORD** value that indicates when the user account expires. The user account is specified in the *username* parameter to the **NetUserSetInfo** function.

The value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. Specify TIMEQ_FOREVER to indicate that the account never expires.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1018 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1018** structure contains the maximum amount of disk space available to a network user account. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1018 {
  DWORD usri1018_max_storage;
} USER_INFO_1018, *PUSER_INFO_1018, *LPUSER_INFO_1018;
```

Members

`usri1018_max_storage`

Specifies a **DWORD** value that indicates the maximum amount of disk space the user can use. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

You must specify USER_MAXSTORAGE_UNLIMITED to indicate that there is no restriction on disk space.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1020 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1020** structure contains the times during which a user can log on to the network. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1020 {
  DWORD  usri1020_units_per_week;
  LPBYTE usri1020_logon_hours;
} USER_INFO_1020, *PUSER_INFO_1020, *LPUSER_INFO_1020;
```

Members

`usri1020_units_per_week`

Specifies a **DWORD** value that indicates the number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri1020_logon_hours** member.

This value must be UNITS_PER_WEEK for LAN Manager 2.0. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

For service applications, the units must be one of the following values: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri1020_logon_hours`

Pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

Each bit in the string represents a unique hour in the week, in Greenwich Mean Time (GMT). The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

USER_INFO_1023 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1023** structure contains the name of the server to which network logon requests should be sent. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1023 {
  LPWSTR usri1023_logon_server;
} USER_INFO_1023, *PUSER_INFO_1023, *LPUSER_INFO_1023;
```

Members

`usri1023_logon_server`

Pointer to a Unicode string that contains the name of the server to which logon requests for the user account should be sent. The user account is specified in the *username* parameter to the **NetUserSetInfo** function.

Server names should be preceded by two backslashes (\\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A null string indicates that requests should be sent to the domain controller.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1024 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1024** structure contains the country/region code for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1024 {
  DWORD usri1024_country_code;
} USER_INFO_1024, *PUSER_INFO_1024, *LPUSER_INFO_1024;
```

Members

`usri1024_country_code`

Specifies a **DWORD** value that indicates the country/region code for the user's language of choice. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

This value is ignored.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1025 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1025** structure contains the code page for a network user's language of choice. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1025 {
  DWORD usri1025_code_page;
} USER_INFO_1025, *PUSER_INFO_1025, *LPUSER_INFO_1025;
```

Members

`usri1025_code_page`

Specifies a **DWORD** value that indicates the code page for the user's language of choice. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

This value is ignored.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1051 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1051** structure contains the relative ID (RID) associated with the user account. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1051 {
  DWORD usri1051_primary_group_id;
} USER_INFO_1051, *PUSER_INFO_1051, *LPUSER_INFO_1051;
```

Members

`usri1051_primary_group_id`

Specifies a **DWORD** value that contains the RID of the Primary Global Group for the user specified in the *username* parameter to the **NetUserSetInfo** function. This member must be the RID of a global group that represents the enrolled user. For more information about RIDs, see SID Components.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1052 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1052** structure contains the path to a network user's profile. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1052 {
  LPWSTR usri1052_profile;
} USER_INFO_1052, *PUSER_INFO_1052, *LPUSER_INFO_1052;
```

Members

`usri1052_profile`

Specifies a Unicode string that contains the path to the user's profile. The user is specified in the *username* parameter to the **NetUserSetInfo** function. This value can be a null string, a local absolute path, or a UNC path.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_1053 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_1053** structure contains user information for network accounts. This information level is valid only when you call the NetUserSetInfo function.

Syntax

```
typedef struct _USER_INFO_1053 {
  LPWSTR usri1053_home_dir_drive;
} USER_INFO_1053, *PUSER_INFO_1053, *LPUSER_INFO_1053;
```

Members

`usri1053_home_dir_drive`

Specifies the drive letter to assign to the user's home directory for logon purposes. The user is specified in the *username* parameter to the **NetUserSetInfo** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_11 structure (lmaccess.h)

2/1/2021 • 7 minutes to read • Edit Online

The **USER_INFO_11** structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, and other user-related network statistics.

Syntax

```
typedef struct _USER_INFO_11 {
  LPWSTR usri11_name;
  LPWSTR usri11_comment;
  LPWSTR usri11_usr_comment;
  LPWSTR usri11_full_name;
  DWORD  usri11_priv;
  DWORD  usri11_auth_flags;
  DWORD  usri11_password_age;
  LPWSTR usri11_home_dir;
  LPWSTR usri11_parms;
  DWORD  usri11_last_logon;
  DWORD  usri11_last_logoff;
  DWORD  usri11_bad_pw_count;
  DWORD  usri11_num_logons;
  LPWSTR usri11_logon_server;
  DWORD  usri11_country_code;
  LPWSTR usri11_workstations;
  DWORD  usri11_max_storage;
  DWORD  usri11_units_per_week;
  PBYTE  usri11_logon_hours;
  DWORD  usri11_code_page;
} USER_INFO_11, *PUSER_INFO_11, *LPUSER_INFO_11;
```

Members

`usri11_name`

Type: **LPWSTR**

A pointer to a Unicode character that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member. For more information, see the following Remarks section.

`usri11_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the user account. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri11_usr_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a user comment. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri11_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri11_priv`

Type: **DWORD**

The level of privilege assigned to the **usri11_name** member. For calls to the NetUserAdd function, this member must be USER_PRIV_USER. For calls to NetUserSetInfo, this member must be the value returned from the NetUserGetInfo function or the NetUserEnum function. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri11_auth_flags`

Type: **DWORD**

A set of bit flags defining the user's operator privileges.

Calls to the NetUserGetInfo function and the NetUserEnum function return a value based on the user's local group membership. If the user is a member of Print Operators, AF_OP_PRINT is set. If the user is a member of Server Operators, AF_OP_SERVER is set. If the user is a member of the Account Operators, AF_OP_ACCOUNTS is set. AF_OP_COMM is never set.

The NetUserAdd and NetUserSetInfo functions ignore this member.

The following restrictions apply:

- When you call the NetUserAdd function, this member must be zero.
- When you call the NetUserSetInfo function, this member must be the value returned from a call to NetUserGetInfo or to NetUserEnum.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.

AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

`usri11_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri11_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri11_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory for the user specified in the **usri11_name** member. The string can be **NULL**.

`usri11_parms`

Type: **LPWSTR**

A pointer to a Unicode string that is reserved for use by applications. This string can be a **NULL** string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

`usri11_last_logon`

Type: **DWORD**

The date and time when the last logon occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. The NetUserAdd and NetUserSetInfo functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logon occurred at the time indicated by the largest retrieved value.

`usri11_last_logoff`

Type: **DWORD**

This member is currently not used.

The date and time when the last logoff occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of zero indicates that the last logoff time is unknown. The **NetUserAdd** function and the **NetUserSetInfo** function ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logoff occurred at the time indicated by the largest retrieved value.

`usri11_bad_pw_count`

Type: **DWORD**

The number of times the user tried to log on to this account using an incorrect password. A value of – 1 indicates that the value is unknown. The NetUserAdd and NetUserSetInfo functions ignore this member.

This member is replicated from the primary domain controller (PDC); it is also maintained on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user tried to log on using an incorrect password is the largest value retrieved.

`usri11_num_logons`

Type: **DWORD**

The number of times the user has logged on successfully to this account. A value of – 1 indicates that the value is unknown. Calls to the **NetUserAdd** and **NetUserSetInfo** functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user logged on successfully is the sum of the retrieved values.

`usri11_logon_server`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A **NULL** string indicates that requests should be sent to the domain controller.

For Windows servers, NetUserGetInfo and NetUserEnum return *. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri11_country_code`

Type: **DWORD**

The country/region code for the user's language of choice.

`usri11_workstations`

Type: **LPWSTR**

A pointer to a Unicode string that contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas. A **NULL** string indicates that there is no restriction. To disable logons from all workstations to this account, set the UF_ACCOUNTDISABLE value in the **usri11_flags** member.

`usri11_max_storage`

Type: **DWORD**

The maximum amount of disk space the user can use. Specify USER_MAXSTORAGE_UNLIMITED to use all available disk space.

`usri11_units_per_week`

Type: **DWORD**

The number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri11_logon_hours** member.

This member must be UNITS_PER_WEEK for LAN Manager 2.0. This element is ignored by the NetUserAdd and NetUserSetInfo functions.

For service applications, the units must be one of the following values: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri11_logon_hours`

Type: **PBYTE**

A pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time (GMT).

The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Specify a **NULL** pointer in this member when calling the NetUserAdd function to indicate no time restriction. Specify a **NULL** pointer when calling the NetUserSetInfo function to indicate that no change is to be made to the times during which the user can log on.

`usri11_code_page`

Type: **DWORD**

The code page for the user's language of choice.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserDel

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_2 structure (lmaccess.h)

2/1/2021 • 10 minutes to read • Edit Online

The **USER_INFO_2** structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, and other user-related network statistics.

Syntax

```
typedef struct _USER_INFO_2 {
  LPWSTR usri2_name;
  LPWSTR usri2_password;
  DWORD  usri2_password_age;
  DWORD  usri2_priv;
  LPWSTR usri2_home_dir;
  LPWSTR usri2_comment;
  DWORD  usri2_flags;
  LPWSTR usri2_script_path;
  DWORD  usri2_auth_flags;
  LPWSTR usri2_full_name;
  LPWSTR usri2_usr_comment;
  LPWSTR usri2_parms;
  LPWSTR usri2_workstations;
  DWORD  usri2_last_logon;
  DWORD  usri2_last_logoff;
  DWORD  usri2_acct_expires;
  DWORD  usri2_max_storage;
  DWORD  usri2_units_per_week;
  PBYTE  usri2_logon_hours;
  DWORD  usri2_bad_pw_count;
  DWORD  usri2_num_logons;
  LPWSTR usri2_logon_server;
  DWORD  usri2_country_code;
  DWORD  usri2_code_page;
} USER_INFO_2, *PUSER_INFO_2, *LPUSER_INFO_2;
```

Members

`usri2_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member. For more information, see the following Remarks section.

`usri2_password`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the password for the user identified by the **usri2_name** member. The length cannot exceed PWLEN bytes. The NetUserEnum and NetUserGetInfo functions return a **NULL** pointer to maintain password security.

By convention, the length of passwords is limited to LM20_PWLEN characters.

`usri2_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri2_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri2_priv`

Type: **DWORD**

The level of privilege assigned to the **usri2_name** member. For calls to the NetUserAdd function, this member must be USER_PRIV_USER. For the NetUserSetInfo function, this member must be the value returned by the **NetUserGetInfo** function or the **NetUserEnum** function. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri2_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory for the user specified by the **usri2_name** member. The string can be **NULL**.

`usri2_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment to associate with the user account. The string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri2_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.

UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri2_script_path`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be **NULL**.

`usri2_auth_flags`

Type: **DWORD**

The user's operator privileges.

Calls to the **NetUserGetInfo** and **NetUserEnum** functions return a value based on the user's local group membership. If the user is a member of Print Operators, AF_OP_PRINT is set. If the user is a member of Server Operators, AF_OP_SERVER is set. If the user is a member of the Account Operators, AF_OP_ACCOUNTS is set. AF_OP_COMM is never set. For more information about user and group account rights, see Privileges.

The following restrictions apply:

- When you call the NetUserAdd function, this member must be zero.
- When you call the NetUserSetInfo function, this member must be the value returned from a call to NetUserGetInfo or to NetUserEnum.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.

AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

`usri2_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri2_usr_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a user comment. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri2_parms`

Type: **LPWSTR**

A pointer to a Unicode string that is reserved for use by applications. This string can be a **NULL** string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

`usri2_workstations`

Type: **LPWSTR**

> **IMPORTANT**
> You should no longer use **usri2_workstations**. Instead, you can control sign-in access to workstations by configuring the User Rights Assignment settings (**Allow log on locally** and **Deny log on locally**, or **Allow log on through Remote Desktop Services** and **Deny log on through Remote Desktop Services**).

A pointer to a Unicode string that contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas. A **NULL** string indicates that there is no restriction. To disable logons from all workstations to this account, set the UF_ACCOUNTDISABLE value in the **usri2_flags** member.

`usri2_last_logon`

Type: **DWORD**

The date and time when the last logon occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. This member is ignored by the NetUserAdd and NetUserSetInfo functions.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logon occurred at the time indicated by the largest retrieved value.

`usri2_last_logoff`

Type: **DWORD**

This member is currently not used.

Indicates when the last logoff occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of zero indicates that the last logoff time is unknown.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logoff occurred at the time indicated by the largest retrieved value.

`usri2_acct_expires`

Type: **DWORD**

The date and time when the account expires. This value is stored as the number of seconds elapsed since 00:00:00, January 1, 1970, GMT. A value of TIMEQ_FOREVER indicates that the account never expires.

`usri2_max_storage`

Type: **DWORD**

The maximum amount of disk space the user can use. Specify USER_MAXSTORAGE_UNLIMITED to use all available disk space.

`usri2_units_per_week`

Type: **DWORD**

The number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri2_logon_hours** member.

This value must be UNITS_PER_WEEK for LAN Manager 2.0. This element is ignored by the NetUserAdd and NetUserSetInfo functions.

For service applications, the units must be one of the following values: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri2_logon_hours`

Type: **PBYTE**

A pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time (GMT).

The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Specify a **NULL** pointer in this member when calling the NetUserAdd function to indicate no time restriction. Specify a **NULL** pointer when calling the NetUserSetInfo function to indicate that no change is to be made to the times during which the user can log on.

`usri2_bad_pw_count`

Type: **DWORD**

The number of times the user tried to log on to the account using an incorrect password. A value of – 1 indicates that the value is unknown. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

This member is replicated from the primary domain controller (PDC); it is also maintained on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user tried to log on using an incorrect password is the largest value retrieved.

`usri2_num_logons`

Type: **DWORD**

The number of times the user logged on successfully to this account. A value of – 1 indicates that the value is unknown. Calls to the **NetUserAdd** and **NetUserSetInfo** functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user logged on successfully is the sum of the retrieved values.

`usri2_logon_server`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A **NULL** string indicates that requests should be sent to the domain controller.

For Windows servers, NetUserGetInfo and NetUserEnum return *. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri2_country_code`

Type: **DWORD**

The country/region code for the user's language of choice.

`usri2_code_page`

Type: **DWORD**

The code page for the user's language of choice.

Remarks

For more information about user and group account rights, see Privileges.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_20 structure (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **USER_INFO_20** structure contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's relative ID (RID).

> **Note**
> The USER_INFO_23 structure supersedes the **USER_INFO_20** structure. It is recommended that applications use the **USER_INFO_23** structure instead of the **USER_INFO_20** structure.

Syntax

```
typedef struct _USER_INFO_20 {
  LPWSTR usri20_name;
  LPWSTR usri20_full_name;
  LPWSTR usri20_comment;
  DWORD  usri20_flags;
  DWORD  usri20_user_id;
} USER_INFO_20, *PUSER_INFO_20, *LPUSER_INFO_20;
```

Members

`usri20_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member. For more information, see the following Remarks section.

`usri20_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri20_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the user account. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri20_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.

UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri20_user_id`

Type: **DWORD**

The user's relative identifier (RID). The RID is determined by the Security Account Manager (SAM) when the user is created. It uniquely defines this user account to SAM within the domain. The NetUserAdd and NetUserSetInfo functions ignore this member. For more information about RIDs, see SID Components.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_21 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_21** structure contains a one-way encrypted LAN Manager 2.*x*-compatible password.

Syntax

```
typedef struct _USER_INFO_21 {
  BYTE usri21_password[ENCRYPTED_PWLEN];
} USER_INFO_21, *PUSER_INFO_21, *LPUSER_INFO_21;
```

Members

`usri21_password`

Specifies a one-way encrypted LAN Manager 2.*x*-compatible password.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_22 structure (lmaccess.h)

2/1/2021 • 10 minutes to read • Edit Online

The **USER_INFO_22** structure contains information about a user account, including the account name, privilege level, the path to the user's home directory, a one-way encrypted LAN Manager 2.*x*-compatible password, and other user-related network statistics.

Syntax

```
typedef struct _USER_INFO_22 {
  LPWSTR usri22_name;
  BYTE   usri22_password[ENCRYPTED_PWLEN];
  DWORD  usri22_password_age;
  DWORD  usri22_priv;
  LPWSTR usri22_home_dir;
  LPWSTR usri22_comment;
  DWORD  usri22_flags;
  LPWSTR usri22_script_path;
  DWORD  usri22_auth_flags;
  LPWSTR usri22_full_name;
  LPWSTR usri22_usr_comment;
  LPWSTR usri22_parms;
  LPWSTR usri22_workstations;
  DWORD  usri22_last_logon;
  DWORD  usri22_last_logoff;
  DWORD  usri22_acct_expires;
  DWORD  usri22_max_storage;
  DWORD  usri22_units_per_week;
  PBYTE  usri22_logon_hours;
  DWORD  usri22_bad_pw_count;
  DWORD  usri22_num_logons;
  LPWSTR usri22_logon_server;
  DWORD  usri22_country_code;
  DWORD  usri22_code_page;
} USER_INFO_22, *PUSER_INFO_22, *LPUSER_INFO_22;
```

Members

`usri22_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member. For more information, see the following Remarks section.

`usri22_password`

Type: **BYTE[ENCRYPTED_PWLEN]**

A one-way encrypted LAN Manager 2.*x*-compatible password.

`usri22_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri22_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri22_priv`

Type: **DWORD**

The level of privilege assigned to the **usri22_name** member. Calls to the **NetUserAdd** function must specify USER_PRIV_USER. When you call the **NetUserSetInfo** function this member must be the value returned from the NetUserGetInfo or the NetUserEnum function. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri22_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory for the user specified by the **usri22_name** member. The string can be null.

`usri22_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the user account. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri22_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.

UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.

UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri22_script_path`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be null.

`usri22_auth_flags`

Type: **DWORD**

The user's operator privileges.

Calls to the **NetUserGetInfo** function and the **NetUserEnum** function return a value based on the user's local group membership. If the user is a member of Print Operators, AF_OP_PRINT, is set. If the user is a member of Server Operators, AF_OP_SERVER, is set. If the user is a member of the Account Operators, AF_OP_ACCOUNTS, is set. AF_OP_COMM is never set.

The following restrictions apply:

- When you call the NetUserAdd function, this member must be zero.
- When you call the NetUserSetInfo function, this member must be the value returned from a call to NetUserGetInfo or to NetUserEnum.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.
AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

`usri22_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri22_usr_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a user comment. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri22_parms`

Type: **LPWSTR**

A pointer to a Unicode string that is reserved for use by applications. This string can be a null string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

`usri22_workstations`

Type: **LPWSTR**

> **IMPORTANT**
> You should no longer use **usri22_workstations**. Instead, you can control sign-in access to workstations by configuring the User Rights Assignment settings (**Allow log on locally** and **Deny log on locally**, or **Allow log on through Remote Desktop Services** and **Deny log on through Remote Desktop Services**).

A pointer to a Unicode string that contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas. A null string indicates that there is no restriction. To disable logons from all workstations to this account, set the UF_ACCOUNTDISABLE value in the **usri22_flags** member.

`usri22_last_logon`

Type: **DWORD**

The date and time when the last logon occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. Calls to the NetUserAdd and the NetUserSetInfo functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logon occurred at the time indicated by the largest retrieved value.

`usri22_last_logoff`

Type: **DWORD**

This member is currently not used.

The date and time when the last logoff occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of zero means that the last logoff time is unknown. This element is ignored by calls to **NetUserAdd** and **NetUserSetInfo**.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logoff occurred at the time indicated by the largest retrieved value.

`usri22_acct_expires`

Type: **DWORD**

The date and time when the account expires. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of TIMEQ_FOREVER indicates that the account never expires.

`usri22_max_storage`

Type: **DWORD**

The maximum amount of disk space the user can use. Specify USER_MAXSTORAGE_UNLIMITED to use all available disk space.

`usri22_units_per_week`

Type: **DWORD**

The number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri22_logon_hours** member.

This value must be UNITS_PER_WEEK for LAN Manager 2.0. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

For service applications, the units must be one of the following: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri22_logon_hours`

Type: **PBYTE**

A pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time (GMT).

The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Specify a null pointer in this member when calling the NetUserAdd function to indicate no time restriction. Specify a null pointer when calling the NetUserSetInfo function to indicate that no change is to be made to the times during which the user can log on.

`usri22_bad_pw_count`

Type: **DWORD**

The number of times the user tried to log on to this account using an incorrect password. A value of – 1 indicates that the value is unknown. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

This member is replicated from the primary domain controller (PDC); it is also maintained on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user tried to log on using an incorrect password is the largest value retrieved.

`usri22_num_logons`

Type: **DWORD**

The number of times the user logged on successfully to this account. A value of – 1 indicates that the value is unknown. Calls to the **NetUserAdd** and **NetUserSetInfo** functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an

accurate value, you must query each BDC in the domain. The number of times the user logged on successfully is the sum of the retrieved values.

`usri22_logon_server`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A null string indicates that requests should be sent to the domain controller.

For Windows servers, the NetUserGetInfo and NetUserEnum functions return *. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

`usri22_country_code`

Type: **DWORD**

The country/region code for the user's language of choice.

This value is ignored.

`usri22_code_page`

Type: **DWORD**

The code page for the user's language of choice.

This value is ignored.

Remarks

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

USER_INFO_23 structure (lmaccess.h)

2/1/2021 • 3 minutes to read • Edit Online

The **USER_INFO_23** structure contains information about a user account, including the account name, the user's full name, a comment associated with the account, and the user's security identifier (SID).

> **Note**
> The **USER_INFO_23** structure supersedes the USER_INFO_20 structure. It is recommended that applications use the **USER_INFO_23** structure instead of the **USER_INFO_20** structure.

Syntax

```
typedef struct _USER_INFO_23 {
  LPWSTR usri23_name;
  LPWSTR usri23_full_name;
  LPWSTR usri23_comment;
  DWORD  usri23_flags;
  PSID   usri23_user_sid;
} USER_INFO_23, *PUSER_INFO_23, *LPUSER_INFO_23;
```

Members

`usri23_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. Calls to the NetUserSetInfo function ignore this member.

`usri23_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri23_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment associated with the user account. This string can be a null string, or it can have any number of characters before the terminating null character.

`usri23_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.
UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.

UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri23_user_sid`

Type: **PSID**

A pointer to a SID structure that contains the security identifier (SID) that uniquely identifies the user. The NetUserAdd and NetUserSetInfo functions ignore this member.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

SID

User Functions

USER_INFO_24 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_INFO_24** structure contains user account information on an account which is connected to an Internet identity. This information includes the Internet provider name for the user, the user's Internet name, and the user's security identifier (SID).

Syntax

```
typedef struct _USER_INFO_24 {
  BOOL    usri24_internet_identity;
  DWORD   usri24_flags;
  LPWSTR usri24_internet_provider_name;
  LPWSTR usri24_internet_principal_name;
  PSID    usri24_user_sid;
} USER_INFO_24, *PUSER_INFO_24, *LPUSER_INFO_24;
```

Members

`usri24_internet_identity`

A boolean value that indicates whether an account is connected to an Internet identity.

This member is true if the account is connected to an Internet identity. The other members in this structure can be used.

If this member is false, then the account is not connected to an Internet identity and other members in this structure should be ignored.

`usri24_flags`

A set of flags. This member must be zero.

`usri24_internet_provider_name`

A pointer to a Unicode string that specifies the Internet provider name.

`usri24_internet_principal_name`

A pointer to a Unicode string that specifies the user's Internet name.

`usri24_user_sid`

The local account SID of the user.

Remarks

A user's account for logging onto Windows can be connected to an Internet identity. The user account can be a local account on a computer or a domain account for computers joined to a domain. The **USER_INFO_24** structure is used to provide information on an account which is connected to an Internet identity.

On Windows 8 and Windows Server 2012, the Internet identity for a connected account can often be used instead of the computer account.

Requirements

Minimum supported client	Windows 8 [desktop apps only]
Minimum supported server	Windows Server 2012 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserGetInfo

SID

USER_INFO_3 structure (lmaccess.h)

2/1/2021 • 11 minutes to read • Edit Online

The **USER_INFO_3** structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, relative identifiers (RIDs), and other user-related network statistics.

Syntax

```
typedef struct _USER_INFO_3 {
  LPWSTR usri3_name;
  LPWSTR usri3_password;
  DWORD  usri3_password_age;
  DWORD  usri3_priv;
  LPWSTR usri3_home_dir;
  LPWSTR usri3_comment;
  DWORD  usri3_flags;
  LPWSTR usri3_script_path;
  DWORD  usri3_auth_flags;
  LPWSTR usri3_full_name;
  LPWSTR usri3_usr_comment;
  LPWSTR usri3_parms;
  LPWSTR usri3_workstations;
  DWORD  usri3_last_logon;
  DWORD  usri3_last_logoff;
  DWORD  usri3_acct_expires;
  DWORD  usri3_max_storage;
  DWORD  usri3_units_per_week;
  PBYTE  usri3_logon_hours;
  DWORD  usri3_bad_pw_count;
  DWORD  usri3_num_logons;
  LPWSTR usri3_logon_server;
  DWORD  usri3_country_code;
  DWORD  usri3_code_page;
  DWORD  usri3_user_id;
  DWORD  usri3_primary_group_id;
  LPWSTR usri3_profile;
  LPWSTR usri3_home_dir_drive;
  DWORD  usri3_password_expired;
} USER_INFO_3, *PUSER_INFO_3, *LPUSER_INFO_3;
```

Members

`usri3_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. For the NetUserSetInfo function, this member is ignored. For more information, see the following Remarks section.

`usri3_password`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the password for the user identified by the **usri3_name** member. The length cannot exceed PWLEN bytes. The NetUserEnum and NetUserGetInfo functions return a **NULL** pointer to maintain password security.

By convention, the length of passwords is limited to LM20_PWLEN characters.

`usri3_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri3_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri3_priv`

Type: **DWORD**

The level of privilege assigned to the **usri3_name** member. The NetUserAdd and NetUserSetInfo functions ignore this member. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri3_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory of the user specified by the **usri3_name** member. The string can be **NULL**.

`usri3_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment to associate with the user account. The string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri3_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.

UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is not supported.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is not supported.

The following values describe the account type. Only one value can be set. You cannot change the account type

using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri3_script_path`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be **NULL**.

`usri3_auth_flags`

Type: **DWORD**

The user's operator privileges.

For the NetUserGetInfo and NetUserEnum functions, the appropriate value is returned based on the local group membership. If the user is a member of Print Operators, AF_OP_PRINT is set. If the user is a member of Server Operators, AF_OP_SERVER is set. If the user is a member of the Account Operators, AF_OP_ACCOUNTS is set. AF_OP_COMM is never set.

The NetUserAdd and NetUserSetInfo functions ignore this member.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.

AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

`usri3_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri3_usr_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a user comment. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri3_parms`

Type: **LPWSTR**

A pointer to a Unicode string that is reserved for use by applications. This string can be a **NULL** string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

`usri3_workstations`

Type: **LPWSTR**

> **IMPORTANT**
> You should no longer use **usri3_workstations**. Instead, you can control sign-in access to workstations by configuring the User Rights Assignment settings (**Allow log on locally** and **Deny log on locally**, or **Allow log on through Remote Desktop Services** and **Deny log on through Remote Desktop Services**).

A pointer to a Unicode string that contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas. If you do not want to restrict the number of workstations, use a **NULL** string. To disable logons from all workstations to this account, set the UF_ACCOUNTDISABLE value in the **usri3_flags** member.

`usri3_last_logon`

Type: **DWORD**

The date and time when the last logon occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. This member is ignored by the NetUserAdd and NetUserSetInfo functions.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logon occurred at the time indicated by the largest retrieved value.

`usri3_last_logoff`

Type: **DWORD**

This member is currently not used.

The date and time when the last logoff occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of zero indicates that the last logoff time is unknown.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logoff occurred at the time indicated by the largest retrieved value.

`usri3_acct_expires`

Type: **DWORD**

The date and time when the account expires. This value is stored as the number of seconds elapsed since 00:00:00, January 1, 1970, GMT. A value of TIMEQ_FOREVER indicates that the account never expires.

`usri3_max_storage`

Type: **DWORD**

The maximum amount of disk space the user can use. Specify USER_MAXSTORAGE_UNLIMITED to use all available disk space.

`usri3_units_per_week`

Type: **DWORD**

The number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri3_logon_hours** member.

This value must be UNITS_PER_WEEK for LAN Manager 2.0. This element is ignored by the NetUserAdd and NetUserSetInfo functions.

For service applications, the units must be one of the following values: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri3_logon_hours`

Type: **PBYTE**

A pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time (GMT).

The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Specify a **NULL** pointer in this member when calling the NetUserAdd function to indicate no time restriction. Specify a **NULL** pointer when calling the NetUserSetInfo function to indicate that no change is to be made to the times during which the user can log on.

`usri3_bad_pw_count`

Type: **DWORD**

The number of times the user tried to log on to the account using an incorrect password. A value of – 1 indicates that the value is unknown. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

This member is replicated from the primary domain controller (PDC); it is also maintained on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user tried to log on using an incorrect password is the largest value retrieved.

`usri3_num_logons`

Type: **DWORD**

The number of times the user logged on successfully to this account. A value of – 1 indicates that the value is unknown. Calls to the **NetUserAdd** and **NetUserSetInfo** functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user logged on successfully is the sum of the retrieved values.

`usri3_logon_server`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A **NULL** string indicates that requests should be sent to the domain controller.

For Windows servers, NetUserGetInfo and NetUserEnum return *. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri3_country_code`

Type: **DWORD**

The country/region code for the user's language of choice.

`usri3_code_page`

Type: **DWORD**

The code page for the user's language of choice.

`usri3_user_id`

Type: **DWORD**

The relative ID (RID) of the user. The RID is determined by the Security Account Manager (SAM) when the user is created. It uniquely defines the user account to SAM within the domain. The NetUserAdd and NetUserSetInfo functions ignore this member. For more information about RIDs, see SID Components.

`usri3_primary_group_id`

Type: **DWORD**

The RID of the Primary Global Group for the user. When you call the **NetUserAdd** function, this member must be DOMAIN_GROUP_RID_USERS (defined in WinNT.h). When you call **NetUserSetInfo**, this member must be the RID of a global group in which the user is enrolled. For more information, see Well-Known SIDs.

`usri3_profile`

Type: **LPWSTR**

A pointer to a Unicode string that specifies a path to the user's profile. This value can be a **NULL** string, a local absolute path, or a UNC path.

`usri3_home_dir_drive`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the drive letter assigned to the user's home directory for logon purposes.

`usri3_password_expired`

Type: **DWORD**

The password expiration information.

The NetUserGetInfo and NetUserEnum functions return zero if the password has not expired (and nonzero if it has).

When you call NetUserAdd or NetUserSetInfo, specify a nonzero value in this member to inform users that they must change their password at the next logon. To turn off this message, call **NetUserSetInfo** and specify zero in this member. Note that you cannot specify zero to negate the expiration of a password that has already expired.

Remarks

The **USER_INFO_3** structure can be used with the NetUserAdd, NetUserEnum, NetUserSetInfo, and NetUserGetInfofunctions.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Note that the USER_INFO_4 structure supersedes the **USER_INFO_3** structure on Windows XP and later. It is recommended that applications use the **USER_INFO_4** structure instead of the **USER_INFO_3** structure with the NetUserAdd, NetUserSetInfo, and NetUserGetInfofunctions on Windows XP and later.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

User Functions

USER_INFO_4 structure (lmaccess.h)

2/1/2021 • 11 minutes to read • Edit Online

The **USER_INFO_4** structure contains information about a user account, including the account name, password data, privilege level, the path to the user's home directory, security identifier (SID), and other user-related network statistics.

Syntax

```
typedef struct _USER_INFO_4 {
  LPWSTR usri4_name;
  LPWSTR usri4_password;
  DWORD  usri4_password_age;
  DWORD  usri4_priv;
  LPWSTR usri4_home_dir;
  LPWSTR usri4_comment;
  DWORD  usri4_flags;
  LPWSTR usri4_script_path;
  DWORD  usri4_auth_flags;
  LPWSTR usri4_full_name;
  LPWSTR usri4_usr_comment;
  LPWSTR usri4_parms;
  LPWSTR usri4_workstations;
  DWORD  usri4_last_logon;
  DWORD  usri4_last_logoff;
  DWORD  usri4_acct_expires;
  DWORD  usri4_max_storage;
  DWORD  usri4_units_per_week;
  PBYTE  usri4_logon_hours;
  DWORD  usri4_bad_pw_count;
  DWORD  usri4_num_logons;
  LPWSTR usri4_logon_server;
  DWORD  usri4_country_code;
  DWORD  usri4_code_page;
  PSID   usri4_user_sid;
  DWORD  usri4_primary_group_id;
  LPWSTR usri4_profile;
  LPWSTR usri4_home_dir_drive;
  DWORD  usri4_password_expired;
} USER_INFO_4, *PUSER_INFO_4, *LPUSER_INFO_4;
```

Members

`usri4_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the user account. For the NetUserSetInfo function, this member is ignored.

`usri4_password`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the password for the user identified by the **usri4_name** member. The length cannot exceed PWLEN bytes. The NetUserGetInfo function returns a **NULL** pointer to maintain password security.

By convention, the length of passwords is limited to LM20_PWLEN characters.

`usri4_password_age`

Type: **DWORD**

The number of seconds that have elapsed since the **usri4_password** member was last changed. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri4_priv`

Type: **DWORD**

The level of privilege assigned to the **usri4_name** member. The NetUserAdd and NetUserSetInfo functions ignore this member. This member can be one of the following values. For more information about user and group account rights, see Privileges.

VALUE	MEANING
USER_PRIV_GUEST	Guest
USER_PRIV_USER	User
USER_PRIV_ADMIN	Administrator

`usri4_home_dir`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path of the home directory of the user specified by the **usri4_name** member. The string can be **NULL**.

`usri4_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a comment to associate with the user account. The string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri4_flags`

Type: **DWORD**

This member can be one or more of the following values.

Note that setting user account control flags may require certain privileges and control access rights. For more information, see the Remarks section of the NetUserSetInfo function.

VALUE	MEANING
UF_SCRIPT	The logon script executed. This value must be set.
UF_ACCOUNTDISABLE	The user's account is disabled.

UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored.
UF_PASSWD_NOTREQD	No password is required.
UF_PASSWD_CANT_CHANGE	The user cannot change the password.
UF_LOCKOUT	The account is currently locked out. You can call the NetUserSetInfo function to clear this value and unlock a previously locked account. You cannot use this value to lock a previously unlocked account.
UF_DONT_EXPIRE_PASSWD	The password should never expire on the account.
UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED	The user's password is stored under reversible encryption in the Active Directory.
UF_NOT_DELEGATED	Marks the account as "sensitive"; other users cannot act as delegates of this user account.
UF_SMARTCARD_REQUIRED	Requires the user to log on to the user account with a smart card.
UF_USE_DES_KEY_ONLY	Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
UF_DONT_REQUIRE_PREAUTH	This account does not require Kerberos preauthentication for logon.
UF_TRUSTED_FOR_DELEGATION	The account is enabled for delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assume a client's identity and authenticate as that user to other remote servers on the network.
UF_PASSWORD_EXPIRED	The user's password has expired. **Windows 2000:** This value is ignored.
UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION	The account is trusted to authenticate a user outside of the Kerberos security package and delegate that user through constrained delegation. This is a security-sensitive setting; accounts with this option enabled should be tightly controlled. This setting allows a service running under the account to assert a client's identity and authenticate as that user to specifically configured services on the network. **Windows XP/2000:** This value is ignored.

The following values describe the account type. Only one value can be set. You cannot change the account type

using the NetUserSetInfo function.

VALUE	MEANING
UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This account provides user access to this domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a computer that is a member of this domain.
UF_SERVER_TRUST_ACCOUNT	This is a computer account for a backup domain controller that is a member of this domain.
UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a domain that trusts other domains.

`usri4_script_path`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file. The string can also be **NULL**.

`usri4_auth_flags`

Type: **DWORD**

The user's operator privileges.

For the NetUserGetInfo function, the appropriate value is returned based on the local group membership. If the user is a member of Print Operators, AF_OP_PRINT is set. If the user is a member of Server Operators, AF_OP_SERVER is set. If the user is a member of the Account Operators, AF_OP_ACCOUNTS is set. AF_OP_COMM is never set.

The NetUserAdd and NetUserSetInfo functions ignore this member.

This member can be one or more of the following values.

VALUE	MEANING
AF_OP_PRINT	The print operator privilege is enabled.
AF_OP_COMM	The communications operator privilege is enabled.
AF_OP_SERVER	The server operator privilege is enabled.

AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

`usri4_full_name`

Type: **LPWSTR**

A pointer to a Unicode string that contains the full name of the user. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri4_usr_comment`

Type: **LPWSTR**

A pointer to a Unicode string that contains a user comment. This string can be a **NULL** string, or it can have any number of characters before the terminating null character.

`usri4_parms`

Type: **LPWSTR**

A pointer to a Unicode string that is reserved for use by applications. This string can be a **NULL** string, or it can have any number of characters before the terminating null character. Microsoft products use this member to store user configuration information. Do not modify this information.

`usri4_workstations`

Type: **LPWSTR**

> **IMPORTANT**
> You should no longer use **usri4_workstations**. Instead, you can control sign-in access to workstations by configuring the User Rights Assignment settings (**Allow log on locally** and **Deny log on locally**, or **Allow log on through Remote Desktop Services** and **Deny log on through Remote Desktop Services**).

A pointer to a Unicode string that contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas. If you do not want to restrict the number of workstations, use a **NULL** string. To disable logons from all workstations to this account, set the UF_ACCOUNTDISABLE value in the **usri4_flags** member.

`usri4_last_logon`

Type: **DWORD**

The date and time when the last logon occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. This member is ignored by the NetUserAdd and NetUserSetInfo functions.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logon occurred at the time indicated by the largest retrieved value.

`usri4_last_logoff`

Type: **DWORD**

This member is currently not used.

The date and time when the last logoff occurred. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT. A value of zero indicates that the last logoff time is unknown.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The last logoff occurred at the time indicated by the largest retrieved value.

`usri4_acct_expires`

Type: **DWORD**

The date and time when the account expires. This value is stored as the number of seconds elapsed since 00:00:00, January 1, 1970, GMT. A value of TIMEQ_FOREVER indicates that the account never expires.

`usri4_max_storage`

Type: **DWORD**

The maximum amount of disk space the user can use. Specify USER_MAXSTORAGE_UNLIMITED to use all available disk space.

`usri4_units_per_week`

Type: **DWORD**

The number of equal-length time units into which the week is divided. This value is required to compute the length of the bit string in the **usri4_logon_hours** member.

This value must be UNITS_PER_WEEK for LAN Manager 2.0. This element is ignored by the NetUserAdd and NetUserSetInfo functions.

For service applications, the units must be one of the following values: SAM_DAYS_PER_WEEK, SAM_HOURS_PER_WEEK, or SAM_MINUTES_PER_WEEK.

`usri4_logon_hours`

Type: **PBYTE**

A pointer to a 21-byte (168 bits) bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time (GMT).

The first bit (bit 0, word 0) is Sunday, 0:00 to 0:59; the second bit (bit 1, word 0) is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Specify a **NULL** pointer in this member when calling the NetUserAdd function to indicate no time restriction. Specify a **NULL** pointer when calling the NetUserSetInfo function to indicate that no change is to be made to the times during which the user can log on.

`usri4_bad_pw_count`

Type: **DWORD**

The number of times the user tried to log on to the account using an incorrect password. A value of – 1 indicates that the value is unknown. Calls to the NetUserAdd and NetUserSetInfo functions ignore this member.

This member is replicated from the primary domain controller (PDC); it is also maintained on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user tried to log on using an incorrect password is the largest value retrieved.

`usri4_num_logons`

Type: **DWORD**

The number of times the user logged on successfully to this account. A value of – 1 indicates that the value is unknown. Calls to the **NetUserAdd** and **NetUserSetInfo** functions ignore this member.

This member is maintained separately on each backup domain controller (BDC) in the domain. To obtain an accurate value, you must query each BDC in the domain. The number of times the user logged on successfully is the sum of the retrieved values.

`usri4_logon_server`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\). To indicate that the logon request can be handled by any logon server, specify an asterisk (*) for the server name. A **NULL** string indicates that requests should be sent to the domain controller.

For Windows servers, the NetUserGetInfo function returns *.

The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri4_country_code`

Type: **DWORD**

The country/region code for the user's language of choice.

`usri4_code_page`

Type: **DWORD**

The code page for the user's language of choice.

`usri4_user_sid`

Type: **PSID**

A pointer to a SID structure that contains the security identifier (SID) that uniquely identifies the user. The NetUserAdd and NetUserSetInfo functions ignore this member.

`usri4_primary_group_id`

Type: **DWORD**

The relative identifier (RID) of the Primary Global Group for the user. When you call the **NetUserAdd** function, this member must be DOMAIN_GROUP_RID_USERS (defined in WinNT.h). When you call **NetUserSetInfo**, this member must be the RID of a global group in which the user is enrolled. For more information, see Well-Known SIDs and SID Components.

`usri4_profile`

Type: **LPWSTR**

A pointer to a Unicode string that specifies a path to the user's profile. This value can be a **NULL** string, a local absolute path, or a UNC path.

`usri4_home_dir_drive`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the drive letter assigned to the user's home directory for logon purposes.

`usri4_password_expired`

Type: **DWORD**

The password expiration information.

The NetUserGetInfo function return zero if the password has not expired (and nonzero if it has).

When you call NetUserAdd or NetUserSetInfo, specify a nonzero value in this member to inform users that they must change their password at the next logon. To turn off this message, call **NetUserSetInfo** and specify zero in this member. Note that you cannot specify zero to negate the expiration of a password that has already expired.

Remarks

The **USER_INFO_4** structure can be used with the NetUserAdd, NetUserSetInfo, and NetUserGetInfofunctions.

User account names are limited to 20 characters and group names are limited to 256 characters. In addition, account names cannot be terminated by a period and they cannot include commas or any of the following printable characters: ", /, , [,], :, |, <, >, +, =, ;, ?, *. Names also cannot include characters in the range 1-31, which are nonprintable.

Note that the **USER_INFO_4** structure supersedes the USER_INFO_3 structure on Windows XP and later. It is recommended that applications use the **USER_INFO_4** structure instead of the **USER_INFO_3** structure with the above functions on Windows XP and later.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserAdd

NetUserEnum

NetUserGetInfo

NetUserSetInfo

Network Management Overview

Network Management Structures

SID

USER_INFO_3

User Functions

USER_MODALS_INFO_0 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_0** structure contains global password information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_0 {
  DWORD usrmod0_min_passwd_len;
  DWORD usrmod0_max_passwd_age;
  DWORD usrmod0_min_passwd_age;
  DWORD usrmod0_force_logoff;
  DWORD usrmod0_password_hist_len;
} USER_MODALS_INFO_0, *PUSER_MODALS_INFO_0, *LPUSER_MODALS_INFO_0;
```

Members

`usrmod0_min_passwd_len`

Specifies the minimum allowable password length. Valid values for this element are zero through LM20_PWLEN.

`usrmod0_max_passwd_age`

Specifies, in seconds, the maximum allowable password age. A value of TIMEQ_FOREVER indicates that the password never expires. The minimum valid value for this element is ONE_DAY. The value specified must be greater than or equal to the value for the **usrmod0_min_passwd_age** member.

`usrmod0_min_passwd_age`

Specifies the minimum number of seconds that can elapse between the time a password changes and when it can be changed again. A value of zero indicates that no delay is required between password updates. The value specified must be less than or equal to the value for the **usrmod0_max_passwd_age** member.

`usrmod0_force_logoff`

Specifies, in seconds, the amount of time between the end of the valid logon time and the time when the user is forced to log off the network. A value of TIMEQ_FOREVER indicates that the user is never forced to log off. A value of zero indicates that the user will be forced to log off immediately when the valid logon time expires.

`usrmod0_password_hist_len`

Specifies the length of password history maintained. A new password cannot match any of the previous **usrmod0_password_hist_len** passwords. Valid values for this element are zero through DEF_MAX_PWHIST.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsGet

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1** structure contains logon server and domain controller information.

Syntax

```
typedef struct _USER_MODALS_INFO_1 {
  DWORD  usrmod1_role;
  LPWSTR usrmod1_primary;
} USER_MODALS_INFO_1, *PUSER_MODALS_INFO_1, *LPUSER_MODALS_INFO_1;
```

Members

`usrmod1_role`

Specifies the role of the logon server. The following values are defined.

VALUE	MEANING
UAS_ROLE_STANDALONE	The logon server is a stand-alone server.
UAS_ROLE_MEMBER	The logon server is a member.
UAS_ROLE_BACKUP	The logon server is a backup.
UAS_ROLE_PRIMARY	The logon server is a domain controller.

If the Netlogon service is not being used, the element should be set to UAS_ROLE_STANDALONE.

`usrmod1_primary`

Pointer to a Unicode string that specifies the name of the domain controller that stores the primary copy of the database for the user account manager.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmaccess.h (include Lm.h)

See also

NetUserModalsGet

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1001 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1001** structure contains the minimum length for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_1001 {
  DWORD usrmod1001_min_passwd_len;
} USER_MODALS_INFO_1001, *PUSER_MODALS_INFO_1001, *LPUSER_MODALS_INFO_1001;
```

Members

`usrmod1001_min_passwd_len`

Specifies the minimum allowable password length. Valid values for this element are zero through PWLEN.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1002 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1002** structure contains the maximum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_1002 {
  DWORD usrmod1002_max_passwd_age;
} USER_MODALS_INFO_1002, *PUSER_MODALS_INFO_1002, *LPUSER_MODALS_INFO_1002;
```

Members

`usrmod1002_max_passwd_age`

Specifies, in seconds, the maximum allowable password age. A value of TIMEQ_FOREVER indicates that the password never expires. The minimum valid value for this element is ONE_DAY. The value specified must be greater than or equal to the value for the usrmod*X*_min_passwd_age member.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1003 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1003** structure contains the minimum duration for passwords in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_1003 {
  DWORD usrmod1003_min_passwd_age;
} USER_MODALS_INFO_1003, *PUSER_MODALS_INFO_1003, *LPUSER_MODALS_INFO_1003;
```

Members

`usrmod1003_min_passwd_age`

Specifies the minimum number of seconds that can elapse between the time a password changes and when it can be changed again. A value of zero indicates that no delay is required between password updates. The value specified must be less than or equal to the value for the usrmod*X*_max_passwd_age member.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1004 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1004** structure contains forced logoff information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_1004 {
  DWORD usrmod1004_force_logoff;
} USER_MODALS_INFO_1004, *PUSER_MODALS_INFO_1004, *LPUSER_MODALS_INFO_1004;
```

Members

`usrmod1004_force_logoff`

Specifies, in seconds, the amount of time between the end of the valid logon time and the time when the user is forced to log off the network. A value of TIMEQ_FOREVER indicates that the user is never forced to log off. A value of zero indicates that the user will be forced to log off immediately when the valid logon time expires.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1005 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1005** structure contains password history information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_1005 {
  DWORD usrmod1005_password_hist_len;
} USER_MODALS_INFO_1005, *PUSER_MODALS_INFO_1005, *LPUSER_MODALS_INFO_1005;
```

Members

`usrmod1005_password_hist_len`

Specifies the length of password history that the system maintains. A new password cannot match any of the previous usrmod*X*_password_hist_len passwords. Valid values for this element are zero through DEF_MAX_PWHIST.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_1006 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1006** structure contains logon server information.

Syntax

```
typedef struct _USER_MODALS_INFO_1006 {
  DWORD usrmod1006_role;
} USER_MODALS_INFO_1006, *PUSER_MODALS_INFO_1006, *LPUSER_MODALS_INFO_1006;
```

Members

`usrmod1006_role`

Specifies the role of the logon server. This member can be one of the following values.

VALUE	MEANING
UAS_ROLE_STANDALONE	Logon server is a stand-alone. Use this value if no logon services are available.
UAS_ROLE_MEMBER	Logon server is a member.
UAS_ROLE_BACKUP	Logon server is a backup.
UAS_ROLE_PRIMARY	Logon server is a domain controller.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

USER_MODALS_INFO_1007 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_1007** structure contains domain controller information.

Syntax

```
typedef struct _USER_MODALS_INFO_1007 {
  LPWSTR usrmod1007_primary;
} USER_MODALS_INFO_1007, *PUSER_MODALS_INFO_1007, *LPUSER_MODALS_INFO_1007;
```

Members

`usrmod1007_primary`

Pointer to a Unicode string that specifies the name of the domain controller that stores the primary copy of the database for the user account manager.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_2 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_2** structure contains the Security Account Manager (SAM) domain name and identifier.

Syntax

```
typedef struct _USER_MODALS_INFO_2 {
  LPWSTR usrmod2_domain_name;
  PSID   usrmod2_domain_id;
} USER_MODALS_INFO_2, *PUSER_MODALS_INFO_2, *LPUSER_MODALS_INFO_2;
```

Members

`usrmod2_domain_name`

Specifies the name of the Security Account Manager (SAM) domain. For a domain controller, this is the name of the domain that the controller is a member of. For workstations, this is the name of the computer.

`usrmod2_domain_id`

Pointer to a SID structure that contains the security identifier (SID) of the domain named by the **usrmod2_domain_name** member.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsGet

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

USER_MODALS_INFO_3 structure (lmaccess.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_MODALS_INFO_3** structure contains lockout information for users and global groups in the security database, which is the security accounts manager (SAM) database or, in the case of domain controllers, the Active Directory.

Syntax

```
typedef struct _USER_MODALS_INFO_3 {
  DWORD usrmod3_lockout_duration;
  DWORD usrmod3_lockout_observation_window;
  DWORD usrmod3_lockout_threshold;
} USER_MODALS_INFO_3, *PUSER_MODALS_INFO_3, *LPUSER_MODALS_INFO_3;
```

Members

`usrmod3_lockout_duration`

Specifies, in seconds, how long a locked account remains locked before it is automatically unlocked.

`usrmod3_lockout_observation_window`

Specifies the maximum time, in seconds, that can elapse between any two failed logon attempts before lockout occurs.

`usrmod3_lockout_threshold`

Specifies the number of invalid password authentications that can occur before an account is marked "locked out."

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmaccess.h (include Lm.h)

See also

NetUserModalsGet

NetUserModalsSet

Network Management Overview

Network Management Structures

User Modal Functions

lmalert.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmalert.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
ALERT_OTHER_INFO	The ALERT_OTHER_INFO macro returns a pointer to the alert-specific data in an alert message. The data follows a STD_ALERT structure, and can be an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.
ALERT_VAR_DATA	The ALERT_VAR_DATA macro returns a pointer to the variable-length portion of an alert message. Variable-length data can follow an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.
NetAlertRaise	The NetAlertRaise function notifies all registered clients when a particular event occurs.
NetAlertRaiseEx	The NetAlertRaiseEx function notifies all registered clients when a particular event occurs. You can call this extended function to simplify the sending of an alert message because NetAlertRaiseEx does not require that you specify a STD_ALERT structure.

Structures

TITLE	DESCRIPTION
ADMIN_OTHER_INFO	The ADMIN_OTHER_INFO structure contains error message information. The NetAlertRaise and NetAlertRaiseEx functions use the ADMIN_OTHER_INFO structure to specify information when raising an administrator's interrupting message.
ERRLOG_OTHER_INFO	The ERRLOG_OTHER_INFO structure contains error log information. The NetAlertRaise and NetAlertRaiseEx functions use the ERRLOG_OTHER_INFO structure to specify information when adding a new entry to the error log.
PRINT_OTHER_INFO	Contains information about a print job.
STD_ALERT	The STD_ALERT structure contains the time and date when a significant event occurred.

TITLE	DESCRIPTION
USER_OTHER_INFO	The USER_OTHER_INFO structure contains user error code information. The NetAlertRaise and NetAlertRaiseEx functions use the USER_OTHER_INFO structure to specify information about an event or condition of interest to a user.

ADMIN_OTHER_INFO structure (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **ADMIN_OTHER_INFO** structure contains error message information. The NetAlertRaise and NetAlertRaiseEx functions use the **ADMIN_OTHER_INFO** structure to specify information when raising an administrator's interrupting message.

Syntax

```
typedef struct _ADMIN_OTHER_INFO {
  DWORD alrtad_errcode;
  DWORD alrtad_numstrings;
} ADMIN_OTHER_INFO, *PADMIN_OTHER_INFO, *LPADMIN_OTHER_INFO;
```

Members

`alrtad_errcode`

Specifies the error code for the new message written to the message log.

`alrtad_numstrings`

Specifies the number (0-9) of consecutive Unicode strings written to the message log.

Remarks

Variable-length data follows the **ADMIN_OTHER_INFO** structure in the alert message buffer if you specify one or more strings in the **alrtad_numstrings** member. The calling application must allocate and free the memory for all structures and variable-length data in an alert message buffer.

See NetAlertRaise and NetAlertRaiseEx for code samples that demonstrate how to raise an administrative alert.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmalert.h (include Lm.h)

See also

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaise

NetAlertRaiseEx

ALERT_OTHER_INFO macro (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **ALERT_OTHER_INFO** macro returns a pointer to the alert-specific data in an alert message. The data follows a STD_ALERT structure, and can be an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.

Syntax

```
void ALERT_OTHER_INFO(
   x
);
```

Parameters

`x`

Pointer to a **STD_ALERT** structure that was specified in a call to the NetAlertRaise function.

Return value

None

Remarks

The **ALERT_OTHER_INFO** macro is defined as follows:

```
#include <windows.h>

#define ALERT_OTHER_INFO(x)    ((LPBYTE)(x) + sizeof(STD_ALERT))
```

See NetAlertRaise for a code sample that uses the **ALERT_OTHER_INFO** macro to retrieve a pointer to the **ADMIN_OTHER_INFO** structure.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

ALERT_VAR_DATA

Alert Functions

NetAlertRaise

Network Management Macros

Network Management Overview

PRINT_OTHER_INFO

STD_ALERT

USER_OTHER_INFO

ALERT_VAR_DATA macro (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **ALERT_VAR_DATA** macro returns a pointer to the variable-length portion of an alert message. Variable-length data can follow an ADMIN_OTHER_INFO, a PRINT_OTHER_INFO, or a USER_OTHER_INFO structure.

Syntax

```
void ALERT_VAR_DATA(
   p
);
```

Parameters

`p`

Pointer to an **ADMIN_OTHER_INFO**, a **PRINT_OTHER_INFO**, or a **USER_OTHER_INFO** structure that was specified in a call to the NetAlertRaise function or the NetAlertRaiseEx function.

Return value

None

Remarks

The **ALERT_VAR_DATA** macro is defined as follows:

```
#include <windows.h>

#define ALERT_VAR_DATA(p)      ((LPBYTE)(p) + sizeof(*p))
```

See NetAlertRaise and NetAlertRaiseEx for code samples that use the **ALERT_VAR_DATA** macro to retrieve a pointer to the variable-length data in an alert message.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

ALERT_OTHER_INFO

Alert Functions

NetAlertRaise

NetAlertRaiseEx

Network Management Macros

Network Management Overview

PRINT_OTHER_INFO

USER_OTHER_INFO

ERRLOG_OTHER_INFO structure (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **ERRLOG_OTHER_INFO** structure contains error log information. The NetAlertRaise and NetAlertRaiseEx functions use the **ERRLOG_OTHER_INFO** structure to specify information when adding a new entry to the error log.

Syntax

```
typedef struct _ERRLOG_OTHER_INFO {
  DWORD alrter_errcode;
  DWORD alrter_offset;
} ERRLOG_OTHER_INFO, *PERRLOG_OTHER_INFO, *LPERRLOG_OTHER_INFO;
```

Members

`alrter_errcode`

Specifies the error code that was written to the error log.

`alrter_offset`

Specifies the offset for the new entry in the error log.

Remarks

The calling application must allocate and free the memory for all structures and variable-length data in an alert message buffer.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

Alert Functions

NetAlertRaise

NetAlertRaiseEx

Network Management Overview

Network Management Structures

PRINT_OTHER_INFO

STD_ALERT

USER_OTHER_INFO

NetAlertRaise function (lmalert.h)

2/1/2021 • 4 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the alerter service is not supported.]

The **NetAlertRaise** function notifies all registered clients when a particular event occurs.

To simplify sending an alert message, you can call the extended function NetAlertRaiseEx instead. **NetAlertRaiseEx** does not require that you specify a STD_ALERT structure.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAlertRaise(
  LPCWSTR AlertType,
  LPVOID  Buffer,
  DWORD   BufferSize
);
```

Parameters

`AlertType`

A pointer to a constant string that specifies the alert class (type of alert) to raise. This parameter can be one of the following predefined values, or a user-defined alert class for network applications. The event name for an alert can be any text string.

NAME	MEANING
ALERT_ADMIN_EVENT	An administrator's intervention is required.
ALERT_ERRORLOG_EVENT	An entry was added to the error log.
ALERT_MESSAGE_EVENT	A user or application received a broadcast message.
ALERT_PRINT_EVENT	A print job completed or a print error occurred.
ALERT_USER_EVENT	An application or resource was used.

`Buffer`

A pointer to the data to send to the clients listening for the interrupting message. The data should begin with a fixed-length STD_ALERT structure followed by additional message data in one ADMIN_OTHER_INFO, ERRLOG_OTHER_INFO, PRINT_OTHER_INFO, or USER_OTHER_INFO structure. Finally, the buffer should include any required variable-length information. For more information, see the code sample in the following Remarks section.

The calling application must allocate and free the memory for all structures and variable data. For more information, see Network Management Function Buffers.

`BufferSize`

The size, in bytes, of the message buffer.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code and a can be one of the following error codes. For a list of all possible error codes, see System Error Codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *AlertEventName* parameter is **NULL** or an empty string, the *Buffer* parameter is **NULL**, or the *BufferSize* parameter is less than the size of the STD_ALERT structure plus the fixed size for the additional message data structure.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned on Windows Vista and later since the Alerter service is not supported.

Remarks

No special group membership is required to successfully execute the **NetAlertRaise** function.

The alerter service must be running on the client computer when you call the **NetAlertRaise** function, or the function fails with ERROR_FILE_NOT_FOUND.

Examples

The following code sample demonstrates how to raise an administrative alert by calling the **NetAlertRaise** function and specifying STD_ALERT and ADMIN_OTHER_INFO structures. First, the sample calculates the size of the message buffer. Then it allocates the buffer with a call to the GlobalAlloc function. The code assigns values to the members of the **STD_ALERT** and the **ADMIN_OTHER_INFO** portions of the buffer. The sample retrieves a pointer to the **ADMIN_OTHER_INFO** structure by calling the ALERT_OTHER_INFO macro. It also retrieves a pointer to the variable data portion of the buffer by calling the ALERT_VAR_DATA macro. Finally, the code sample frees the memory allocated for the buffer with a call to the GlobalFree function.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <windows.h>
#include <stdio.h>
#include <time.h>
#include <lm.h>

const int ALERT_VAR_DATA_SIZE = 216;

int wmain(int argc, wchar_t *argv[])
{
   int nBufferSize;
   LPVOID pAlertOtherInfo;
   PSTD_ALERT pStdAlert;              // STD_ALERT structure
   PADMIN_OTHER_INFO pAdminOtherInfo; // ADMIN_OTHER_INFO structure
```

```
LPVOID pVarData;
time_t now;
DWORD dwResult;
//
// Check command line arguments.
//
if (argc != 2)
{
   fwprintf(stderr, L"Usage: %s LogFileName\n", argv[0]);
   exit(1);
}
// Calculate the buffer size;
//  then allocate the memory for the buffer.
//
nBufferSize  = sizeof(STD_ALERT) + ALERT_VAR_DATA_SIZE;
pAlertOtherInfo = (LPVOID) GlobalAlloc(GPTR, nBufferSize);

if (pAlertOtherInfo == NULL)
{
    fwprintf(stderr, L"Unable to allocate memory\n");
    exit(1);
}

//
// Assign values to the STD_ALERT portion of the buffer.
//   (This is required when you call NetAlertRaise.)
//
pStdAlert = (PSTD_ALERT)pAlertOtherInfo;
time( &now );
pStdAlert->alrt_timestamp = (DWORD)now;
wcscpy_s(pStdAlert->alrt_eventname, EVLEN + 1, ALERT_ADMIN_EVENT);
wcscpy_s(pStdAlert->alrt_servicename, SNLEN + 1, argv[0]);
//
// Retrieve the pointer to the ADMIN_OTHER_INFO structure
//  that follows the STD_ALERT portion of the buffer.
//   Do this by calling the ALERT_OTHER_INFO macro.
//
pAdminOtherInfo = (PADMIN_OTHER_INFO)ALERT_OTHER_INFO(pAlertOtherInfo);
//
// Assign values to the ADMIN_OTHER_INFO structure.
//
pAdminOtherInfo->alrtad_numstrings = 1;
//
// Error 2377, NERR_LogOverflow, indicates
//  a log file is full.
//
pAdminOtherInfo->alrtad_errcode = 2377;
//
// Retrieve the pointer to the variable data portion
//  of the buffer by calling the ALERT_VAR_DATA macro.
//
pVarData = (LPTSTR)ALERT_VAR_DATA(pAdminOtherInfo);
//
// Supply the log file name for error 2377.
//
wcsncpy_s((wchar_t*) pVarData, ALERT_VAR_DATA_SIZE/2,
        argv[1],
        ALERT_VAR_DATA_SIZE/2 );
//
// Send an administrative alert by calling the
//  NetAlertRaise function.
//
dwResult = NetAlertRaise(ALERT_ADMIN_EVENT,
                         pAlertOtherInfo,
                         nBufferSize);
//
// Display the results of the function call.
//
if (dwResult != NERR_Success)
```

```
        wprintf(L"NetAlertRaise failed: %d\n", dwResult);
    else
        wprintf(L"Administrative alert raised successfully.\n");
    //
    // Free the allocated memory.
    //
    GlobalFree(pAlertOtherInfo);

    return (dwResult);
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmalert.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

ADMIN_OTHER_INFO

ALERT_OTHER_INFO

ALERT_VAR_DATA

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaiseEx

Network Management Functions

Network Management Overview

PRINT_OTHER_INFO

STD_ALERT

USER_OTHER_INFO

NetAlertRaiseEx function (lmalert.h)

2/1/2021 • 6 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the alerter service is not supported.]

The **NetAlertRaiseEx** function notifies all registered clients when a particular event occurs. You can call this extended function to simplify the sending of an alert message because **NetAlertRaiseEx** does not require that you specify a STD_ALERT structure.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAlertRaiseEx(
  LPCWSTR AlertType,
  LPVOID  VariableInfo,
  DWORD   VariableInfoSize,
  LPCWSTR ServiceName
);
```

Parameters

`AlertType`

A pointer to a constant string that specifies the alert class (type of alert) to raise. This parameter can be one of the following predefined values, or a user-defined alert class for network applications. (The event name for an alert can be any text string.)

NAME	MEANING
ALERT_ADMIN_EVENT	An administrator's intervention is required.
ALERT_ERRORLOG_EVENT	An entry was added to the error log.
ALERT_MESSAGE_EVENT	A user or application received a broadcast message.
ALERT_PRINT_EVENT	A print job completed or a print error occurred.
ALERT_USER_EVENT	An application or resource was used.

`VariableInfo`

A pointer to the data to send to the clients listening for the interrupting message. The data should consist of one ADMIN_OTHER_INFO, ERRLOG_OTHER_INFO, PRINT_OTHER_INFO, or USER_OTHER_INFO structure followed by any required variable-length information. For more information, see the code sample in the following Remarks section.

The calling application must allocate and free the memory for all structures and variable data. For more information, see Network Management Function Buffers.

`VariableInfoSize`

The number of bytes of variable information in the buffer pointed to by the *VariableInfo* parameter.

`ServiceName`

A pointer to a constant string that specifies the name of the service raising the interrupting message.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code and a can be one of the following error codes. For a list of all possible error codes, see System Error Codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *AlertEventName* parameter is **NULL** or an empty string, the *ServiceName* parameter is **NULL** or an empty string, the *VariableInfo* parameter is **NULL**, or the *VariableInfoSize* parameter is greater than 512 minus the size of the STD_ALERT structure.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned on Windows Vista and later since the Alerter service is not supported.

Remarks

No special group membership is required to successfully execute the **NetAlertRaiseEx** function.

The alerter service must be running on the client computer when you call the **NetAlertRaiseEx** function, or the function fails with ERROR_FILE_NOT_FOUND.

Examples

The following code sample demonstrates how to raise the following types of interrupting messages (alerts) by calling the **NetAlertRaiseEx** function:

- An administrative alert by specifying an ADMIN_OTHER_INFO structure
- A print alert by specifying a PRINT_OTHER_INFO structure
- A user alert by specifying a USER_OTHER_INFO structure

In each instance the code assigns values to the members of the relevant alert information structure. Following this, the sample retrieves a pointer to the portion of the message buffer that follows the structure by calling the ALERT_VAR_DATA macro. The code also fills in the variable-length strings in this portion of the buffer. Finally, the sample calls **NetAlertRaiseEx** to send the alert.

Note that the calling application must allocate and free the memory for all structures and variable-length data in an alert message buffer.

To pass a user-defined structure and valid strings in a user alert, you must create an event message file and link it with your application. You must also register the application in the **EventMessageFile** subkey in the **EventLog** section of the registry. If you do not register the application, the user alert will contain the information you pass in the variable-length strings that follow the USER_OTHER_INFO structure. For more information about **EventMessageFile**, see Event Logging.

```
#ifndef UNICODE
#define UNICODE
#endif

#pragma comment(lib, "netapi32.lib")

#include <windows.h>
#include <lm.h>
#include <stdio.h>
#include <time.h>
//
// Define default strings.
//
#define PROGRAM_NAME    TEXT("NETALRT")
#define szComputerName  TEXT("\\\\TESTCOMPUTER")
#define szUserName      TEXT("TEST")
#define szQueueName     TEXT("PQUEUE")
#define szDestName      TEXT("MYPRINTER")
#define szStatus        TEXT("OK")
//
// Define structure sizes.
//
#define VAREDSIZE 312  // maximum size of the variable length message
char buff[VAREDSIZE];
//
int main()
{
   time_t             now;
   PADMIN_OTHER_INFO  pAdminInfo; // ADMIN_OTHER_INFO structure
   PPRINT_OTHER_INFO  pPrintInfo; // PRINT_OTHER_INFO structure
   PUSER_OTHER_INFO   pUserInfo;  // USER_OTHER_INFO structure
   TCHAR              *p;
   DWORD dwResult;

   time( &now );  // Retrieve the current time to print it later.

   //
   // Sending an administrative alert
   //
   // Assign values to the members of the ADMIN_OTHER_INFO structure.
   //
   pAdminInfo = (PADMIN_OTHER_INFO) buff;
   ZeroMemory(pAdminInfo, VAREDSIZE);
   //
   // Error 2377, NERR_LogOverflow, indicates
   //  a log file is full.
   //
   pAdminInfo->alrtad_errcode = 2377;
   pAdminInfo->alrtad_numstrings = 1;
   //
   // Retrieve a pointer to the variable data portion at the
   //  end of the buffer by calling the ALERT_VAR_DATA macro.
   //
   p = (LPTSTR) ALERT_VAR_DATA(pAdminInfo);
   //
   // Fill in the variable-length, concatenated strings
   //  that follow the ADMIN_OTHER_INFO structure. These strings
   //  will be written to the message log.
   //
   wcscpy_s(p,VAREDSIZE/2, TEXT("'C:\\MYLOG.TXT'"));
   //
   // Call the NetAlertRaiseEx function to raise the
   //  administrative alert.
   //
   dwResult = NetAlertRaiseEx(ALERT_ADMIN_EVENT, pAdminInfo, 255 , TEXT("MYSERVICE"));
   //
   // Display the results of the function call.
   //
```

```
if (dwResult != NERR_Success)
{
   wprintf(L"NetAlertRaiseEx failed: %d\n", dwResult);
   return -1;
}
else
   wprintf(L"Administrative alert raised successfully.\n");


//
// Sending a print alert
//
// Assign values to the members of the PRINT_OTHER_INFO structure.
//
pPrintInfo = (PPRINT_OTHER_INFO) buff;
ZeroMemory(pPrintInfo, VAREDSIZE);
pPrintInfo->alrtpr_jobid = 5457;
pPrintInfo->alrtpr_status = 0;
pPrintInfo->alrtpr_submitted = (DWORD) now;
pPrintInfo->alrtpr_size = 1000;
//
// Retrieve a pointer to the variable data portion at the
//  end of the buffer by calling the ALERT_VAR_DATA macro.
//
p = (LPTSTR) ALERT_VAR_DATA(pPrintInfo);
//
// Fill in the variable-length, concatenated strings
//  that follow the PRINT_OTHER_INFO structure.
//
wcscpy_s(p, VAREDSIZE/2, szComputerName); // computername
p += lstrlen(p) + 1;
wcscpy_s(p, (VAREDSIZE/2)-wcslen(szComputerName)-1, szUserName);     // user name
p += lstrlen(p) + 1;
wcscpy_s(p, (VAREDSIZE/2)-wcslen(szComputerName)-wcslen(szUserName)-2,
    szQueueName);    // printer queuename
p += lstrlen(p) + 1;
wcscpy_s(p, (VAREDSIZE/2)-wcslen(szComputerName)-wcslen(szUserName)-wcslen(szQueueName)-3,
    szDestName);     // destination or printer name (optional)
p += lstrlen(p) + 1;
wcscpy_s(p, (VAREDSIZE/2)-wcslen(szComputerName)-wcslen(szUserName)-wcslen(szQueueName)
    - wcslen(szDestName)-4, szStatus);       // status of the print job (optional)
//
// Call the NetAlertRaiseEx function to raise the
//  print alert.
//
dwResult = NetAlertRaiseEx(ALERT_PRINT_EVENT, pPrintInfo, VAREDSIZE, TEXT("MYSERVICE"));
//
// Display the results of the function call.
//
if (dwResult != NERR_Success)
{
   wprintf(L"NetAlertRaiseEx failed: %d\n", dwResult);
   return -1;
}
else
   wprintf(L"Print alert raised successfully.\n");


//
// Sending a user alert
//
// Assign values to the members of the USER_OTHER_INFO structure.
//
pUserInfo  = (PUSER_OTHER_INFO)  buff;
ZeroMemory(pUserInfo, VAREDSIZE);
pUserInfo->alrtus_errcode = 0xffff;
pUserInfo->alrtus_numstrings = 1;
//
// Retrieve a pointer to the variable data portion at the
```

```
    //  end of the buffer by calling the ALERT_VAR_DATA macro.
    //
    p = (LPTSTR) ALERT_VAR_DATA(pUserInfo);
    //
    // Fill in the variable-length, concatenated strings
    //  that follow the USER_OTHER_INFO structure.
    //
    wcscpy_s(p,(VAREDSIZE/2), TEXT("C:\\USERLOG.TXT"));
    p += lstrlen(p) + 1;
    wcscpy_s(p, (VAREDSIZE/2) - wcslen(TEXT("C:\\USERLOG.TXT"))-1, szUserName);
    //
    // Call the NetAlertRaiseEx function to raise the
    //  user alert.
    //
    dwResult = NetAlertRaiseEx(ALERT_USER_EVENT, pUserInfo, VAREDSIZE, TEXT("MYSERVICE"));
    //
    // Display the results of the function call.
    //
    if (dwResult != NERR_Success)
    {
       wprintf(L"NetAlertRaiseEx failed: %d\n", dwResult);
       return -1;
    }
    else
       wprintf(L"User alert raised successfully.\n");

    return(dwResult);
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmalert.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

ADMIN_OTHER_INFO

ALERT_VAR_DATA

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaise

Network Management Functions

Network Management Overview

PRINT_OTHER_INFO

USER_OTHER_INFO

PRINT_OTHER_INFO structure (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **PRINT_OTHER_INFO** structure contains information about a print job. The NetAlertRaise and NetAlertRaiseEx functions use the **PRINT_OTHER_INFO** structure to specify information when a job has finished printing, or when a printer needs intervention.

Syntax

```
typedef struct _PRINT_OTHER_INFO {
  DWORD alrtpr_jobid;
  DWORD alrtpr_status;
  DWORD alrtpr_submitted;
  DWORD alrtpr_size;
} PRINT_OTHER_INFO, *PPRINT_OTHER_INFO, *LPPRINT_OTHER_INFO;
```

Members

`alrtpr_jobid`

Type: **DWORD**

The identification number of the print job.

`alrtpr_status`

Type: **DWORD**

A bitmask describing the status of the print job.

You can obtain the overall status of the job by checking PRJOB_QSTATUS (bits 0 and 1).

Possible values for the print job status are listed in the *Lmalert.h* header file.

VALUE	MEANING
PRJOB_QS_QUEUED 0	The print job is in the queue waiting to be scheduled.
PRJOB_QS_PAUSED 1	The print job is in the queue, but it has been paused. (When a job is paused, it cannot be scheduled.)
PRJOB_QS_SPOOLING 2	The print job is in the process of being spooled.
PRJOB_QS_PRINTING 3	The job is currently printing.

If the print job is in the PRJOB_QS_PRINTING state, you can check bits 2 through 8 for the device's status (PRJOB_DEVSTATUS). Bit 15 is also meaningful.

Possible values for the device's status are listed in the *Lmalert.h* header file.

VALUE	MEANING
PRJOB_COMPLETE 0x4	The job has completed printing.
PRJOB_INTERV 0x8	The destination printer requires an operator's intervention.
PRJOB_ERROR 0x10	There is an error at the destination printer.
PRJOB_DESTOFFLINE 0x20	The destination printer is offline.
PRJOB_DESTPAUSED 0x40	The destination printer is paused.
PRJOB_NOTIFY 0x80	A printing alert should be raised.
PRJOB_DESTNOPAPER 0x100	The destination printer is out of paper.
PRJOB_DELETED 0x8000	The printing job is being deleted.

`alrtpr_submitted`

Type: **DWORD**

The time at which the print job was submitted. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT.

`alrtpr_size`

Type: **DWORD**

The size, in bytes, of the print job.

Remarks

Additional variable-length data follows the **PRINT_OTHER_INFO** structure in the alert message buffer. The information is in the form of contiguous null-terminated character strings, as follows.

STRING	MEANING
computername	The computer that submitted the print job.
username	The user who submitted the print job.
queuename	The print queue to which the job was submitted.
destination	The printer destination (device) to which the print job was routed.
status	The status of the print job.

The calling application must allocate and free the memory for all structures and variable-length data in an alert message buffer.

See NetAlertRaiseEx for a code sample that demonstrates how to raise a print alert.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaise

NetAlertRaiseEx

Network Management Overview

Network Management Structures

STD_ALERT

USER_OTHER_INFO

STD_ALERT structure (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **STD_ALERT** structure contains the time and date when a significant event occurred. The structure also contains an alert class and the name of the application that is raising the alert message. You must specify the **STD_ALERT** structure when you send an alert message using the NetAlertRaise function.

Syntax

```
typedef struct _STD_ALERT {
  DWORD alrt_timestamp;
  WCHAR alrt_eventname[EVLEN + 1];
  WCHAR alrt_servicename[SNLEN + 1];
} STD_ALERT, *PSTD_ALERT, *LPSTD_ALERT;
```

Members

`alrt_timestamp`

Type: **DWORD**

The time and date of the event. This value is stored as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT.

`alrt_eventname`

Type: **WCHAR[EVLEN + 1]**

A Unicode string indicating the alert class (type of event). This parameter can be one of the following predefined values, or another alert class that you have defined for network applications. (The event name for an alert can be any text string.)

NAME	MEANING
ALERT_ADMIN_EVENT	An administrator's intervention is required.
ALERT_ERRORLOG_EVENT	An entry was added to the error log.
ALERT_MESSAGE_EVENT	A user or application received a broadcast message.
ALERT_PRINT_EVENT	A print job completed or a print error occurred.
ALERT_USER_EVENT	An application or resource was used.

`alrt_servicename`

Type: **WCHAR[SNLEN + 1]**

A Unicode string indicating the service application that is raising the alert message.

Remarks

The **STD_ALERT** structure must be followed by one ADMIN_OTHER_INFO, ERRLOG_OTHER_INFO, PRINT_OTHER_INFO, or USER_OTHER_INFO structure. These structures can optionally be followed by variable-length data. The calling application must allocate the memory for all structures and variable-length data in an alert message buffer.

See NetAlertRaise for a code sample that raises an administrative alert using a **STD_ALERT** structure and an ADMIN_OTHER_INFO structure.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaise

Network Management Overview

Network Management Structures

PRINT_OTHER_INFO

USER_OTHER_INFO

USER_OTHER_INFO structure (lmalert.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USER_OTHER_INFO** structure contains user error code information. The NetAlertRaise and NetAlertRaiseEx functions use the **USER_OTHER_INFO** structure to specify information about an event or condition of interest to a user.

Syntax

```
typedef struct _USER_OTHER_INFO {
  DWORD alrtus_errcode;
  DWORD alrtus_numstrings;
} USER_OTHER_INFO, *PUSER_OTHER_INFO, *LPUSER_OTHER_INFO;
```

Members

`alrtus_errcode`

Specifies the error code for the new message in the message log.

`alrtus_numstrings`

Specifies the number (0-9) of consecutive Unicode strings in the message log.

Remarks

Additional variable-length data follows the **USER_OTHER_INFO** structure in the alert message buffer. The information is in the form of contiguous null-terminated character strings, as follows.

STRING	MEANING
username	The user who created the session.
computername	The computer that created the session.

The calling application must allocate and free the memory for all structures and variable-length data in an alert message buffer.

See NetAlertRaiseEx for a code sample that demonstrates how to raise a user alert.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmalert.h (include Lm.h)

See also

ADMIN_OTHER_INFO

Alert Functions

ERRLOG_OTHER_INFO

NetAlertRaise

NetAlertRaiseEx

Network Management Overview

Network Management Structures

PRINT_OTHER_INFO

STD_ALERT

lmapibuf.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmapibuf.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetApiBufferAllocate	The NetApiBufferAllocate function allocates memory from the heap. Use this function only when compatibility with the NetApiBufferFree function is required. Otherwise, use the memory management functions.
NetApiBufferFree	The NetApiBufferFree function frees the memory that the NetApiBufferAllocate function allocates. Applications should also call NetApiBufferFree to free the memory that other network management functions use internally to return information.
NetApiBufferReallocate	The NetApiBufferReallocate function changes the size of a buffer allocated by a previous call to the NetApiBufferAllocate function.
NetApiBufferSize	The NetApiBufferSize function returns the size, in bytes, of a buffer allocated by a call to the NetApiBufferAllocate function.

NetApiBufferAllocate function (lmapibuf.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetApiBufferAllocate** function allocates memory from the heap. Use this function only when compatibility with the NetApiBufferFree function is required. Otherwise, use the memory management functions.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetApiBufferAllocate(
  DWORD  ByteCount,
  LPVOID *Buffer
);
```

Parameters

`ByteCount`

Number of bytes to be allocated.

`Buffer`

Receives a pointer to the allocated buffer.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to successfully execute the ApiBuffer functions.

For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Examples

The following code sample demonstrates how to use the network management ApiBuffer functions.

The sample first calls the **NetApiBufferAllocate** function to allocate memory and then the NetApiBufferSize function to retrieve the size of the allocated memory. Following this, the sample calls NetApiBufferReallocate to change the size of the memory allocation. Finally, the sample calls NetApiBufferFree to free the memory. In each case, the sample prints a message indicating success or failure.

```
#ifndef UNICODE
#define UNICODE
#endif

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

#pragma comment(lib, "netapi32.lib")
```

```
void PrintError(LPSTR lpszApi, DWORD res);

int main()
{
   PUSER_INFO_10 p;
   DWORD res, dwSize;

   // Call the NetApiBufferAllocate function
   //   to allocate the memory. If successful,
   //   print a message.
   //
   res = NetApiBufferAllocate(1024, (LPVOID *) &p);
   if(res == NERR_Success)
   {
      printf("NetApiBufferAllocate:   Allocated 1024 bytes.\n");

      // Call the NetApiBufferSize function
      //   to retrieve the size of the allocated buffer.
      //   If successful, print the size.
      //
      res = NetApiBufferSize(p, &dwSize);
      if(res == NERR_Success)
      {
         printf("NetApiBufferSize:       Buffer has %u bytes.\n", dwSize);

         // Call the NetApiBufferReallocate function
         //   to change the size of the allocated memory.
         //   If successful, print the new size of the buffer.
         //
         res = NetApiBufferReallocate(p, dwSize * 2, (LPVOID *) &p);
         if(res == NERR_Success)
            printf("NetApiBufferReallocate: Re-Allocated %u bytes.\n", dwSize * 2);
         else
            PrintError("NetApiBufferReallocate", res);

         // Call the NetApiBufferFree function
         //    to free the allocated memory.
         //    If successful, print a message.
         //
         res = NetApiBufferFree(p);
         if(res == NERR_Success)
            printf("Freed Buffer\n");

         else
            PrintError("NetApiBufferFree", res);
      }
      else
         PrintError("NetApiBufferSize", res);
   }
   else
      PrintError("NetApiBufferAllocate", res);
   return 0;
}

void PrintError(LPSTR lpszApi, DWORD res)
{
   printf("%s: Error %u\n", lpszApi, res);
   return;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmapibuf.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Api Buffer Functions

NetApiBufferFree

NetApiBufferReallocate

Network Management Functions

Network Management Overview

NetApiBufferFree function (lmapibuf.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetApiBufferFree** function frees the memory that the NetApiBufferAllocate function allocates. Applications should also call **NetApiBufferFree** to free the memory that other network management functions use internally to return information.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetApiBufferFree(
  _Frees_ptr_opt_ LPVOID Buffer
);
```

Parameters

`Buffer`

A pointer to a buffer returned previously by another network management function or memory allocated by calling the NetApiBufferAllocate function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

The **NetApiBufferFree** function is used to free memory used by network management functions. This function is used in two cases:

- To free memory explicitly allocated by calls in an application to the NetApiBufferAllocate function when the memory is no longer needed.
- To free memory allocated internally by calls in an application to remotable network management functions that return information to the caller. The RPC run-time library internally allocates the buffer containing the return information.

Many network management functions retrieve information and return this information as a buffer that may contain a complex structure, an array of structures, or an array of nested structures. These functions use the RPC run-time library to internally allocate the buffer containing the return information, whether the call is to a local computer or a remote server. For example, the NetServerEnum function retrieves a lists of servers and returns this information as an array of structures pointed to by the *bufptr* parameter. When the function is successful, memory is allocated internally by the **NetServerEnum** function to store the array of structures returned in the *bufptr* parameter to the application. When this array of structures is no longer needed, the **NetApiBufferFree** function should be called by the application with the *Buffer* parameter set to the *bufptr* parameter returned by **NetServerEnum** to free this internal memory used. In these cases, the **NetApiBufferFree** function frees all of the internal memory allocated for the buffer including memory for nested structures, pointers to strings, and other data.

No special group membership is required to successfully execute the **NetApiBufferFree** function or any of the

other ApiBuffer functions.

For a code sample that demonstrates how to use of the **NetApiBufferFree** function to free memory explicitly allocated by an application, see the NetApiBufferAllocate function.

For a code sample that demonstrates how to use of the **NetApiBufferFree** function to free memory internally allocated by a network management function to return information, see the NetServerEnum function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmapibuf.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Api Buffer Functions

NetApiBufferAllocate

NetApiBufferReallocate

NetApiBufferSize

Network Management Functions

Network Management Overview

Network Management Function Buffer Lengths

Network Management Function Buffers

NetApiBufferReallocate function (lmapibuf.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetApiBufferReallocate** function changes the size of a buffer allocated by a previous call to the NetApiBufferAllocate function.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetApiBufferReallocate(
  _Frees_ptr_opt_ LPVOID OldBuffer,
  DWORD                  NewByteCount,
  LPVOID                 *NewBuffer
);
```

Parameters

`OldBuffer`

Pointer to the buffer returned by a call to the NetApiBufferAllocate function.

`NewByteCount`

Specifies the new size of the buffer, in bytes.

`NewBuffer`

Receives the pointer to the reallocated buffer.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to successfully execute the ApiBuffer functions.

For a code sample that demonstrates how to use the network management ApiBuffer functions, see NetApiBufferAllocate.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows

Header	lmapibuf.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Api Buffer Functions

NetApiBufferAllocate

NetApiBufferFree

Network Management Functions

Network Management Overview

NetApiBufferSize function (lmapibuf.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetApiBufferSize** function returns the size, in bytes, of a buffer allocated by a call to the NetApiBufferAllocate function.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetApiBufferSize(
  LPVOID  Buffer,
  LPDWORD ByteCount
);
```

Parameters

`Buffer`

Pointer to a buffer returned by the NetApiBufferAllocate function.

`ByteCount`

Receives the size of the buffer, in bytes.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to successfully execute the ApiBuffer functions.

For a code sample that demonstrates how to use the network management ApiBuffer functions, see NetApiBufferAllocate.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmapibuf.h (include Lm.h)
Library	Netapi32.lib

DLL	Netapi32.dll

See also

Api Buffer Functions

NetApiBufferAllocate

NetApiBufferFree

Network Management Functions

Network Management Overview

lmat.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmat.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetScheduleJobAdd	The NetScheduleJobAdd function submits a job to run at a specified future time and date. This function requires that the schedule service be started on the computer to which the job is submitted.
NetScheduleJobDel	The NetScheduleJobDel function deletes a range of jobs queued to run at a computer. This function requires that the schedule service be started at the computer to which the job deletion request is being sent.
NetScheduleJobEnum	The NetScheduleJobEnum function lists the jobs queued on a specified computer. This function requires that the schedule service be started.
NetScheduleJobGetInfo	The NetScheduleJobGetInfo function retrieves information about a particular job queued on a specified computer. This function requires that the schedule service be started.

Structures

TITLE	DESCRIPTION
AT_ENUM	The AT_ENUM structure contains information about a submitted job. The NetScheduleJobEnum function uses this structure to enumerate and return information about an entire queue of submitted jobs.
AT_INFO	The AT_INFO structure contains information about a job.

AT_ENUM structure (lmat.h)

2/1/2021 • 2 minutes to read • Edit Online

The **AT_ENUM** structure contains information about a submitted job. The NetScheduleJobEnum function uses this structure to enumerate and return information about an entire queue of submitted jobs.

Syntax

```
typedef struct _AT_ENUM {
  DWORD     JobId;
  DWORD_PTR JobTime;
  DWORD     DaysOfMonth;
  UCHAR     DaysOfWeek;
  UCHAR     Flags;
  LPWSTR    Command;
} AT_ENUM, *PAT_ENUM, *LPAT_ENUM;
```

Members

`JobId`

Type: **DWORD**

The job identifier of a submitted (queued) job.

`JobTime`

Type: **DWORD_PTR**

A pointer to the time of day at which the job is scheduled to run. The time is the local time at a computer on which the schedule service is running; it is measured from midnight, and is expressed in milliseconds.

`DaysOfMonth`

Type: **DWORD**

A set of bit flags representing the days of the month. For each bit that is set, the scheduled job will run at the time specified by the **JobTime** member, on the corresponding day of the month. Bit 0 corresponds to the first day of the month, and so on.

The value of the bitmask is zero if the job was scheduled to run only once, at the first occurrence specified in the **JobTime** member

`DaysOfWeek`

Type: **UCHAR**

A set of bit flags representing the days of the week. For each bit that is set, the scheduled job will run at the time specified by the **JobTime** member, on the corresponding day of the week. Bit 0 corresponds to Monday, and so on.

The value of the bitmask is zero if the job was scheduled to run only once, at the first occurrence specified in the **JobTime** member.

`Flags`

Type: **UCHAR**

A set of bit flags describing job properties. This member can be one or more of the following values.

VALUE	MEANING
JOB_RUN_PERIODICALLY	This flag is equal to its original value, that is, the value when the job was submitted.
JOB_EXEC_ERROR	If this flag is set, it indicates that the schedule service failed to successfully execute the job the last time it was scheduled to run.
JOB_RUNS_TODAY	If this flag is set, it indicates that the job is scheduled to execute on the current day; the value of the **JobTime** member is greater than the current time of day at the computer where the job is queued.
JOB_NONINTERACTIVE	This flag is equal to its original value, that is, the value when the job was submitted.

`Command`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the command, batch program, or binary file to execute.

Remarks

For more information about setting the bit flags to schedule jobs that execute once, jobs that execute multiple times, and jobs that execute periodically without deletion, see the NetScheduleJobAdd function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmat.h (include Lm.h)

See also

NetScheduleJobEnum

Network Management Overview

Network Management Structures

Schedule Functions

AT_INFO structure (lmat.h)

2/1/2021 • 3 minutes to read • Edit Online

The **AT_INFO** structure contains information about a job. The NetScheduleJobAdd function uses the structure to specify information when scheduling a job. The NetScheduleJobGetInfo function uses the structure to retrieve information about a job that has already been submitted.

Syntax

```
typedef struct _AT_INFO {
  DWORD_PTR JobTime;
  DWORD     DaysOfMonth;
  UCHAR     DaysOfWeek;
  UCHAR     Flags;
  LPWSTR    Command;
} AT_INFO, *PAT_INFO, *LPAT_INFO;
```

Members

`JobTime`

Type: **DWORD_PTR**

A pointer to a value that indicates the time of day at which the job is scheduled to run. The time is the local time at a computer on which the schedule service is running; it is measured from midnight, and is expressed in milliseconds.

`DaysOfMonth`

Type: **DWORD**

A set of bit flags representing the days of the month. For each bit that is set, the scheduled job will run at the time specified by the **JobTime** member, on the corresponding day of the month. Bit 0 corresponds to the first day of the month, and so on.

The value of the bitmask is zero if the job was scheduled to run only once, at the first occurrence specified by the **JobTime** member.

`DaysOfWeek`

Type: **UCHAR**

A set of bit flags representing the days of the week. For each bit that is set, the scheduled job will run at the time specified by the **JobTime** member, on the corresponding day of the week. Bit 0 corresponds to Monday, and so on.

The value of the bitmask is zero if the job was scheduled to run only once, at the first occurrence specified by the **JobTime** member.

`Flags`

Type: **UCHAR**

A set of bit flags describing job properties.

When you submit a job using a call to the NetScheduleJobAdd function, you can specify one of the following values.

VALUE	MEANING
JOB_RUN_PERIODICALLY	If you set this flag, the job runs, and continues to run, on each day for which a corresponding bit is set in the **DaysOfMonth** member or the **DaysOfWeek** member. The job is not deleted after it executes. If this flag is clear, the job runs only once for each bit set in these members. The job is deleted after it executes once.
JOB_ADD_CURRENT_DATE	If you set this flag, the job executes at the first occurrence of **JobTime** member at the computer where the job is queued. Setting this flag is equivalent to setting the bit for the current day in the **DaysOfMonth** member.
JOB_NONINTERACTIVE	If you set this flag, the job does not run interactively. If this flag is clear, the job runs interactively.

When you call NetScheduleJobGetInfo to retrieve job information, the function can return one or more of the following values.

VALUE	MEANING
JOB_RUN_PERIODICALLY	This flag is equal to its original value, that is, the value when the job was submitted.
JOB_EXEC_ERROR	If this flag is set, it indicates that the schedule service failed to successfully execute the job the last time it was scheduled to run.
JOB_RUNS_TODAY	If this flag is set, it indicates that the job is scheduled to execute on the current day; the value of the **JobTime** member is greater than the current time of day at the computer where the job is queued.
JOB_NONINTERACTIVE	This flag bit is equal to its original value, that is, the value when the job was submitted.

`Command`

Type: **LPWSTR**

A pointer to a Unicode string that contains the name of the command, batch program, or binary file to execute.

Remarks

For more information about scheduling jobs that execute once, jobs that execute multiple times, and jobs that execute periodically without deletion, see NetScheduleJobAdd.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmat.h (include Lm.h)

See also

NetScheduleJobAdd

NetScheduleJobGetInfo

Network Management Overview

Network Management Structures

Schedule Functions

NetScheduleJobAdd function (lmat.h)

2/1/2021 • 3 minutes to read • Edit Online

[**NetScheduleJobAdd** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **NetScheduleJobAdd** function submits a job to run at a specified future time and date. This function requires that the schedule service be started on the computer to which the job is submitted.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetScheduleJobAdd(
  LPCWSTR Servername,
  LPBYTE  Buffer,
  LPDWORD JobId
);
```

Parameters

`Servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`Buffer`

A pointer to an AT_INFO structure describing the job to submit. For more information about scheduling jobs using different job properties, see the following Remarks section and Network Management Function Buffers.

`JobId`

A pointer that receives a job identifier for the newly submitted job. This entry is valid only if the function returns successfully.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

Normally only members of the local Administrators group on the computer where the schedule job is being added can successfully execute this function. If the server name passed in the string pointed to by the *Servername* parameter is a remote server, then only members of the local Administrators group on the remote server can successfully execute this function.

If the following registry value has the least significant bit set (for example, 0x00000001), then users belonging to the Server Operators group can also successfully execute this function.

HKLM\System\CurrentControlSet\Control\Lsa\SubmitControl

The following are examples of how to schedule jobs using different properties supported by the **NetScheduleJobAdd** function.

To schedule a job that executes once:

- Set the **DaysOfMonth** member of the AT_INFO structure to zero.
- Set the **DaysOfWeek** member of the AT_INFO structure to zero.
- Set the **JobTime** member of the AT_INFO structure to the time the job should execute.

The job executes at the time specified by the **JobTime** member of the AT_INFO structure pointed to by the *Buffer* parameter. After the job executes, it is deleted.

To schedule and delete a job that executes multiple times:

- Set the appropriate bits in the **DaysOfMonth** member of the AT_INFO structure or
- Set the appropriate bits in the **DaysOfWeek** member of the AT_INFO structure.
- Set the **JobTime** member of the AT_INFO structure to the time the job should execute.

> **Note** You do not need to set both the **DaysOfMonth** and the **DaysOfWeek** members of the AT_INFO structure.

The job executes at the time specified by the **JobTime** member of the AT_INFO structure pointed to by the *Buffer* parameter, once for each day set in the **DaysOfMonth** or **DaysOfWeek** members of the AT_INFO structure. After each job executes, the corresponding bit is cleared. When the last bit is cleared, the job is deleted.

To schedule a job that executes periodically:

- Set the appropriate bits in the **DaysOfMonth** member of the AT_INFO structure or
- Set the appropriate bits in the **DaysOfWeek** member of the AT_INFO structure.
- Set the **JobTime** member of the AT_INFO structure to the time the job should execute.
- Set the job submission flag JOB_RUN_PERIODICALLY in the **Flags** member of the AT_INFO structure.

> **Note** You do not need to set both the **DaysOfMonth** and the **DaysOfWeek** members of the AT_INFO structure.

The job will execute periodically, at the time specified by the **JobTime** member of the AT_INFO structure pointed to by the *Buffer* parameter, on each day set in the **DaysOfMonth** or **DaysOfWeek** member of the AT_INFO structure. The job will not be deleted as a result of the repeated executions. The only way to delete the job is by an explicit call to the NetScheduleJobDel function.

See the AT_INFO structure for a description of the **DaysOfWeek**, **DaysOfMonth**, and job property bitmasks.

On Windows 2000, the earlier AT service and the Task Scheduler were combined. The Task Scheduler service was only accurate to the minute. Therefore, the **NetScheduleJobAdd** function only uses hours and minutes specified in the **JobTime** member of the AT_INFO structure when a job is scheduled to run.

Starting with Windows Vista, the precision for the Task Scheduler was increased to the second. Therefore, the **NetScheduleJobAdd** function uses only the hours, minutes, and seconds specified in the **JobTime** member of the AT_INFO structure when a job is scheduled to run.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmat.h (include Lmat.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

AT_INFO

NetScheduleJobDel

NetScheduleJobEnum

NetScheduleJobGetInfo

Network Management Functions

Network Management Overview

Schedule Functions

NetScheduleJobDel function (lmat.h)

2/1/2021 • 2 minutes to read • Edit Online

[**NetScheduleJobDel** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **NetScheduleJobDel** function deletes a range of jobs queued to run at a computer. This function requires that the schedule service be started at the computer to which the job deletion request is being sent.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetScheduleJobDel(
  LPCWSTR Servername,
  DWORD   MinJobId,
  DWORD   MaxJobId
);
```

Parameters

`Servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`MinJobId`

The minimum job identifier. Jobs with a job identifier smaller than *MinJobId* will not be deleted.

`MaxJobId`

The maximum job identifier. Jobs with a job identifier larger than *MaxJobId* will not be deleted.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

Normally only members of the local Administrators group on the computer where the schedule job is being deleted can successfully execute this function. If the server name passed in the string pointed to by the *Servername* parameter is a remote server, then only members of the local Administrators group on the server can successfully execute this function.

If the following registry value has the least significant bit set (for example, 0x00000001), then users belonging to the Server Operators group can also successfully execute this function.

HKLM\System\CurrentControlSet\Control\Lsa\SubmitControl

Call the NetScheduleJobEnum function to retrieve the job identifier for one or more scheduled jobs.

The **NetScheduleJobDel** function deletes all jobs whose job identifiers are in the range *MinJobId* through *MaxJobId*.

To delete all scheduled jobs at the server, you can call **NetScheduleJobDel** specifying *MinJobId* equal to 0 and *MaxJobId* equal to – 1. To delete one job, specify the job's identifier for both the *MinJobId* parameter and the *MaxJobId* parameter.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmat.h (include Lmat.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetScheduleJobAdd

NetScheduleJobEnum

NetScheduleJobGetInfo

Network Management Functions

Network Management Overview

Schedule Functions

NetScheduleJobEnum function (lmat.h)

2/1/2021 • 2 minutes to read • Edit Online

[**NetScheduleJobEnum** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **NetScheduleJobEnum** function lists the jobs queued on a specified computer. This function requires that the schedule service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetScheduleJobEnum(
  LPCWSTR Servername,
  LPBYTE  *PointerToBuffer,
  DWORD   PrefferedMaximumLength,
  LPDWORD EntriesRead,
  LPDWORD TotalEntries,
  LPDWORD ResumeHandle
);
```

Parameters

`Servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`PointerToBuffer`

A pointer to the buffer that receives the data. The return information is an array of AT_ENUM structures. The buffer is allocated by the system and must be freed using a single call to the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`PrefferedMaximumLength`

A value that indicates the preferred maximum length of the returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`EntriesRead`

A pointer to a value that receives the count of elements actually enumerated.

`TotalEntries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`ResumeHandle`

A pointer to a value that contains a resume handle which is used to continue a job enumeration. The handle

should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, then no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

Normally only members of the local Administrators group on the computer where the schedule job is being enumerated can successfully execute this function. If the server name passed in the string pointed to by the *Servername* parameter is a remote server, then only members of the local Administrators group on the server can successfully execute this function.

If the following registry value has the least significant bit set (for example, 0x00000001), then users belonging to the Server Operators group can also successfully execute this function.

HKLM\System\CurrentControlSet\Control\Lsa\SubmitControl

Each entry returned contains an AT_ENUM structure. The value of the **JobId** member can be used when calling functions that require a job identifier parameter, such as the NetScheduleJobDel function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmat.h (include Lmat.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

AT_ENUM

NetScheduleJobAdd

NetScheduleJobDel

NetScheduleJobGetInfo

Network Management Functions

Network Management Overview

Schedule Functions

NetScheduleJobGetInfo function (lmat.h)

2/1/2021 • 2 minutes to read • Edit Online

[**NetScheduleJobGetInfo** is no longer available for use as of Windows 8. Instead, use the Task Scheduler 2.0 Interfaces.

]

The **NetScheduleJobGetInfo** function retrieves information about a particular job queued on a specified computer. This function requires that the schedule service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetScheduleJobGetInfo(
  LPCWSTR Servername,
  DWORD   JobId,
  LPBYTE  *PointerToBuffer
);
```

Parameters

`Servername`

A pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`JobId`

A value that indicates the identifier of the job for which to retrieve information.

`PointerToBuffer`

A pointer to the buffer that receives the AT_INFO structure describing the specified job. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

Normally only members of the local Administrators group on the computer where the schedule job is being enumerated can successfully execute this function. If the server name passed in the string pointed to by the *Servername* parameter is a remote server, then only members of the local Administrators group on the server can successfully execute this function.

If the following registry value has the least significant bit set (for example, 0x00000001), then users belonging to the Server Operators group can also successfully execute this function.

HKLM\System\CurrentControlSet\Control\Lsa\SubmitControl

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmat.h (include Lmat.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

AT_INFO

NetScheduleJobAdd

NetScheduleJobDel

NetScheduleJobEnum

Network Management Functions

Network Management Overview

Schedule Functions

lmaudit.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmaudit.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetAuditClear	The NetAuditClear function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetAuditRead	The NetAuditRead function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetAuditWrite	The NetAuditWrite function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

NetAuditClear function (lmaudit.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetAuditClear** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAuditClear(
  LPCWSTR server,
  LPCWSTR backupfile,
  LPCWSTR service
);
```

Parameters

`server`

TBD

`backupfile`

TBD

`service`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmaudit.h

NetAuditRead function (lmaudit.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetAuditRead** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAuditRead(
  LPCWSTR server,
  LPCWSTR service,
  LPHLOG  auditloghandle,
  DWORD   offset,
  LPDWORD reserved1,
  DWORD   reserved2,
  DWORD   offsetflag,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD bytesread,
  LPDWORD totalavailable
);
```

Parameters

`server`

TBD

`service`

TBD

`auditloghandle`

TBD

`offset`

TBD

`reserved1`

TBD

`reserved2`

TBD

`offsetflag`

TBD

`bufptr`

TBD

`prefmaxlen`

TBD

`bytesread`

TBD

`totalavailable`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmaudit.h

NetAuditWrite function (lmaudit.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetAuditWrite** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAuditWrite(
  DWORD   type,
  LPBYTE  buf,
  DWORD   numbytes,
  LPCWSTR service,
  LPBYTE  reserved
);
```

Parameters

`type`

TBD

`buf`

TBD

`numbytes`

TBD

`service`

TBD

`reserved`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmaudit.h

lmconfig.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmconfig.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetConfigGet	The NetConfigGet function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.
NetConfigGetAll	The NetConfigGetAll function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.
NetConfigSet	The NetConfigSet function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.

NetConfigGet function (lmconfig.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetConfigGet** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetConfigGet(
  LPCWSTR server,
  LPCWSTR component,
  LPCWSTR parameter,
  LPBYTE  *bufptr
);
```

Parameters

`server`

TBD

`component`

TBD

`parameter`

TBD

`bufptr`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmconfig.h

NetConfigGetAll function (lmconfig.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetConfigGetAll** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetConfigGetAll(
  LPCWSTR server,
  LPCWSTR component,
  LPBYTE  *bufptr
);
```

Parameters

`server`

TBD

`component`

TBD

`bufptr`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmconfig.h

NetConfigSet function (lmconfig.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetConfigSet** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the registry.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetConfigSet(
  LPCWSTR server,
  LPCWSTR reserved1,
  LPCWSTR component,
  DWORD   level,
  DWORD   reserved2,
  LPBYTE  buf,
  DWORD   reserved3
);
```

Parameters

`server`

TBD

`reserved1`

TBD

`component`

TBD

`level`

TBD

`reserved2`

TBD

`buf`

TBD

`reserved3`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmconfig.h

lmerrlog.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmerrlog.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetErrorLogClear	The NetErrorLogClear function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetErrorLogRead	The NetErrorLogRead function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.
NetErrorLogWrite	The NetErrorLogWrite function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

NetErrorLogClear function (lmerrlog.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetErrorLogClear** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetErrorLogClear(
  LPCWSTR UncServerName,
  LPCWSTR BackupFile,
  LPBYTE  Reserved
);
```

Parameters

`UncServerName`

TBD

`BackupFile`

TBD

`Reserved`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmerrlog.h

NetErrorLogRead function (lmerrlog.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetErrorLogRead** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetErrorLogRead(
  LPCWSTR UncServerName,
  LPWSTR  Reserved1,
  LPHLOG  ErrorLogHandle,
  DWORD   Offset,
  LPDWORD Reserved2,
  DWORD   Reserved3,
  DWORD   OffsetFlag,
  LPBYTE  *BufPtr,
  DWORD   PrefMaxSize,
  LPDWORD BytesRead,
  LPDWORD TotalAvailable
);
```

Parameters

`UncServerName`

TBD

`Reserved1`

TBD

`ErrorLogHandle`

TBD

`Offset`

TBD

`Reserved2`

TBD

`Reserved3`

TBD

`OffsetFlag`

TBD

`BufPtr`

TBD

`PrefMaxSize`

TBD

`BytesRead`

TBD

`TotalAvailable`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmerrlog.h

NetErrorLogWrite function (lmerrlog.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetErrorLogWrite** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use event logging.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetErrorLogWrite(
  LPBYTE  Reserved1,
  DWORD   Code,
  LPCWSTR Component,
  LPBYTE  Buffer,
  DWORD   NumBytes,
  LPBYTE  MsgBuf,
  DWORD   StrCount,
  LPBYTE  Reserved2
);
```

Parameters

`Reserved1`

TBD

`Code`

TBD

`Component`

TBD

`Buffer`

TBD

`NumBytes`

TBD

`MsgBuf`

TBD

`StrCount`

TBD

`Reserved2`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmerrlog.h

lmjoin.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmjoin.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetAddAlternateComputerName	Adds an alternate name for the specified computer.
NetCreateProvisioningPackage	Creates a provisioning package that provisions a computer account for later use in an offline domain join operation. The package may also contain information about certificates and policies to add to the machine during provisioning.
NetEnumerateComputerNames	Enumerates names for the specified computer.
NetFreeAadJoinInformation	Frees the memory allocated for the specified DSREG_JOIN_INFO structure, which contains join information for a tenant and which you retrieved by calling the NetGetAadJoinInformation function.
NetGetAadJoinInformation	Retrieves the join information for the specified tenant. This function examines the join information for Microsoft Azure Active Directory and the work account that the current user added.
NetGetJoinableOUs	The NetGetJoinableOUs function retrieves a list of organizational units (OUs) in which a computer account can be created.
NetGetJoinInformation	The NetGetJoinInformation function retrieves join status information for the specified computer.
NetJoinDomain	The NetJoinDomain function joins a computer to a workgroup or domain.
NetProvisionComputerAccount	Provisions a computer account for later use in an offline domain join operation.
NetRemoveAlternateComputerName	Removes an alternate name for the specified computer.
NetRenameMachineInDomain	The NetRenameMachineInDomain function changes the name of a computer in a domain.
NetRequestOfflineDomainJoin	Executes locally on a machine to modify a Windows operating system image mounted on a volume.

TITLE	DESCRIPTION
NetRequestProvisioningPackageInstall	Executes locally on a machine to modify a Windows operating system image mounted on a volume.
NetSetPrimaryComputerName	Sets the primary computer name for the specified computer.
NetUnjoinDomain	The NetUnjoinDomain function unjoins a computer from a workgroup or a domain.
NetValidateName	The NetValidateName function verifies that a name is valid for name type specified(computer name, workgroup name, domain name, or DNS computer name).

Structures

TITLE	DESCRIPTION
DSREG_JOIN_INFO	Contains information about how a device is joined to Microsoft Azure Active Directory.
DSREG_USER_INFO	Contains information about a user account that is used to join a device to Microsoft Azure Active Directory.
NETSETUP_PROVISIONING_PARAMS	The NETSETUP_PROVISIONING_PARAMS structure contains information that is used when creating a provisioning package using the NetCreateProvisionPackage function.

Enumerations

TITLE	DESCRIPTION
DSREG_JOIN_TYPE	Specifies the possible ways that a device can be joined to Microsoft Azure Active Directory.

DSREG_JOIN_INFO structure (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

Contains information about how a device is joined to Microsoft Azure Active Directory.

Syntax

```
typedef struct _DSREG_JOIN_INFO {
  DSREG_JOIN_TYPE joinType;
  PCCERT_CONTEXT  pJoinCertificate;
  LPWSTR          pszDeviceId;
  LPWSTR          pszIdpDomain;
  LPWSTR          pszTenantId;
  LPWSTR          pszJoinUserEmail;
  LPWSTR          pszTenantDisplayName;
  LPWSTR          pszMdmEnrollmentUrl;
  LPWSTR          pszMdmTermsOfUseUrl;
  LPWSTR          pszMdmComplianceUrl;
  LPWSTR          pszUserSettingSyncUrl;
  DSREG_USER_INFO *pUserInfo;
} DSREG_JOIN_INFO, *PDSREG_JOIN_INFO;
```

Members

`joinType`

An enumeration value that specifies the type of the join.

`pJoinCertificate`

Representations of the certification for the join.

`pszDeviceId`

The identifier of the device.

`pszIdpDomain`

A string that represents Azure Active Directory (Azure AD).

`pszTenantId`

The identifier of the joined Azure AD tenant.

`pszJoinUserEmail`

The email address for the joined account.

`pszTenantDisplayName`

The display name for the joined account.

`pszMdmEnrollmentUrl`

The URL to use to enroll in the Mobile Device Management (MDM) service.

`pszMdmTermsOfUseUrl`

The URL that provides information about the terms of use for the MDM service.

`pszMdmComplianceUrl`

The URL that provides information about compliance for the MDM service.

`pszUserSettingSyncUrl`

The URL for synchronizing user settings.

`pUserInfo`

Information about the user account that was used to join a device to Azure AD.

Requirements

Minimum supported client	Windows 10 [desktop apps only]
Minimum supported server	Windows Server 2016 [desktop apps only]
Header	lmjoin.h

See also

CERT_CONTEXT

DSREG_JOIN_TYPE

DSREG_USER_INFO

DSREG_JOIN_TYPE enumeration (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

Specifies the possible ways that a device can be joined to Microsoft Azure Active Directory.

Syntax

```
typedef enum _DSREG_JOIN_TYPE {
  DSREG_UNKNOWN_JOIN,
  DSREG_DEVICE_JOIN,
  DSREG_WORKPLACE_JOIN
} DSREG_JOIN_TYPE, *PDSREG_JOIN_TYPE;
```

Constants

NAME	DESCRIPTION
DSREG_UNKNOWN_JOIN	The type of join is not known.
DSREG_DEVICE_JOIN	The device is joined to Azure Active Directory (Azure AD).
DSREG_WORKPLACE_JOIN	An Azure AD work account is added on the device.

Requirements

Minimum supported client	Windows 10 [desktop apps only]
Minimum supported server	Windows Server 2016 [desktop apps only]
Header	lmjoin.h

DSREG_USER_INFO structure (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

Contains information about a user account that is used to join a device to Microsoft Azure Active Directory.

Syntax

```
typedef struct _DSREG_USER_INFO {
  LPWSTR pszUserEmail;
  LPWSTR pszUserKeyId;
  LPWSTR pszUserKeyName;
} DSREG_USER_INFO, *PDSREG_USER_INFO;
```

Members

`pszUserEmail`

The email address of the user.

`pszUserKeyId`

The identifier of the Microsoft Passport key that is provisioned for the user.

`pszUserKeyName`

The name of the Microsoft Passport key that is provisioned for the user.

Requirements

Minimum supported client	Windows 10 [desktop apps only]
Minimum supported server	Windows Server 2016 [desktop apps only]
Header	lmjoin.h

NetAddAlternateComputerName function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetAddAlternateComputerName** function adds an alternate name for the specified computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetAddAlternateComputerName(
  LPCWSTR Server,
  LPCWSTR AlternateName,
  LPCWSTR DomainAccount,
  LPCWSTR DomainAccountPassword,
  ULONG   Reserved
);
```

Parameters

`Server`

A pointer to a constant string that specifies the name of the computer on which to execute this function. If this parameter is **NULL**, the local computer is used.

`AlternateName`

A pointer to a constant string that specifies the alternate name to add. This name must be in the form of a fully qualified DNS name.

`DomainAccount`

A pointer to a constant string that specifies the domain account to use for accessing the machine account object for the computer specified in the *Server* parameter in Active Directory. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is not used if the server to execute this function is not joined to a domain.

`DomainAccountPassword`

A pointer to a constant string that specifies the password matching the domain account passed in the *DomainAccount* parameter. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is ignored if the *DomainAccount* parameter is **NULL**. This parameter is not used if the server to execute this function is not joined to a domain.

`Reserved`

Reserved for future use. This parameter should be **NULL**.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller was not a member of the Administrators local group on the target computer.
ERROR_INVALID_NAME	A name parameter is incorrect. This error is returned if the *AlternateName* parameter does not contain valid name.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *DomainAccount* parameter does not contain a valid domain. This error is also returned if the *DomainAccount* parameter is not **NULL** and the *DomainAccountPassword* parameter is not **NULL** but does not contain a Unicode string.
ERROR_NOT_ENOUGH_MEMORY	Not enough memory is available to process this command.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the target computer specified in the *Server* parameter on which this function executes is running on Windows 2000 and earlier.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetAddAlternateComputerName** function is supported on Windows XP and later.

The **NetAddAlternateComputerName** function is used to set secondary network names for computers. The primary name is the name used for authentication and maps to the machine account name.

The **NetAddAlternateComputerName** function requires that the caller is a member of the Administrators local group on the target computer.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)

Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetEnumerateComputerNames

NetJoinDomain

NetRemoveAlternateComputerName

NetRenameMachineInDomain

NetSetPrimaryComputerName

NetUnjoinDomain

SetComputerNameEx

NetCreateProvisioningPackage function (lmjoin.h)

2/1/2021 • 9 minutes to read • Edit Online

The NetCreateProvisioningPackage function creates a provisioning package that provisions a computer account for later use in an offline domain join operation. The package may also contain information about certificates and policies to add to the machine during provisioning.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetCreateProvisioningPackage(
  PNETSETUP_PROVISIONING_PARAMS pProvisioningParams,
  PBYTE                         *ppPackageBinData,
  DWORD                         *pdwPackageBinDataSize,
  LPWSTR                        *ppPackageTextData
);
```

Parameters

`pProvisioningParams`

A pointer to a NETSETUP_PROVISIONING_PARAMS structure that contains information about the provisioning package.

The following values are defined for the members of this structure:

VALUE	MEANING
dwVersion	The version of Windows in the provisioning package. This member should use the following value defined in the *Lmjoin.h* header file: NETSETUP_PROVISIONING_PARAMS_CURRENT_VERSION (0x00000001)
lpDomain	A pointer to a constant null-terminated character string that specifies the name of the domain where the computer account is created.
lpMachineName	A pointer to a constant null-terminated character string that specifies the short name of the machine from which the computer account attribute sAMAccountName is derived by appending a '$'. This parameter must contain a valid DNS or NetBIOS machine name.
lpMachineAccountOU	An optional pointer to a constant null-terminated character string that contains the RFC 1779 format name of the organizational unit (OU) where the computer account will be created. If you specify this parameter, the string must contain a full path, for example, OU=testOU,DC=domain,DC=Domain,DC=com. Otherwise, this parameter must be **NULL**. If this parameter is **NULL**, the well known computer object container will be used as published in the domain.

lpDcName	An optional pointer to a constant null-terminated character string that contains the name of the domain controller to target.
dwProvisionOptions	A set of bit flags that define provisioning options. This parameter can be one or more of the values specified for the *dwOptions* parameter passed to the NetProvisionComputerAccount function. These possible values are defined in the *Lmjoin.h* header file. The **NETSETUP_PROVISION_ROOT_CA_CERTS** option is only supported on Windows 8 and Windows Server 2012.
aCertTemplateNames	A optional pointer to an array of **NULL**-terminated certificate template names.
cCertTemplateNames	When **aCertTemplateNames** is not NULL, this member provides an explicit count of the number of items in the array.
aMachinePolicyNames	An optional pointer to an array of **NULL**-terminated machine policy names.
cMachinePolicyNames	When **aMachinePolicyNames** is not **NULL**, this member provides an explicit count of the number of items in the array.
aMachinePolicyPaths	An optional pointer to an array of character strings. Each array element is a NULL-terminated character string which specifies the full or partial path to a file in the Registry Policy File format. For more information on the Registry Policy File Format , see Registry Policy File Format The path could be a UNC path on a remote server.
cMachinePolicyPaths	When **aMachinePolicyPaths** is not **NULL**, this member provides an explicit count of the number of items in the array.

`ppPackageBinData`

An optional pointer that will receive the package required by NetRequestOfflineDomainJoin function to complete an offline domain join, if the NetProvisionComputerAccount function completes successfully. The data is returned as an opaque binary buffer which may be passed to **NetRequestOfflineDomainJoin** function.

If this parameter is **NULL**, then *pPackageTextData* parameter must not be **NULL**. If this parameter is not **NULL**, then the *pPackageTextData* parameter must be **NULL**.

`pdwPackageBinDataSize`

A pointer to a value that receives the size, in bytes, of the buffer returned in the *pProvisionBinData* parameter.

This parameter must not be **NULL** if the *pPackageBinData* parameter is not **NULL**. This parameter must be **NULL** when the *pPackageBinData* parameter is **NULL**.

`ppPackageTextData`

An optional pointer that will receive the package required by NetRequestOfflineDomainJoin function to complete an offline domain join, if the NetProvisionComputerAccount function completes successfully. The data is returned in string form for embedding in an unattended setup answer file.

If this parameter is **NULL**, then the *pPackageBinData* parameter must not be **NULL**. If this parameter is not **NULL**, then the the *pPackageBinData* parameter must be **NULL**.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller does not have sufficient privileges to complete the operation.
ERROR_INVALID_DOMAIN_ROLE	This operation is only allowed for the Primary Domain Controller of the domain. This error is returned if a domain controller name was specified in the **lpDcName** of the NETSETUP_PROVISIONING_PARAMS struct pointed to by the *pProvisioningParams* parameter, but the computer specified could not be validated as a domain controller for the target domain specified in the **lpDomain** of the **NETSETUP_PROVISIONING_PARAMS**.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is also returned if both the *pProvisioningParams* parameter is **NULL**. This error is also returned if the **lpDomain** or **lpMachineName** member of the NETSETUP_PROVISIONING_PARAMS struct pointed to by the *pProvisioningParams* parameter is **NULL**.
ERROR_NO_SUCH_DOMAIN	The specified domain did not exist.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the **lpMachineAccountOU** member was specified in the NETSETUP_PROVISIONING_PARAMS struct pointed to by the *pProvisioningParams* parameter and the domain controller is running on an earlier versions of Windows that does not support this parameter.
NERR_DS8DCRequired	The specified domain controller does not meet the version requirement for this operation.
NERR_LDAPCapableDCRequired	This operation requires a domain controller which supports LDAP.
NERR_UserExists	The account already exists in the domain and the **NETSETUP_PROVISION_REUSE_ACCOUNT** bit was not specified in the **dwProvisionOptions** member of the NETSETUP_PROVISIONING_PARAMS struct pointed to by the *pProvisioningParams* parameter.
NERR_WkstaNotStarted	The Workstation service has not been started.

RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetCreateProvisioningPackage** function is supported on Windows 8 and Windows Server 2012 for offline join operations. For Windows 7, use the NetProvisionComputerAccount function.

The **NetCreateProvisioningPackage** function is used to provision a computer account for later use in an offline domain join operation using the NetRequestProvisioningPackageInstall function.

The offline domain join scenario uses two functions:

- **NetCreateProvisioningPackage** is a provisioning function that is first called to perform the network operations necessary to create and configure the computer object in Active Directory. The output from the **NetCreateProvisioningPackage** is a package used for the next step.
- NetRequestProvisioningPackageInstall, an image initialization function, is called to inject the output from the **NetCreateProvisioningPackage** provisioning function into a Windows operating system image for use during pre-installation and post-installation.

Changes to Windows initialization code will detect this saved state and affect the local-only portion of domain join.

When the *pPackageBinData* and *pdwPackageBinDataSize* out pointers are used, set the *pPackageTextData* out pointer to NULL. When *pPackageTextData* is used, set the *pPackageBinData* and *pdwPackageBinDataSize* out pointers to NULL.

The *pProvisioningParams* parameter specifies data to include in the provisioning package. The package includes information relevant to the domain join, and it can also include information about policies and certificates to install on the machine. The provisioning package can be used in four ways:

- Domain join
- Domain join and installation of certificates
- Domain join and installation of policies
- Domain join and installation of certificates and policies

The **NetCreateProvisioningPackage** function creates or reuses the machine account in the domain, collects all necessary metadata and returns it in a package. The package can be consumed by the offline domain join request operation supplying all the necessary input to complete the domain join during first boot without any network operations (local state updates only).

Security Note: The package returned by the **NetCreateProvisioningPackage** function contains very sensitive data. It should be treated just as securely as a plaintext password. The package contains the machine account password and other information about the domain, including the domain name, the name of a domain controller, and the security ID (SID) of the domain. If the package is being transported physically or over the network, care must be taken to transport it securely. The design makes no provisions for securing this data. This problem exists today with unattended setup answer files which can carry a number of secrets including domain user passwords. The caller must secure the package. Solutions to this problem are varied. As an example, a pre-exchanged key could be used to encrypt a session between the consumer and provisioning entity enabling a secure transfer of the package.

The package returned in the *pPackageBinData* parameter by the **NetCreateProvisioningPackage** function is versioned to allow interoperability and serviceability scenarios between different versions of Windows (such as

joining a client, provisioning a machine, and using a domain controller). A package created on Windows 8 or Windows Server 2012 can be used Windows 7 or Windows Server 2008 R2, however only domain join information will take effect (certificates and policies are not supported). The offline join scenario currently does not limit the lifetime of the package returned by the **NetCreateProvisioningPackage** function.

For offline domain joins, the access check performed depends on the configuration of the domain. Computer account creation is enabled using three methods:

- Domain administrators have rights to create computer accounts.
- The SD on a container can delegate the rights to create computer accounts.
- By default, authenticated users may create computer accounts by privilege. Authenticated users are limited to creating a limited number of accounts that is specified as a quota on the domain (the default value is 10). For more information, see the ms-DS-MachineAccountQuota attribute in the Active Directory schema.

The **NetCreateProvisioningPackage** function works only with a writable domain controller and does not function against a read-only domain controller. Once provisioning is done against a writable domain controller and the account is replicated to a read-only domain controller, the other portions of the offline domain join operation do not require access to a domain controller.

If the **NetCreateProvisioningPackage** function is successful, the pointer in the *pPackageBinData* or *pPackageTextData* parameter (depending on which parameter was not **NULL**) is returned with the serialized data for use in an offline join operation or as text in an unattended setup file.

All phases of the provisioning process append to a *NetSetup.log* file on the local computer. The provisoning process can include up to three different computers: the computer where the provisioning package is created, the computer that requests the installation of the package, and the computer where the package is installed. There will be *NetSetup.log* file information stored on all three computers according to the operation performed. Reviewing the contents of these files is the most common means of troubleshooting online and offline provisioning errors. Provisioning operations undertaken by admins are logged to the *NetSetup.log* file in the *%WINDIR%\Debug*. Provisioning operations performed by non-admins are logged to the *NetSetup.log* file in the *%USERPROFILE%\Debug* folder.

For more information on offline domain join operations, see the Offline Domain Join Step-by-Step Guide.

Joining (and unjoining) a computer to a domain using NetJoinDomain and NetUnjoinDomain is performed only by a member of the Administrators local group on the target computer. Note that the domain administrator can set additional requirements for joining the domain using delegation and assignment of privileges.

Requirements

Minimum supported client	Windows 8 [desktop apps only]
Minimum supported server	Windows Server 2012 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NETSETUP_PROVISIONING_PARAMS

NetJoinDomain

NetRenameMachineInDomain

NetRequestOfflineDomainJoin

NetRequestProvisioningPackageInstall

NetUnjoinDomain

Network Management Functions

Network Management Overview

NetEnumerateComputerNames function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetEnumerateComputerNames** function enumerates names for the specified computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetEnumerateComputerNames(
  LPCWSTR               Server,
  NET_COMPUTER_NAME_TYPE NameType,
  ULONG                 Reserved,
  PDWORD                EntryCount,
  LPWSTR                **ComputerNames
);
```

Parameters

`Server`

A pointer to a constant string that specifies the name of the computer on which to execute this function. If this parameter is **NULL**, the local computer is used.

`NameType`

The type of the name queried. This member can be one of the following values defined in the **NET_COMPUTER_NAME_TYPE** enumeration defined in the *Lmjoin.h* header file.

VALUE	MEANING
NetPrimaryComputerName	The primary computer name.
NetAlternateComputerNames	Alternate computer names.
NetAllComputerNames	All computer names.
NetComputerNameTypeMax	Indicates the end of the range that specifies the possible values for the type of name to be queried.

`Reserved`

Reserved for future use. This parameter should be **NULL**.

`EntryCount`

A pointer to a DWORD value that returns the number of names returned in the buffer pointed to by the *ComputerNames* parameter if the function succeeds.

`ComputerNames`

A pointer to an array of pointers to names. If the function call is successful, this parameter will return the computer names that match the computer type name specified in the *NameType* parameter.

When the application no longer needs this array, this buffer should be freed by calling NetApiBufferFree function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller was not a member of the Administrators local group on the target computer.
ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_NOT_ENOUGH_MEMORY	Not enough memory is available to process this command.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the target computer specified in the *Server* parameter on which this function executes is running on Windows 2000 and earlier.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetEnumerateComputerNames** function is supported on Windows Vista and later.

The **NetEnumerateComputerNames** function is used to request the names a computer currently has configured.

The **NetEnumerateComputerNames** function requires that the caller is a member of the Administrators local group on the target computer.

Requirements

Minimum supported client	Windows XP [desktop apps only]

Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetAddAlternateComputerName

NetJoinDomain

NetRemoveAlternateComputerName

NetRenameMachineInDomain

NetSetPrimaryComputerName

NetUnjoinDomain

SetComputerNameEx

NetFreeAadJoinInformation function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

Frees the memory allocated for the specified DSREG_JOIN_INFO structure, which contains join information for a tenant and which you retrieved by calling the NetGetAadJoinInformation function.

Syntax

```
VOID NET_API_FUNCTION NetFreeAadJoinInformation(
  PDSREG_JOIN_INFO pJoinInfo
);
```

Parameters

`pJoinInfo`

Pointer to the DSREG_JOIN_INFO structure for which you want to free the memory.

Return value

This function does not return a value.

Requirements

Minimum supported client	Windows 10 [desktop apps only]
Minimum supported server	Windows Server 2016 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h
Library	Netapi32.lib
DLL	Netapi32.dll

See also

DSREG_JOIN_INFO

NetGetAadJoinInformation

NetGetAadJoinInformation function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

Retrieves the join information for the specified tenant. This function examines the join information for Microsoft Azure Active Directory and the work account that the current user added.

Syntax

```
HRESULT NET_API_FUNCTION NetGetAadJoinInformation(
  LPCWSTR          pcszTenantId,
  PDSREG_JOIN_INFO *ppJoinInfo
);
```

Parameters

`pcszTenantId`

The tenant identifier for the joined account. If the device is not joined to Azure Active Directory (Azure AD), and the user currently logged into Windows added no Azure AD work accounts for the specified tenant, the buffer that the *ppJoinInfo* parameter points to is set to NULL.

If the specified tenant ID is NULL or empty, *ppJoinInfo* is set to the default join account information, or NULL if the device is not joined to Azure AD and the current user added no Azure AD work accounts.

```
                    The default join account is one of the following:
```

- The Azure AD account, if the device is joined to Azure AD.
- The Azure AD work account that the current user added, if the device is not joined to Azure AD, but the current user added a single Azure AD work account.
- Any of the Azure AD work accounts that the current user added, if the device is not joined to Azure AD, but the current user added multiple Azure AD work accounts. The algorithm for selecting one of the work accounts is not specified.

`ppJoinInfo`

The join information for the tenant that the *pcszTenantId* parameter specifies. If this parameter is NULL, the device is not joined to Azure AD and the current user added no Azure AD work accounts. You must call the NetFreeAadJoinInformation function to free the memory allocated for this structure.

Return value

If this function succeeds, it returns **S_OK**. Otherwise, it returns an **HRESULT** error code.

Requirements

Minimum supported client	Windows 10 [desktop apps only]

Minimum supported server	Windows Server 2016 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetFreeAadJoinInformation

NetGetJoinableOUs function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGetJoinableOUs** function retrieves a list of organizational units (OUs) in which a computer account can be created.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGetJoinableOUs(
  LPCWSTR lpServer,
  LPCWSTR lpDomain,
  LPCWSTR lpAccount,
  LPCWSTR lpPassword,
  DWORD   *OUCount,
  LPWSTR  **OUs
);
```

Parameters

`lpServer`

Pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which to call the function. If this parameter is **NULL**, the local computer is used.

`lpDomain`

Pointer to a constant string that specifies the name of the domain for which to retrieve the list of OUs that can be joined.

`lpAccount`

Pointer to a constant string that specifies the account name to use when connecting to the domain controller. The string must specify either a domain NetBIOS name and user account (for example, "REDMOND\user") or the user principal name (UPN) of the user in the form of an Internet-style login name (for example, "someone@example.com"). If this parameter is **NULL**, the caller's context is used.

`lpPassword`

If the *lpAccount* parameter specifies an account name, this parameter must point to the password to use when connecting to the domain controller. Otherwise, this parameter must be **NULL**.

`OUCount`

Receives the count of OUs returned in the list of joinable OUs.

`OUs`

Pointer to an array that receives the list of joinable OUs. This array is allocated by the system and must be freed using a single call to the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
NERR_DefaultJoinRequired	The destination domain controller does not support creating computer accounts in OUs.

Remarks

No special group membership is required to successfully execute the **NetGetJoinableOUs** function.

For more information about organizational units, see Managing Users in the Active Directory documentation.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetGetJoinInformation

Network Management Functions

Network Management Overview

NetGetJoinInformation function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetGetJoinInformation** function retrieves join status information for the specified computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetGetJoinInformation(
  LPCWSTR               lpServer,
  LPWSTR                *lpNameBuffer,
  PNETSETUP_JOIN_STATUS BufferType
);
```

Parameters

`lpServer`

Pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which to call the function. If this parameter is **NULL**, the local computer is used.

`lpNameBuffer`

Pointer to the buffer that receives the NetBIOS name of the domain or workgroup to which the computer is joined. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`BufferType`

Receives the join status of the specified computer. This parameter can have one of the following values.

```
typedef enum _NETSETUP_JOIN_STATUS {

    NetSetupUnknownStatus = 0,
    NetSetupUnjoined,
    NetSetupWorkgroupName,
    NetSetupDomainName

} NETSETUP_JOIN_STATUS, *PNETSETUP_JOIN_STATUS;
```

These values have the following meanings.

VALUE	MEANING
NetSetupUnknownStatus	The status is unknown.
NetSetupUnjoined	The computer is not joined.
NetSetupWorkgroupName	The computer is joined to a workgroup.

NetSetupDomainName	The computer is joined to a domain.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be the following error code or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.

Remarks

No special group membership is required to successfully execute the **NetGetJoinInformation** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll; Wkscli.dll

See also

NetGetJoinableOUs

Network Management Functions

Network Management Overview

NetJoinDomain function (lmjoin.h)

2/1/2021 • 8 minutes to read • Edit Online

The **NetJoinDomain** function joins a computer to a workgroup or domain.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetJoinDomain(
  LPCWSTR lpServer,
  LPCWSTR lpDomain,
  LPCWSTR lpMachineAccountOU,
  LPCWSTR lpAccount,
  LPCWSTR lpPassword,
  DWORD   fJoinOptions
);
```

Parameters

`lpServer`

A pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which to execute the domain join operation. If this parameter is **NULL**, the local computer is used.

`lpDomain`

A pointer to a constant null-terminated character string that specifies the name of the domain or workgroup to join.

Optionally, you can specify the preferred domain controller to perform the join operation. In this instance, the string must be of the form *DomainName**MachineName*, where *DomainName* is the name of the domain to join, and *MachineName* is the name of the domain controller to perform the join.

`lpMachineAccountOU`

Optionally specifies the pointer to a constant null-terminated character string that contains the RFC 1779 format name of the organizational unit (OU) for the computer account. If you specify this parameter, the string must contain a full path, for example, OU=testOU,DC=domain,DC=Domain,DC=com. Otherwise, this parameter must be **NULL**.

`lpAccount`

A pointer to a constant null-terminated character string that specifies the account name to use when connecting to the domain controller. The string must specify either a domain NetBIOS name and user account (for example, *REDMOND**user*) or the user principal name (UPN) of the user in the form of an Internet-style login name (for example, "someone@example.com"). If this parameter is **NULL**, the caller's context is used.

`lpPassword`

If the *lpAccount* parameter specifies an account name, this parameter must point to the password to use when connecting to the domain controller. Otherwise, this parameter must be **NULL**.

You can specify a local machine account password rather than a user password for unsecured joins. For more information, see the description of the NETSETUP_MACHINE_PWD_PASSED flag described in the *fJoinOptions* parameter.

`fJoinOptions`

A set of bit flags defining the join options. This parameter can be one or more of the following values defined in the *Lmjoin.h* header file.

VALUE	MEANING
NETSETUP_JOIN_DOMAIN 0x00000001	Joins the computer to a domain. If this value is not specified, joins the computer to a workgroup.
NETSETUP_ACCT_CREATE 0x00000002	Creates the account on the domain.
NETSETUP_WIN9X_UPGRADE 0x00000010	The join operation is occurring as part of an upgrade.
NETSETUP_DOMAIN_JOIN_IF_JOINED 0x00000020	Allows a join to a new domain even if the computer is already joined to a domain.
NETSETUP_JOIN_UNSECURE 0x00000040	Performs an unsecured join. This option requests a domain join to a pre-created account without authenticating with domain user credentials. This option can be used in conjunction with **NETSETUP_MACHINE_PWD_PASSED** option. In this case, *lpPassword* is the password of the pre-created machine account. Prior to Windows Vista with SP1 and Windows Server 2008, an unsecure join did not authenticate to the domain controller. All communication was performed using a null (unauthenticated) session. Starting with Windows Vista with SP1 and Windows Server 2008, the machine account name and password are used to authenticate to the domain controller.
NETSETUP_MACHINE_PWD_PASSED 0x00000080	Indicates that the *lpPassword* parameter specifies a local machine account password rather than a user password. This flag is valid only for unsecured joins, which you must indicate by also setting the NETSETUP_JOIN_UNSECURE flag. If you set this flag, then after the join operation succeeds, the machine password will be set to the value of *lpPassword*, if that value is a valid machine password.
NETSETUP_DEFER_SPN_SET 0x00000100	Indicates that the service principal name (SPN) and the DnsHostName properties on the computer object should not be updated at this time. Typically, these properties are updated during the join operation. Instead, these properties should be updated during a subsequent call to the NetRenameMachineInDomain function. These properties are always updated during the rename operation. For more information, see the following Remarks section.

NETSETUP_JOIN_DC_ACCOUNT 0x00000200	Allow the domain join if existing account is a domain controller. **Note** This flag is supported on Windows Vista and later.
NETSETUP_JOIN_WITH_NEW_NAME 0x00000400	Join the target machine specified in *lpServer* parameter with a new name queried from the registry on the machine specified in the *lpServer* parameter. This option is used if SetComputerNameEx has been called prior to rebooting the machine. The new computer name will not take effect until a reboot. With this option, the caller instructs the **NetJoinDomain** function to use the new name during the domain join operation. A reboot is required after calling **NetJoinDomain** successfully at which time both the computer name change and domain membership change will have taken affect. **Note** This flag is supported on Windows Vista and later.
NETSETUP_JOIN_READONLY 0x00000800	Join the target machine specified in *lpServer* parameter using a pre-created account without requiring a writable domain controller. This option provides the ability to join a machine to domain if an account has already been provisioned and replicated to a read-only domain controller. The target read-only domain controller is specified as part of the *lpDomain* parameter, after the domain name delimited by a '\' character. This provisioning must include the machine secret. The machine account must be added via group membership into the allowed list for password replication policy, and the account password must be replicated to the read-only domain controller prior to the join operation. For more information, see the information on Password Replication Policy Administration. Starting with Windows 7, an alternate mechanism is to use the offline domain join mechanism. For more information, see the **NetProvisionComputerAccount** and **NetRequestOfflineDomainJoin** functions. **Note** This flag is supported on Windows Vista and later.
NETSETUP_AMBIGUOUS_DC 0x00001000	When joining the domain don't try to set the preferred domain controller in the registry. **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.

NETSETUP_NO_NETLOGON_CACHE 0x00002000	When joining the domain don't create the Netlogon cache. **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.
NETSETUP_DONT_CONTROL_SERVICES 0x00004000	When joining the domain don't force Netlogon service to start. **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.
NETSETUP_SET_MACHINE_NAME 0x00008000	When joining the domain for offline join only, set target machine hostname and NetBIOS name. **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.
NETSETUP_FORCE_SPN_SET 0x00010000	When joining the domain, override other settings during domain join and set the service principal name (SPN). **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.
NETSETUP_NO_ACCT_REUSE 0x00020000	When joining the domain, do not reuse an existing account. **Note** This flag is supported on Windows 7, Windows Server 2008 R2, and later.
NETSETUP_IGNORE_UNSUPPORTED_FLAGS 0x10000000	If this bit is set, unrecognized flags will be ignored by the **NetJoinDomain** function and **NetJoinDomain** will behave as if the flags were not set.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller was not a member of the Administrators local group on the target computer.

ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *lpDomain* parameter is **NULL**.
ERROR_NO_SUCH_DOMAIN	The specified domain did not exist.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the computer specified in the *lpServer* parameter does not support some of the options passed in the *fJoinOptions* parameter.
NERR_InvalidWorkgroupName	The specified workgroup name is not valid.
NERR_SetupAlreadyJoined	The computer is already joined to a domain.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

Joining (and unjoining) a computer to a domain or workgroup can be performed only by a member of the Administrators local group on the target computer. Note that the domain administrator can set additional requirements for joining the domain using delegation and assignment of privileges.

If you call the **NetJoinDomain** function remotely, you must supply credentials because you cannot delegate credentials under these circumstances.

Different processes, or different threads of the same process, should not call the **NetJoinDomain** function at the same time. This situation can leave the computer in an inconsistent state.

If you encounter a problem during a join operation, you should not delete a computer account and immediately follow the deletion with another join attempt. This can lead to replication-related problems that are difficult to investigate. When you delete a computer account, wait until the change has replicated to all domain controllers before attempting another join operation.

A system reboot is required after calling the **NetJoinDomain** function for the operation to complete.

Windows Server 2003 and Windows XP: When a call to the **NetJoinDomain** function precedes a call to the NetRenameMachineInDomain function, you should defer the update of the SPN and DnsHostName properties on the computer object until the rename operation. This is because the join operation can fail in certain situations. An example of such a situation is when the SPN that is derived from the current computer name is not valid in the new domain that the computer is joining, but the SPN derived from the new name that the computer will have after the rename operation is valid in the new domain. In this situation, the call to **NetJoinDomain** fails unless you defer the update of the two properties until the rename operation by specifying the NETSETUP_DEFER_SPN_SET flag in the *fJoinOptions* parameter when you call **NetJoinDomain**.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetAddAlternateComputerName

NetCreateProvisioningPackage

NetEnumerateComputerNames

NetProvisionComputerAccount

NetRemoveAlternateComputerName

NetRenameMachineInDomain

NetRequestOfflineDomainJoin

NetRequestProvisioningPackageInstall

NetSetPrimaryComputerName

NetUnjoinDomain

Network Management Functions

Network Management Overview

Offline Domain Join Step-by-Step Guide

Password Replication Policy Administration

NetProvisionComputerAccount function (lmjoin.h)

2/1/2021 • 8 minutes to read • Edit Online

The **NetProvisionComputerAccount** function provisions a computer account for later use in an offline domain join operation.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetProvisionComputerAccount(
  LPCWSTR lpDomain,
  LPCWSTR lpMachineName,
  LPCWSTR lpMachineAccountOU,
  LPCWSTR lpDcName,
  DWORD   dwOptions,
  PBYTE   *pProvisionBinData,
  DWORD   *pdwProvisionBinDataSize,
  LPWSTR  *pProvisionTextData
);
```

Parameters

`lpDomain`

A pointer to a **NULL**-terminated character string that specifies the name of the domain where the computer account is created.

`lpMachineName`

A pointer to a **NULL**-terminated character string that specifies the short name of the machine from which the computer account attribute sAMAccountName is derived by appending a '$'. This parameter must contain a valid DNS or NetBIOS machine name.

`lpMachineAccountOU`

An optional pointer to a **NULL**-terminated character string that contains the RFC 1779 format name of the organizational unit (OU) where the computer account will be created. If you specify this parameter, the string must contain a full path, for example, OU=testOU,DC=domain,DC=Domain,DC=com. Otherwise, this parameter must be **NULL**.

If this parameter is **NULL**, the well known computer object container will be used as published in the domain.

`lpDcName`

An optional pointer to a **NULL**-terminated character string that contains the name of the domain controller to target.

`dwOptions`

A set of bit flags that define provisioning options. This parameter can be one or more of the following values defined in the *Lmjoin.h* header file.

VALUE	MEANING

NETSETUP_PROVISION_DOWNLEVEL_PRIV_SUPPORT 0x00000001	If the caller requires account creation by privilege, this option will cause a retry on failure using account creation functions enabling interoperability with domain controllers running on earlier versions of Windows. The *lpMachineAccountOU* is not supported when using downlevel privilege support.
NETSETUP_PROVISION_REUSE_ACCOUNT 0x00000002	If the named account already exists, an attempt will be made to reuse the existing account. This option requires sufficient credentials for this operation (Domain Administrator or the object owner).
NETSETUP_PROVISION_USE_DEFAULT_PASSWORD 0x00000004	Use the default machine account password which is the machine name in lowercase. This is largely to support the older unsecure join model where the pre-created account typically used this default password. **Note** Applications should avoid using this option if possible. This option as well as NetJoinDomain function with dwOptions set to NETSETUP_JOIN_UNSECURE for unsecure join should only be used on earlier versions of Windows.
NETSETUP_PROVISION_SKIP_ACCOUNT_SEARCH 0x00000008	Do not try to find the account on any domain controller in the domain. This option makes the operation faster, but should only be used when the caller is certain that an account by the same name hasn't recently been created. This option is only valid when the *lpDcName* parameter is specified. When the prerequisites are met, this option allows for must faster provisioning useful for scenarios such as batch processing.
NETSETUP_PROVISION_ROOT_CA_CERTS 0x00000010	This option retrieves all of the root Certificate Authority certificates on the local machine and adds them to the provisioning package when no certificate template names are provided as part of the provisioning package (the **aCertTemplateNames** member of the NETSETUP_PROVISIONING_PARAMS struct passed in the *pProvisioningParams* parameter to the NetCreateProvisioningPackage function is NULL). **Note** This flag is only supported by the NetCreateProvisioningPackage function on Windows 8, Windows Server 2012, and later.

`pProvisionBinData`

An optional pointer that will receive the opaque binary blob of serialized metadata required by NetRequestOfflineDomainJoin function to complete an offline domain join, if the **NetProvisionComputerAccount** function completes successfully. The data is returned as an opaque binary buffer which may be passed to **NetRequestOfflineDomainJoin** function.

If this parameter is **NULL**, then *pProvisionTextData* parameter must not be **NULL**. If this parameter is not **NULL**,

then the *pProvisionTextData* parameter must be **NULL**.

`pdwProvisionBinDataSize`

A pointer to a value that receives the size, in bytes, of the buffer returned in the *pProvisionBinData* parameter.

This parameter must not be **NULL** if the *pProvisionBinData* parameter is not **NULL**. This parameter must be **NULL** when the *pProvisionBinData* parameter is **NULL**.

`pProvisionTextData`

An optional pointer that will receive the opaque binary blob of serialized metadata required by NetRequestOfflineDomainJoin function to complete an offline domain join, if the **NetProvisionComputerAccount** function completes successfully. The data is returned in string form for embedding in an unattended setup answer file.

If this parameter is **NULL**, then the *pProvisionBinData* parameter must not be **NULL**. If this parameter is not **NULL**, then the the *pProvisionBinData* parameter must be **NULL**.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller does not have sufficient privileges to complete the operation.
ERROR_INVALID_DOMAIN_ROLE	This operation is only allowed for the Primary Domain Controller of the domain. This error is returned if a domain controller name was specified in the *lpDcName* parameter, but the computer specified could not be validated as a domain controller for the target domain specified in the *lpDomain* parameter.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *lpDomain* or *lpMachineName* parameter is **NULL**. This error is also returned if both the *pProvisionBinData* and *pProvisionTextData* parameters are **NULL**.
ERROR_NO_SUCH_DOMAIN	The specified domain did not exist.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the *lpMachineAccountOU* parameter was specified and the domain controller is running on an earlier versions of Windows that does not support this parameter.
NERR_DS8DCRequired	The specified domain controller does not meet the version requirement for this operation.
NERR_LDAPCapableDCRequired	This operation requires a domain controller which supports LDAP.

NERR_UserExists	The account already exists in the domain and the NETSETUP_PROVISION_REUSE_ACCOUNT bit was not specified in the *dwOptions* parameter.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetProvisionComputerAccount** function is supported on Windows 7 and Windows Server 2008 R2 for offline join operations. On Windows 8 or Windows Server 2008 R2, it is recommended that the NetCreateProvisioningPackage function be used instead of the **NetProvisionComputerAccount** function.

The **NetProvisionComputerAccount** function is used to provision a computer account for later use in an offline domain join operation using the NetRequestOfflineDomainJoin function. The offline domain join scenario uses these functions as follows:

- **NetProvisionComputerAccount** is a provisioning function that is first called to perform the network operations necessary to create and configure the computer object in Active Directory. The output from the **NetProvisionComputerAccount** is an opaque binary blob of serialized metadata used for the next step.
- NetRequestOfflineDomainJoin, an image initialization function, is then called to inject the output from the **NetProvisionComputerAccount** provisioning function into a Windows operating system image to be used during installation.

Changes to Windows initialization code will detect this saved state and affect the local only portion of domain join.

The **NetProvisionComputerAccount** function will create or reuse the machine account in the domain, collect all necessary metadata and return it in an opaque versioned binary blob or as text for embedding in an unattended setup answer file. The opaque binary blob can be consumed by the offline domain join request operation supplying all the necessary input to complete the domain join during first boot without any network operations (local state updates only).

Security Note: The blob returned by the **NetProvisionComputerAccount** function contains very sensitive data. It should be treated just as securely as a plaintext password. The blob contains the machine account password and other information about the domain, including the domain name, the name of a domain controller, and the security ID (SID) of the domain. If the blob is being transported physically or over the network, care must be taken to transport it securely. The design makes no provisions for securing this data. This problem exists today with unattended setup answer files which can carry a number of secrets including domain user passwords. The caller must secure the blob and the unattended setup files. Solutions to this problem are varied. As an example, a pre-exchanged key could be used to encrypt a session between the consumer and provisioning entity enabling a secure transfer of the opaque blob.

The opaque blob returned in the *pProvisionBinData* parameter by the **NetProvisionComputerAccount** function is versioned to allow interoperability and serviceability scenarios between different versions of Windows (joining client, provisioning machine, and domain controller). The offline join scenario currently does not limit the lifetime of the blob returned by the **NetProvisionComputerAccount** function.

For offline domain joins, the access check performed depends on the configuration of the domain. Computer account creation is enabled using three methods:

- Domain administrators have rights to create computer accounts.
- The SD on a container can delegate the rights to create computer accounts.
- By default, authenticated users may create computer accounts by privilege. Authenticated users are limited to creating a limited number of accounts that is specified as a quota on the domain (the default value is 10). For more information, see the ms-DS-MachineAccountQuota attribute in the Active Directory schema.

The **NetProvisionComputerAccount** function works only with a writable domain controller and does not function against a read-only domain controller. Once provisioning is done against a writable domain controller and the account is replicated to a read-only domain controller, then the other portions of offline domain join operation do not require access to a domain controller.

If the **NetProvisionComputerAccount** function is successful, the pointer in the *pProvisionBinData* or *pProvisionTextData* parameter (depending on which was parameter was not **NULL**) is returned with the serialized data for use in an offline join operation or as text in an unattended setup file.

For more information on offline domain join operations, see the Offline Domain Join Step-by-Step Guide.

Joining (and unjoining) a computer to a domain using NetJoinDomain and NetUnjoinDomain can be performed only by a member of the Administrators local group on the target computer. Note that the domain administrator can set additional requirements for joining the domain using delegation and assignment of privileges.

Requirements

Minimum supported client	Windows 7 [desktop apps only]
Minimum supported server	Windows Server 2008 R2 [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetCreateProvisioningPackage

NetJoinDomain

NetRenameMachineInDomain

NetRequestOfflineDomainJoin

NetUnjoinDomain

Network Management Functions

Network Management Overview

Offline Domain Join Step-by-Step Guide

ms-DS-MachineAccountQuota

NetRemoveAlternateComputerName function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetRemoveAlternateComputerName** function removes an alternate name for the specified computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRemoveAlternateComputerName(
  LPCWSTR Server,
  LPCWSTR AlternateName,
  LPCWSTR DomainAccount,
  LPCWSTR DomainAccountPassword,
  ULONG   Reserved
);
```

Parameters

`Server`

A pointer to a constant string that specifies the name of the computer on which to execute this function. If this parameter is **NULL**, the local computer is used.

`AlternateName`

A pointer to a constant string that specifies the alternate name to remove. This name must be in the form of a fully qualified DNS name.

`DomainAccount`

A pointer to a constant string that specifies the domain account to use for accessing the machine account object for the computer specified in the *Server* parameter in Active Directory. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is not used if the server to execute this function is not joined to a domain.

`DomainAccountPassword`

A pointer to a constant string that specifies the password matching the domain account passed in the *DomainAccount* parameter. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is ignored if the *DomainAccount* parameter is **NULL**. This parameter is not used if the server to execute this function is not joined to a domain.

`Reserved`

Reserved for future use. This parameter should be **NULL**.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller was not a member of the Administrators local group on the target computer.
ERROR_INVALID_NAME	A name parameter is incorrect. This error is returned if the *AlternateName* parameter does not contain valid name.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *DomainAccount* parameter does not contain a valid domain. This error is also returned if the *DomainAccount* parameter is not **NULL** and the *DomainAccountPassword* parameter is not **NULL** but does not contain a Unicode string.
ERROR_NOT_ENOUGH_MEMORY	Not enough memory is available to process this command.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the target computer specified in the *Server* parameter on which this function executes is running on Windows 2000 and earlier.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetRemoveAlternateComputerName** function is supported on Windows XP and later.

The **NetRemoveAlternateComputerName** function is used to remove secondary computer names configured for the target computer.

The **NetRemoveAlternateComputerName** function requires that the caller is a member of the Administrators local group on the target computer.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows

Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetAddAlternateComputerName

NetEnumerateComputerNames

NetJoinDomain

NetRenameMachineInDomain

NetSetPrimaryComputerName

NetUnjoinDomain

SetComputerNameEx

NetRenameMachineInDomain function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetRenameMachineInDomain** function changes the name of a computer in a domain.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRenameMachineInDomain(
  LPCWSTR lpServer,
  LPCWSTR lpNewMachineName,
  LPCWSTR lpAccount,
  LPCWSTR lpPassword,
  DWORD   fRenameOptions
);
```

Parameters

`lpServer`

A pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which to call the function. If this parameter is **NULL**, the local computer is used.

`lpNewMachineName`

A pointer to a constant string that specifies the new name of the computer. If specified, the local computer name is changed as well. If this parameter is **NULL**, the function assumes you have already called the SetComputerNameEx function.

`lpAccount`

A pointer to a constant string that specifies an account name to use when connecting to the domain controller. If this parameter is **NULL**, the caller's context is used.

`lpPassword`

If the *lpAccount* parameter specifies an account name, this parameter must point to the password to use when connecting to the domain controller. Otherwise, this parameter must be **NULL**.

`fRenameOptions`

The rename options. If this parameter is NETSETUP_ACCT_CREATE, the function renames the account in the domain.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the account name passed in the *lpAccount* parameter did not have sufficient access rights for the operation.

ERROR_INVALID_PARAMETER	A parameter is incorrect.
NERR_SetupNotJoined	The computer is not currently joined to a domain.
NERR_SetupDomainController	This computer is a domain controller and cannot be unjoined from a domain.

Remarks

Renaming a domain computer can be performed only by a user that is a member of the Administrators local group on the target computer and that also is a member of the Administrators group on the domain or has the Account Operator privilege on the domain. If you call the **NetRenameMachineInDomain** function remotely, you must supply credentials because you cannot delegate credentials under these circumstances.

Different processes, or different threads of the same process, should not call the **NetRenameMachineInDomain** function at the same time. This situation can leave the computer in an inconsistent state.

The **NERR_SetupNotJoined** and **NERR_SetupDomainController** return values are defined in the Lmerr.h header file. This header file is automatically included by the Lm.h header file and should not be included directly.

A system reboot is required after calling the **NetRenameMachineInDomain** function for the operation to complete.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetAddAlternateComputerName

NetEnumerateComputerNames

NetRemoveAlternateComputerName

NetSetPrimaryComputerName

NetUnjoinDomain

Network Management Functions

Network Management Overview

SetComputerNameEx

NetRequestOfflineDomainJoin function (lmjoin.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetRequestOfflineDomainJoin** function executes locally on a machine to modify a Windows operating system image mounted on a volume. The registry is loaded from the image and provisioning blob data is written where it can be retrieved during the completion phase of an offline domain join operation.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRequestOfflineDomainJoin(
  BYTE    *pProvisionBinData,
  DWORD   cbProvisionBinDataSize,
  DWORD   dwOptions,
  LPCWSTR lpWindowsPath
);
```

Parameters

`pProvisionBinData`

A pointer to a buffer required to initialize the registry of a Windows operating system image to process the final local state change during the completion phase of the offline domain join operation.

The opaque binary blob of serialized metadata passed in the *pProvisionBinData* parameter is returned by the NetProvisionComputerAccount function.

`cbProvisionBinDataSize`

The size, in bytes, of the buffer pointed to by the *pProvisionBinData* parameter.

This parameter must not be **NULL**.

`dwOptions`

A set of bit flags that define options for this function. This parameter can be one or more of the following values defined in the *Lmjoin.h* header file.

VALUE	MEANING
NETSETUP_PROVISION_ONLINE_CALLER 0x40000000	This flag is required if the *lpWindowsPath* parameter references the currently running Windows operating system directory rather than an offline Windows operating system image mounted on an accessible volume. If this flag is specified, the **NetRequestOfflineDomainJoin** function must be invoked by a member of the local Administrators group.

`lpWindowsPath`

A pointer to a constant null-terminated character string that specifies the path to a Windows operating system image under which the registry hives are located. This image must be offline and not currently booted unless the *dwOptions* parameter contains **NETSETUP_PROVISION_ONLINE_CALLER** in which case the locally running operating system directory is allowed.

This path could be a UNC path on a remote server.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller does not have sufficient privileges to complete the operation.
ERROR_ELEVATION_REQUIRED	The requested operation requires elevation.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *pProvisionBinData*, *cbProvisionBinDataSize*, or *lpWindowsPath* parameters are **NULL**. This error is also returned if the buffer pointed to by the *pProvisionBinData* parameter does not contain valid data in the blob for the domain, machine account name, or machine account password. This error is also returned if the string pointed to *lpWindowsPath* parameter does not specific the path to a Windows operating system image.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the specified server does not support this operation. For example, if the *lpWindowsPath* parameter references a Windows installation configured as a domain controller.
NERR_WkstaNotStarted	The Workstation service has not been started.

Remarks

The **NetRequestOfflineDomainJoin** function is supported on Windows 7 for offline domain join operations.

The **NetRequestOfflineDomainJoin** function is used locally on a machine to modify a Windows operating system image mounted on a volume. The registry is loaded for the image and provisioning blob data is written where it can be retrieved during the completion phase of an offline domain join operation. The offline domain join scenario uses these functions as follows:

- NetProvisionComputerAccount is a provisioning function that is first called to perform the network operations necessary to create and configure the computer object in Active Directory. The output from the **NetProvisionComputerAccount** is an opaque binary blob of serialized metadata used for the next step.
- **NetRequestOfflineDomainJoin** , an image initialization function, is then called to inject the output from the NetProvisionComputerAccount provisioning function into a Windows operating system image to be used during installation. Changes to Windows initialization code will detect this saved state and affect the local only portion of domain join.

The NetProvisionComputerAccount function will create or reuse the machine account in the domain, collect all necessary metadata and return it in an opaque versioned binary blob or as text for embedding in an unattended setup answer file. The opaque binary blob can be consumed by the offline domain join request operation supplying all the necessary input to complete the domain join during first boot without any network operations (local state updates only). Note that the blob contains machine account password material essentially in the

clear. The design makes no provisions for securing this data. This problem exists today with unattended setup answer files which can carry a number of secrets including domain user passwords. The caller must secure the blob and the unattended setup files. Solutions to this problem are varied. As an example, a pre-exchanged key could be used to encrypt a session between the consumer and provisioning entity enabling a secure transfer of the opaque blob .

The opaque blob returned in the *pProvisionBinData* parameter by the NetProvisionComputerAccount function is versioned to allow interoperability and serviceability scenarios between different versions of Windows (joining client, provisioning machine, and domain controller). The offline join scenario currently does not limit the lifetime of the blob returned by the **NetProvisionComputerAccount** function.

For more information on offline domain join operations, see the Offline Domain Join Step-by-Step Guide.

Requirements

Minimum supported client	Windows 7 [desktop apps only]
Minimum supported server	None supported
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetCreateProvisioningPackage

NetJoinDomain

NetProvisionComputerAccount

NetRenameMachineInDomain

NetRequestProvisioningPackageInstall

NetUnjoinDomain

Network Management Functions

Network Management Overview

Offline Domain Join Step-by-Step Guide

NetRequestProvisioningPackageInstall function (lmjoin.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetRequestProvisioningPackageInstall** function executes locally on a machine to modify a Windows operating system image mounted on a volume. The registry is loaded from the image and provisioning package data is written where it can be retrieved during the completion phase of an offline domain join operation.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRequestProvisioningPackageInstall(
  BYTE    *pPackageBinData,
  DWORD   dwPackageBinDataSize,
  DWORD   dwProvisionOptions,
  LPCWSTR lpWindowsPath,
  PVOID   pvReserved
);
```

Parameters

`pPackageBinData`

A pointer to a buffer required to initialize the registry of a Windows operating system image to process the final local state change during the completion phase of the offline domain join operation.

The opaque binary blob of serialized metadata passed in the *pPackageBinData* parameter is returned by the NetCreateProvisioningPackage function.

`dwPackageBinDataSize`

The size, in bytes, of the buffer pointed to by the *pPackageBinData* parameter.

This parameter must not be **NULL**.

`dwProvisionOptions`

A set of bit flags that define options for this function. This parameter uses one or more of the following values defined in the *Lmjoin.h* header file.

VALUE	MEANING
NETSETUP_PROVISION_ONLINE_CALLER 0x40000000	This flag is required if the *lpWindowsPath* parameter references the currently running Windows operating system directory rather than an offline Windows operating system image mounted on an accessible volume. If this flag is specified, the **NetRequestProvisioningPackageInstall** function must be invoked by a member of the local Administrators group.

`lpWindowsPath`

A pointer to a **NULL**-terminated character string that specifies the path to a Windows operating system image under which the registry hives are located. This image must be offline and not currently booted unless the

dwProvisionOptions parameter contains **NETSETUP_PROVISION_ONLINE_CALLER**, in which case, the locally running operating system directory is allowed.

This path could be a UNC path on a remote server.

`pvReserved`

Reserved for future use.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following Network Management error codes.

RETURN CODE	DESCRIPTION
NERR_NoOfflineJoinInfo	The offline join completion information was not found.
NERR_BadOfflineJoinInfo	The offline join completion information was bad.
NERR_CantCreateJoinInfo	Unable to create offline join information. Please ensure you have access to the specified path location and permissions to modify its contents. Running as an elevated administrator may be required.
NERR_BadDomainJoinInfo	The domain join info being saved was incomplete or bad.
NERR_JoinPerformedMustRestart	Offline join operation successfully completed but a restart is needed.
NERR_NoJoinPending	There was no offline join operation pending.
NERR_ValuesNotSet	Unable to set one or more requested machine or domain name values on the local computer.
NERR_CantVerifyHostname	Could not verify the current machine's hostname against the saved value in the join completion information.
NERR_CantLoadOfflineHive	Unable to load the specified offline registry hive. Please ensure you have access to the specified path location and permissions to modify its contents. Running as an elevated administrator may be required.
NERR_ConnectionInsecure	The minimum session security requirements for this operation were not met.
NERR_ProvisioningBlobUnsupported	Computer account provisioning blob version is not supported.

Remarks

The NetRequestProvisioningPackageInstall function is supported on Windows 8 for offline domain join operations. For Windows 7, use **NetRequestOfflineDomainJoin**.

The offline domain join scenario uses two functions:

- NetCreateProvisioningPackage is a provisioning function that is first called to perform the network operations necessary to create and configure the computer object in Active Directory. The output from the **NetCreateProvisioningPackage** is a package used for the next step.
- **NetRequestProvisioningPackageInstall**, an image initialization function, is called to inject the output from the NetCreateProvisioningPackage provisioning function into a Windows operating system image for use during installation.

Changes to Windows initialization code will detect this saved state and affect the local-only portion of domain join and install any certificate and policy information that may have been present in the package.

The NetCreateProvisioningPackage function will create or reuse the machine account in the domain, collect all necessary metadata and return it in a package. The package can be consumed by the offline domain join request operation supplying all the necessary input to complete the domain join during first boot without any network operations (local state updates only).

Security Note: The package created by the NetCreateProvisioningPackage function contains very sensitive data. It should be treated just as securely as a plaintext password. The package contains the machine account password and other information about the domain, including the domain name, the name of a domain controller, and the security ID (SID) of the domain. If the package is being transported physically or over the network, care must be taken to transport it securely. The design makes no provisions for securing this data. This problem exists today with unattended setup answer files which can carry a number of secrets including domain user passwords. The caller must secure the package. Solutions to this problem are varied. As an example, a pre-exchanged key could be used to encrypt a session between the consumer and provisioning entity enabling a secure transfer of the package.

The package returned in the *pPackageBinData* parameter by the NetCreateProvisioningPackage function is versioned to allow interoperability and serviceability scenarios between different versions of Windows (such as joining a client, provisioning a machine, and using a domain controller). The offline join scenario currently does not limit the lifetime of the package returned by the **NetCreateProvisioningPackage** function.

All phases of the provisioning process append to a *NetSetup.log* file on the local computer. The provisoning process can include up to three different computers: the computer where the provisioning package is created, the computer that requests the installation of the package, and the computer where the package is installed. There will be *NetSetup.log* file information stored on all three computers according to the operation performed. Reviewing the contents of these files is the most common means of troubleshooting online and offline provisioning errors. Provisioning operations undertaken by admins are logged to the *NetSetup.log* file in the *%WINDIR%\Debug*. Provisioning operations performed by non-admins are logged to the *NetSetup.log* file in the *%USERPROFILE%\Debug* folder.

Requirements

Minimum supported client	Windows 8 [desktop apps only]
Minimum supported server	Windows Server 2012 [desktop apps only]
Target Platform	Windows

Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NETSETUP_PROVISIONING_PARAMS

NetCreateProvisioningPackage

NetJoinDomain

NetProvisionComputerAccount

NetRenameMachineInDomain

NetRequestOfflineDomainJoin

NetUnjoinDomain

Network Management Functions

Network Management Overview

NetSetPrimaryComputerName function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetSetPrimaryComputerName** function sets the primary computer name for the specified computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetSetPrimaryComputerName(
  LPCWSTR Server,
  LPCWSTR PrimaryName,
  LPCWSTR DomainAccount,
  LPCWSTR DomainAccountPassword,
  ULONG   Reserved
);
```

Parameters

`Server`

A pointer to a constant string that specifies the name of the computer on which to execute this function. If this parameter is **NULL**, the local computer is used.

`PrimaryName`

A pointer to a constant string that specifies the primary name to set. This name must be in the form of a fully qualified DNS name.

`DomainAccount`

A pointer to a constant string that specifies the domain account to use for accessing the machine account object for the computer specified in the *Server* parameter in Active Directory. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is not used if the server to execute this function is not joined to a domain.

`DomainAccountPassword`

A pointer to a constant string that specifies the password matching the domain account passed in the *DomainAccount* parameter. If this parameter is **NULL**, then the credentials of the user executing this routine are used.

This parameter is ignored if the *DomainAccount* parameter is **NULL**. This parameter is not used if the server to execute this function is not joined to a domain.

`Reserved`

Reserved for future use. This parameter should be **NULL**.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	Access is denied. This error is returned if the caller was not a member of the Administrators local group on the target computer.
ERROR_INVALID_NAME	A name parameter is incorrect. This error is returned if the *PrimaryName* parameter does not contain valid name.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *DomainAccount* parameter does not contain a valid domain. This error is also returned if the *DomainAccount* parameter is not **NULL** and the *DomainAccountPassword* parameter is not **NULL** but does not contain a Unicode string.
ERROR_NOT_ENOUGH_MEMORY	Not enough memory is available to process this command.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the target computer specified in the *Server* parameter on which this function executes is running on Windows 2000 and earlier.
NERR_WkstaNotStarted	The Workstation service has not been started.
RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
RPC_S_PROTSEQ_NOT_SUPPORTED	The remote procedure call protocol sequence is not supported.

Remarks

The **NetSetPrimaryComputerName** function is supported on Windows XP and later.

The **NetSetPrimaryComputerName** function is used as part of computer rename operations. The specified name will be removed from the alternate name list configured for the target computer and configured as the primary name. The computer account name will be changed to match the primary name. The previous primary computer name is moved to the alternate computer name list configured for the computer.

The **NetSetPrimaryComputerName** function requires that the caller is a member of the Administrators local group on the target computer.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Target Platform	Windows

Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetAddAlternateComputerName

NetEnumerateComputerNames

NetJoinDomain

NetRemoveAlternateComputerName

NetRenameMachineInDomain

NetUnjoinDomain

SetComputerNameEx

NETSETUP_PROVISIONING_PARAMS structure (lmjoin.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NETSETUP_PROVISIONING_PARAMS** structure contains information that is used when creating a provisioning package using the NetCreateProvisionPackage function.

Syntax

```
typedef struct _NETSETUP_PROVISIONING_PARAMS {
  DWORD   dwVersion;
  LPCWSTR lpDomain;
  LPCWSTR lpHostName;
  LPCWSTR lpMachineAccountOU;
  LPCWSTR lpDcName;
  DWORD   dwProvisionOptions;
  LPCWSTR *aCertTemplateNames;
  DWORD   cCertTemplateNames;
  LPCWSTR *aMachinePolicyNames;
  DWORD   cMachinePolicyNames;
  LPCWSTR *aMachinePolicyPaths;
  DWORD   cMachinePolicyPaths;
  LPWSTR  lpNetbiosName;
  LPWSTR  lpSiteName;
  LPWSTR  lpPrimaryDNSDomain;
} NETSETUP_PROVISIONING_PARAMS, *PNETSETUP_PROVISIONING_PARAMS;
```

Members

`dwVersion`

The version of Windows in the provisioning package. This parameter should use the following value defined in the *Lmjoin.h* header file.

VALUE	MEANING
NETSETUP_PROVISIONING_PARAMS_CURRENT_VERSION 0x00000001	The version for this package is Windows Server 2012.

`lpDomain`

A pointer to a **NULL**-terminated character string that specifies the name of the domain where the computer account is created.

`lpHostName`

A pointer to a **NULL**-terminated character string that specifies the short name of the machine from which the computer account attribute sAMAccountName is derived by appending a '$'. This parameter must contain a valid DNS or NetBIOS machine name.

`lpMachineAccountOU`

A optional pointer to a **NULL**-terminated character string that contains the RFC 1779 format name of the organizational unit (OU) where the computer account will be created. If you specify this parameter, the string must contain a full path, for example, OU=testOU,DC=domain,DC=Domain,DC=com. Otherwise, this parameter must be **NULL**.

If this parameter is **NULL**, the well known computer object container will be used as published in the domain.

`lpDcName`

An optional pointer to a **NULL**-terminated character string that contains the name of the domain controller to target.

`dwProvisionOptions`

A set of bit flags that define provisioning options. This parameter can be one or more of the following values defined in the *Lmjoin.h* header file.

VALUE	MEANING
NETSETUP_PROVISION_DOWNLEVEL_PRIV_SUPPORT 0x00000001	If the caller requires account creation by privilege, this option will cause a retry on failure using account creation functions enabling interoperability with domain controllers running on earlier versions of Windows. The *lpMachineAccountOU* is not supported when using downlevel privilege support.
NETSETUP_PROVISION_REUSE_ACCOUNT 0x00000002	If the named account already exists, an attempt will be made to reuse the existing account. This option requires sufficient credentials for this operation (Domain Administrator or the object owner).
NETSETUP_PROVISION_USE_DEFAULT_PASSWORD 0x00000004	Use the default machine account password which is the machine name in lowercase. This is largely to support the older unsecure join model where the pre-created account typically used this default password.
NETSETUP_PROVISION_SKIP_ACCOUNT_SEARCH 0x00000008	Do not try to find the account on any domain controller in the domain. This option makes the operation faster, but should only be used when the caller is certain that an account by the same name hasn't recently been created. This option is only valid when the *lpDcName* parameter is specified. When the prerequisites are met, this option allows for must faster provisioning useful for scenarios such as batch processing.
NETSETUP_PROVISION_ROOT_CA_CERTS 0x00000010	This option retrieves all of the root Certificate Authority certificates on the local machine and adds them to the provisioning package. **Note** This flag is only supported by the NetCreateProvisioningPackage function on Windows 8, Windows Server 2012, and later.

`aCertTemplateNames`

A pointer to an array of **NULL**-terminated certificate template names.

`cCertTemplateNames`

When **aCertTemplateNames** is not **NULL**, this member provides an explicit count of the number of items in the array.

`aMachinePolicyNames`

A pointer to an array of **NULL**-terminated machine policy names.

`cMachinePolicyNames`

When **aMachinePolicyNames** is not **NULL**, this member provides an explicit count of the number of items in the array.

`aMachinePolicyPaths`

A pointer to an array of character strings. Each array element is a NULL-terminated character string which specifies the full or partial path to a file in the Registry Policy File format. For more information on the Registry Policy File Format , see Registry Policy File Format

This path could be a UNC path on a remote server.

`cMachinePolicyPaths`

When **aMachinePolicyPaths** is not **NULL**, this member provides an explicit count of the number of items in the array.

`lpNetbiosName`

TBD

`lpSiteName`

TBD

`lpPrimaryDNSDomain`

TBD

Remarks

The **NETSETUP_PROVISIONING_PARAMS** structure provides flags for the NetCreateProvisioningPackage function which is supported on Windows 8 and Windows Server 2012 for offline join operations.

In addition to domain joins, the provisioning package can provide certificates and policies to the machine. The provisioning package can be used in four ways:

- Domain join
- Domain join and installation of certificates
- Domain join and installation of policies
- Domain join and installation of certificates and policies

When certificates need to be added to the package, this structure provides the **aCertTemplateNames** member as an array of **NULL**-terminated certificate template names. The **aCertTemplateNames** member requires the **cCertTemplateNames** member to provide an explicit count of the number of items in the array.

There are two different ways to add policies. You can use one or both methods:

- Policy name—An array of **NULL**-terminated policy names is provided in the **aMachinePolicyNames** member. During runtime, the policy name is mapped to the policy name in AD and the GUID that represents the policy in the enterprise space is retrieved. The **aMachinePolicyNames** member requires the

cMachinePolicyNames member to provide an explicit count of the number of items in the array.

- Policy path—A pointer to an array of **NULL**-terminated character strings provided in the **aMachinePolicyPaths** member which specify the path to a file in the Registry Policy File format. For more information on the Registry Policy File Format , see Registry Policy File Format. The policy path is a full or relative path to the policy file.

Requirements

Minimum supported client	Windows 8 [desktop apps only]
Minimum supported server	Windows Server 2012 [desktop apps only]
Header	lmjoin.h (include Lm.h)

See also

NetCreateProvisionPackage

NetJoinDomain

NetProvisionComputerAccount

NetRenameMachineInDomain

NetRequestOfflineDomainJoin

NetRequestProvisioningPackageInstall

NetUnjoinDomain

Network Management Functions

Network Management Overview

NetUnjoinDomain function (lmjoin.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetUnjoinDomain** function unjoins a computer from a workgroup or a domain.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUnjoinDomain(
  LPCWSTR lpServer,
  LPCWSTR lpAccount,
  LPCWSTR lpPassword,
  DWORD   fUnjoinOptions
);
```

Parameters

`lpServer`

A pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`lpAccount`

A pointer to a constant string that specifies the account name to use when connecting to the domain controller. The string must specify either a domain NetBIOS name and user account (for example, *REDMOND\user*) or the user principal name (UPN) of the user in the form of an Internet-style login name (for example, "someone@example.com"). If this parameter is **NULL**, the caller's context is used.

`lpPassword`

If the *lpAccount* parameter specifies an account name, this parameter must point to the password to use when connecting to the domain controller. Otherwise, this parameter must be **NULL**.

`fUnjoinOptions`

Specifies the unjoin options. If this parameter is NETSETUP_ACCT_DELETE, the account is disabled when the unjoin occurs. Note that this option does not delete the account. Currently, there are no other unjoin options defined.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes or one of the system error codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	A parameter is incorrect.
NERR_SetupNotJoined	The computer is not currently joined to a domain.

NERR_SetupDomainController	This computer is a domain controller and cannot be unjoined from a domain.

Remarks

Unjoining (and joining) a computer to a domain or workgroup can be performed only by a member of the Administrators local group on the target computer. If you call the **NetUnjoinDomain** function remotely, you must supply credentials because you cannot delegate credentials under these circumstances.

Different processes, or different threads of the same process, should not call the **NetUnjoinDomain** function at the same time. This situation can leave the computer in an inconsistent state.

A system reboot is required after calling the NetRenameMachineInDomain function for the operation to complete.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetJoinDomain

NetRenameMachineInDomain

Network Management Functions

Network Management Overview

NetValidateName function (lmjoin.h)

2/1/2021 • 6 minutes to read • Edit Online

The **NetValidateName** function verifies that a name is valid for name type specified(computer name, workgroup name, domain name, or DNS computer name).

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetValidateName(
  LPCWSTR            lpServer,
  LPCWSTR            lpName,
  LPCWSTR            lpAccount,
  LPCWSTR            lpPassword,
  NETSETUP_NAME_TYPE NameType
);
```

Parameters

`lpServer`

A pointer to a constant string that specifies the DNS or NetBIOS name of the computer on which to call the function. If this parameter is **NULL**, the local computer is used.

`lpName`

A pointer to a constant string that specifies the name to validate. Depending on the value specified in the *NameType* parameter, the *lpName* parameter can point to a computer name, workgroup name, domain name, or DNS computer name.

`lpAccount`

If the *lpName* parameter is a domain name, this parameter points to an account name to use when connecting to the domain controller. The string must specify either a domain NetBIOS name and user account (for example, "REDMOND\user") or the user principal name (UPN) of the user in the form of an Internet-style login name (for example, "someone@example.com"). If this parameter is **NULL**, the caller's context is used.

`lpPassword`

If the *lpAccount* parameter specifies an account name, this parameter must point to the password to use when connecting to the domain controller. Otherwise, this parameter must be **NULL**.

`NameType`

The type of the name passed in the *lpName* parameter to validate. This parameter can be one of the values from the NETSETUP_NAME_TYPE enumeration type defined in the *Lmjoin.h* header file.

Note that the *Lmjoin.h* header is automatically included by the *Lm.h* header file. The *Lmjoin.h* header files should not be used directly.

The following list shows the possible values for this parameter.

VALUE	MEANING

NetSetupUnknown 0	The nametype is unknown. If this value is used, the **NetValidateName** function fails with **ERROR_INVALID_PARAMETER**.
NetSetupMachine 1	Verify that the NetBIOS computer name is valid and that it is not in use.
NetSetupWorkgroup 2	Verify that the workgroup name is valid.
NetSetupDomain 3	Verify that the domain name exists and that it is a domain.
NetSetupNonExistentDomain 4	Verify that the domain name is not in use.
NetSetupDnsMachine 5	Verify that the DNS computer name is valid. This value is supported on Windows 2000 and later. The application must be compiled with **_WIN32_WINNT >= 0x0500** to use this value.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
DNS_ERROR_INVALID_NAME_CHAR	The DNS name contains an invalid character. This error is returned if the *NameType* parameter specified is **NetSetupDnsMachine** and the DNS name in the *lpName* parameter contains an invalid character.
DNS_ERROR_NON_RFC_NAME	The DNS name does not comply with RFC specifications. This error is returned if the *NameType* parameter specified is **NetSetupDnsMachine** and the DNS name in the *lpName* parameter does not comply with RFC specifications.
ERROR_DUP_NAME	A duplicate name already exists on the network.
ERROR_INVALID_COMPUTERNAME	The format of the specified computer name is not valid.
ERROR_INVALID_PARAMETER	A parameter is incorrect. This error is returned if the *lpName* parameter is **NULL** or the *NameType* parameter is specified as **NetSetupUnknown** or an unknown nametype.

ERROR_NO_SUCH_DOMAIN	The specified domain does not exist.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if a remote computer was specified in the *lpServer* parameter and this call is not supported on the remote computer.
NERR_InvalidComputer	The specified computer name is not valid. This error is returned if the *NameType* parameter specified is **NetSetupDnsMachine** or **NetSetupMachine** and the specified computer name is not valid.
NERR_InvalidWorkgroupName	The specified workgroup name is not valid. This error is returned if the *NameType* parameter specified is **NetSetupWorkgroup** and the specified workgroup name is not valid.
RPC_S_SERVER_UNAVAILABLE	The RPC server is not available. This error is returned if a remote computer was specified in the *lpServer* parameter and the RPC server is not available.
RPC_E_REMOTE_DISABLED	Remote calls are not allowed for this process. This error is returned if a remote computer was specified in the *lpServer* parameter and remote calls are not allowed for this process.

Remarks

The **NetValidateName** function validates a name based on the nametype specified.

If the *NameType* parameter is **NetSetupMachine**, the name passed in the *lpName* parameter must be syntactically correct as a NetBIOS name and the name must not currently be in use on the network.

If the *NameType* parameter is **NetSetupWorkgroup**, the name passed in the *lpName* parameter must be syntactically correct as a NetBIOS name, the name must not currently be in use on the network as a unique name, and the name must be different from the computer name.

If the *NameType* parameter is **NetSetupDomain**, the name passed in the *lpName* parameter must be syntactically correct as a NetBIOS or DNS name and the name must currently be registered as a domain name.

If the *NameType* parameter is **NetSetupNonExistentDomain**, the name passed in the *lpName* parameter must be syntactically correct as a NetBIOS or DNS name and the name must currently not be registered as a domain name.

If the *NameType* parameter is **NetSetupDnsMachine**, the name passed in the *lpName* parameter must be syntactically correct as a DNS name.

NetBIOS names are limited to maximum length of 16 characters.

No special group membership is required to successfully execute the **NetValidateName** function.

Examples

The following example validates a name for a specific type.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t * argv[])
{

    NET_API_STATUS nStatus;

    LPCWSTR lpServer = NULL;
    LPCWSTR lpName = NULL;
    LPCWSTR lpAccount = NULL;
    LPCWSTR lpPassword = NULL;
    DWORD dwNameType = NetSetupUnknown; // unknown name type

    if (argc != 3 && argc != 4 && argc != 6) {
        wprintf(L"Usage: %ws Server Name AccountName Password> nametype\n",
                argv[0]);
        wprintf(L"Usage: %ws Server Name nametype\n", argv[0]);
        wprintf(L"Usage: %ws Name nametype\n", argv[0]);
        wprintf(L"     %ws Client2 2\n", argv[0]);
        wprintf(L"     %ws Myserver Client2 3\n", argv[0]);
        wprintf(L"     %ws Myserver Client2 domain\\user password 3\n", argv[0]);
        exit(1);
    }
    // The request is not for the primary domain.
    //
    if (argc == 3) {
        lpName = argv[1];
        dwNameType = _wtoi(argv[2]);
    }

    if (argc == 4) {
        lpServer = argv[1];
        lpName = argv[2];
        dwNameType = _wtoi(argv[3]);
    }

    if (argc == 6) {
        lpServer = argv[1];
        lpName = argv[2];
        lpAccount = argv[3];
        lpPassword = argv[4];
        dwNameType = _wtoi(argv[5]);
    }

    wprintf(L"Calling NetValidateName with parameters\n");
    wprintf(L"    lpServer = %ws\n", lpServer);
    wprintf(L"    lpName = %ws\n", lpName);
    wprintf(L"    lpAccount = %ws\n", lpAccount);
    wprintf(L"    lpPassword = %ws\n", lpPassword);
    wprintf(L"    NameType  = %d ", dwNameType);
    switch (dwNameType) {
    case NetSetupUnknown:
        wprintf(L"(NetSetupUnknown)\n");
        break;
    case NetSetupMachine:
        wprintf(L"(NetSetupMachine)\n");
        break;
    case NetSetupWorkgroup:
        wprintf(L"(NetSetupWorkgroup)\n");
        break;
    case NetSetupDomain:
        wprintf(L"(NetSetupDomain)\n");
        break;
    case NetSetupNonExistentDomain:
        wprintf(L"(NetSetupNonExistentDomain)\n");
```

```
        break;
#if(_WIN32_WINNT >= 0x0500)
    case NetSetupDnsMachine:
        wprintf(L"(NetSetupDnsMachine)\n");
        break;
#endif
    default:
        wprintf(L"Other unknown nametype)\n");
        break;
    }

    //
    // Call the NetValidateName function to validate the name
    //
    nStatus = NetValidateName(lpServer,
                              lpName, lpAccount, lpPassword, (NETSETUP_NAME_TYPE) dwNameType);
    //
    // If the call succeeds,
    //
    if ((nStatus == NERR_Success)) {
        wprintf(L"NetValidateName was successful\n", nStatus);
    } else {
        wprintf(L"NetValidateName failed with error: %lu (0x%lx)\n", nStatus,
                nStatus);
        wprintf(L"   Error = ");
        switch (nStatus) {
        case ERROR_INVALID_PARAMETER:
            wprintf(L"ERROR_INVALID_PARAMETER\n");
            break;
        case ERROR_DUP_NAME:
            wprintf(L"ERROR_DUP_NAME\n");
            break;
        case ERROR_NO_SUCH_DOMAIN:
            wprintf(L"ERROR_NO_SUCH_DOMAIN\n");
            break;
        case ERROR_NOT_SUPPORTED:
            wprintf(L"ERROR_NOT_SUPPORTED\n");
            break;
        case ERROR_INVALID_COMPUTERNAME:
            wprintf(L"ERROR_INVALID_COMPUTERNAME\n");
            break;
        case DNS_ERROR_INVALID_NAME_CHAR:
            wprintf(L"DNS_ERROR_INVALID_NAME_CHAR\n");
            break;
        case DNS_ERROR_NON_RFC_NAME:
            wprintf(L"DNS_ERROR_NON_RFC_NAME\n");
            break;
        case NERR_InvalidComputer:
            wprintf(L"NERR_InvalidComputer\n");
            break;
        case NERR_InvalidWorkgroupName:
            wprintf(L"NERR_InvalidWorkgroupName\n");
            break;
        case RPC_S_SERVER_UNAVAILABLE:
            wprintf(L"RPC_S_SERVER_UNAVAILABLE\n");
            break;
        case RPC_E_REMOTE_DISABLED:
            wprintf(L"RPC_E_REMOTE_DISABLED\n");
            break;
        default:
            wprintf(L"Other error, see Winerror.h or lmerr.h)\n");
            break;
        }
    }

    return nStatus;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmjoin.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Directory Service Functions

NetGetJoinInformation

NetGetJoinableOUs

NetJoinDomain

NetRenameMachineInDomain

NetUnjoinDomain

Network Management Functions

Network Management Overview

lmmsg.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmmsg.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetMessageBufferSend	The NetMessageBufferSend function sends a buffer of information to a registered message alias.
NetMessageNameAdd	The NetMessageNameAdd function registers a message alias in the message name table. The function requires that the messenger service be started.
NetMessageNameDel	The NetMessageNameDel function deletes a message alias in the message name table. The function requires that the messenger service be started.
NetMessageNameEnum	The NetMessageNameEnum function lists the message aliases that receive messages on a specified computer. The function requires that the messenger service be started.
NetMessageNameGetInfo	The NetMessageNameGetInfo function retrieves information about a particular message alias in the message name table. The function requires that the messenger service be started.

Structures

TITLE	DESCRIPTION
MSG_INFO_0	The MSG_INFO_0 structure specifies a message alias.
MSG_INFO_1	The MSG_INFO_1 structure specifies a message alias. This structure exists only for compatibility. Message forwarding is not supported.

MSG_INFO_0 structure (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

The **MSG_INFO_0** structure specifies a message alias.

Syntax

```
typedef struct _MSG_INFO_0 {
  LPWSTR msgi0_name;
} MSG_INFO_0, *PMSG_INFO_0, *LPMSG_INFO_0;
```

Members

`msgi0_name`

Pointer to a Unicode string that specifies the alias to which the message is to be sent. The constant LEN specifies the maximum number of characters in the string.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmmsg.h (include Lm.h)

See also

Message Functions

NetMessageNameEnum

NetMessageNameGetInfo

Network Management Overview

Network Management Structures

MSG_INFO_1 structure (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

The **MSG_INFO_1** structure specifies a message alias. This structure exists only for compatibility. Message forwarding is not supported.

Syntax

```
typedef struct _MSG_INFO_1 {
  LPWSTR msgi1_name;
  DWORD  msgi1_forward_flag;
  LPWSTR msgi1_forward;
} MSG_INFO_1, *PMSG_INFO_1, *LPMSG_INFO_1;
```

Members

`msgi1_name`

Pointer to a Unicode string that specifies the alias to which the message is to be sent. The constant LEN specifies the maximum number of characters in the string.

`msgi1_forward_flag`

This member must be zero.

`msgi1_forward`

This member must be **NULL**.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmmsg.h (include Lm.h)

See also

Message Functions

NetMessageNameEnum

NetMessageNameGetInfo

Network Management Overview

Network Management Structures

NetMessageBufferSend function (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the messenger service is not supported.]

The **NetMessageBufferSend** function sends a buffer of information to a registered message alias.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetMessageBufferSend(
  LPCWSTR servername,
  LPCWSTR msgname,
  LPCWSTR fromname,
  LPBYTE  buf,
  DWORD   buflen
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`msgname`

Pointer to a constant string that specifies the message alias to which the message buffer should be sent.

`fromname`

Pointer to a constant string specifying who the message is from. If this parameter is **NULL**, the message is sent from the local computer name.

`buf`

Pointer to a buffer that contains the message text. For more information, see Network Management Function Buffers.

`buflen`

Specifies a value that contains the length, in bytes, of the message text pointed to by the *buf* parameter.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.

ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_NOT_SUPPORTED	This request is not supported. This error is returned on Windows Vista and later.
NERR_NameNotFound	The user name could not be found.
NERR_NetworkError	A general failure occurred in the network hardware.

Remarks

If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits only Domain Admins and Account Operators to call this function. On a member server or workstation, only Administrators and Server Operators can call this function. For more information, see Security Requirements for the Network Management Functions. For more information on ACLs and ACEs, see Access Control Model.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmmsg.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Message Functions

NetMessageNameAdd

NetMessageNameDel

NetMessageNameEnum

NetMessageNameGetInfo

Network Management Functions

Network Management Overview

NetMessageNameAdd function (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the messenger service is not supported.]

The **NetMessageNameAdd** function registers a message alias in the message name table. The function requires that the messenger service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetMessageNameAdd(
  LPCWSTR servername,
  LPCWSTR msgname
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`msgname`

Pointer to a constant string that specifies the message alias to add. The string cannot be more than 15 characters long.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.
ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_NOT_SUPPORTED	This request is not supported. This error is returned on Windows Vista and later.
NERR_AlreadyExists	The message alias already exists on this computer. For more information, see the following Remarks section.
NERR_DuplicateName	The name specified is already in use as a message alias on the network.

NERR_NetworkError	A general failure occurred in the network hardware.
NERR_TooManyNames	The maximum number of message aliases has been exceeded.

Remarks

Only members of the Administrators local group can successfully execute the **NetMessageNameAdd** function on a remote server.

The forward action flag is no longer a parameter to the LAN Manager 2.*x* **NetMessageNameAdd** function because message forwarding is no longer supported. If the **NetMessageNameAdd** function detects that a forwarded version of *msgname* exists on the network, the function will fail with error NERR_Already_Exists.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmmsg.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Message Functions

NetMessageNameDel

Network Management Functions

Network Management Overview

NetMessageNameDel function (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the messenger service is not supported.]

The **NetMessageNameDel** function deletes a message alias in the message name table. The function requires that the messenger service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetMessageNameDel(
  LPCWSTR servername,
  LPCWSTR msgname
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`msgname`

Pointer to a constant string that specifies the message alias to delete. The string cannot be more than 15 characters long.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.
ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_NOT_SUPPORTED	This request is not supported. This error is returned on Windows Vista and later.
NERR_DelComputerName	A message alias that is also a computer name cannot be deleted.
NERR_IncompleteDel	The message alias was not successfully deleted from all networks.

NERR_NameInUse	The message alias is currently in use. Try again later.
NERR_NotLocalName	The message alias is not on the local computer.

Remarks

Only members of the Administrators local group can successfully execute the **NetMessageNameDel** function on a remote server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmmsg.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Message Functions

NetMessageNameAdd

Network Management Functions

Network Management Overview

NetMessageNameEnum function (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the messenger service is not supported.]

The **NetMessageNameEnum** function lists the message aliases that receive messages on a specified computer. The function requires that the messenger service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetMessageNameEnum(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return message aliases. The *bufptr* parameter points to an array of MSG_INFO_0 structures.
1	Return message aliases. The *bufptr* parameter points to an array of MSG_INFO_1 structures. This level exists only for compatibility. Message forwarding is not supported.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of the returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`resume_handle`

Pointer to a value that contains a resume handle which is used to continue an existing message alias search. The handle should be zero on the first call and left unchanged for subsequent calls. If *resume_handle* is **NULL**, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
ERROR_NOT_SUPPORTED	This request is not supported. This error is returned on Windows Vista and later.
NERR_BufTooSmall	The supplied buffer is too small.

Remarks

Only members of the Administrators local group can successfully execute the **NetMessageNameEnum** function on a remote server.

To retrieve information about a particular message alias in the message name table, you can call the NetMessageNameGetInfo function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmmsg.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

MSG_INFO_0

MSG_INFO_1

Message Functions

NetMessageNameGetInfo

Network Management Functions

Network Management Overview

NetMessageNameGetInfo function (lmmsg.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is not supported as of Windows Vista because the messenger service is not supported.]

The **NetMessageNameGetInfo** function retrieves information about a particular message alias in the message name table. The function requires that the messenger service be started.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetMessageNameGetInfo(
  LPCWSTR servername,
  LPCWSTR msgname,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`msgname`

Pointer to a constant string that specifies the message alias for which to return information.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the message alias. The *bufptr* parameter points to a MSG_INFO_0 structure.
1	Return the message alias. The *bufptr* parameter points to a MSG_INFO_1 structure. This level exists only for compatibility. Message forwarding is not supported.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The caller does not have the appropriate access to complete the operation.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	A parameter is incorrect.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
ERROR_NOT_SUPPORTED	This request is not supported. This error is returned on Windows Vista and later.
NERR_NotLocalName	The message alias is not on the local computer.

Remarks

Only members of the Administrators local group can successfully execute the **NetMessageNameGetInfo** function on a remote server.

To list all the message aliases in a message name table, you can call the NetMessageNameEnum function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmmsg.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

MSG_INFO_0

MSG_INFO_1

Message Functions

NetMessageNameEnum

Network Management Functions

Network Management Overview

lmremutl.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmremutl.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetRemoteComputerSupports	The NetRemoteComputerSupports function queries the redirector to retrieve the optional features the remote system supports.
NetRemoteTOD	The NetRemoteTOD function returns the time of day information from a specified server.

Structures

TITLE	DESCRIPTION
TIME_OF_DAY_INFO	The TIME_OF_DAY_INFO structure contains information about the time of day from a remote server.

NetRemoteComputerSupports function (lmremutl.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetRemoteComputerSupports** function queries the redirector to retrieve the optional features the remote system supports. Features include Unicode, Remote Procedure Call (RPC), and Remote Administration Protocol support. The function establishes a network connection if one does not exist.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRemoteComputerSupports(
  LPCWSTR UncServerName,
  DWORD   OptionsWanted,
  LPDWORD OptionsSupported
);
```

Parameters

`UncServerName`

Pointer to a constant string that specifies the name of the remote server to query. If this parameter is **NULL**, the local computer is used.

`OptionsWanted`

Specifies a value that contains a set of bit flags indicating the features of interest. This parameter must be at least one of the following values.

VALUE	MEANING
SUPPORTS_REMOTE_ADMIN_PROTOCOL	Requests Remote Administration Protocol support.
SUPPORTS_RPC	Requests RPC support.
SUPPORTS_SAM_PROTOCOL	Requests Security Account Manager (SAM) support.
SUPPORTS_UNICODE	Requests Unicode standard support.
SUPPORTS_LOCAL	Requests support for the first three values listed in this table. If UNICODE is defined by the calling application, requests the four features listed previously.

`OptionsSupported`

Pointer to a value that receives a set of bit flags. The flags indicate which features specified by the *OptionsWanted* parameter are implemented on the computer specified by the *UncServerName* parameter. (All other bits are set to zero.)

The value of this parameter is valid only when the **NetRemoteComputerSupports** function returns NERR_Success.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	Either the *OptionsWanted* parameter or the *OptionsSupported* parameter is **NULL**; both parameters are required.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

No special group membership is required to successfully execute the **NetRemoteComputerSupports** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmremutl.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerGetInfo

Network Management Functions

Network Management Overview

Remote Utility Functions

NetRemoteTOD function (lmremutl.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetRemoteTOD** function returns the time of day information from a specified server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetRemoteTOD(
  LPCWSTR UncServerName,
  LPBYTE  *BufferPtr
);
```

Parameters

`UncServerName`

Pointer to a constant string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`BufferPtr`

Pointer to the address that receives the TIME_OF_DAY_INFO information structure. This buffer is allocated by the system and must be freed using the NetApiBufferFree function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to successfully execute the **NetRemoteTOD** function.

Examples

The following code sample demonstrates how to retrieve and print the current date and time with a call to the **NetRemoteTOD** function. To do this, the sample uses the TIME_OF_DAY_INFO structure. Finally, the sample frees the memory allocated for the information buffer.

```
#include <stdio.h>
#include <windows.h>
#include <lm.h>
#pragma comment(lib, "netapi32.lib")

#ifndef UNICODE
#define UNICODE
#endif

int wmain(int argc, wchar_t *argv[])
{
   LPTIME_OF_DAY_INFO pBuf = NULL;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Call the NetRemoteTOD function.
   //
   nStatus = NetRemoteTOD((LPCWSTR) pszServerName,
                          (LPBYTE *)&pBuf);
   //
   // If the function succeeds, display the current date and time.
   //
   if (nStatus == NERR_Success)
   {
      if (pBuf != NULL)
      {
         fprintf(stderr, "\nThe current date is: %d/%d/%d\n",
                 pBuf->tod_month, pBuf->tod_day, pBuf->tod_year);
         fprintf(stderr, "The current time is: %d:%d:%d\n",
                 pBuf->tod_hours, pBuf->tod_mins, pBuf->tod_secs);
      }
   }
   //
   // Otherwise, display a system error.
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated buffer.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Target Platform	Windows
Header	lmremutl.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Network Management Functions

Network Management Overview

Remote Utility Functions

TIME_OF_DAY_INFO

TIME_OF_DAY_INFO structure (lmremutl.h)

2/1/2021 • 2 minutes to read • Edit Online

The **TIME_OF_DAY_INFO** structure contains information about the time of day from a remote server.

Syntax

```
typedef struct _TIME_OF_DAY_INFO {
  DWORD tod_elapsedt;
  DWORD tod_msecs;
  DWORD tod_hours;
  DWORD tod_mins;
  DWORD tod_secs;
  DWORD tod_hunds;
  LONG  tod_timezone;
  DWORD tod_tinterval;
  DWORD tod_day;
  DWORD tod_month;
  DWORD tod_year;
  DWORD tod_weekday;
} TIME_OF_DAY_INFO, *PTIME_OF_DAY_INFO, *LPTIME_OF_DAY_INFO;
```

Members

`tod_elapsedt`

Type: **DWORD**

The number of seconds since 00:00:00, January 1, 1970, GMT.

`tod_msecs`

Type: **DWORD**

The number of milliseconds from an arbitrary starting point (system reset).

Typically, this member is read twice, once when the process begins and again at the end. To determine the elapsed time between the process's start and finish, you can subtract the first value from the second.

`tod_hours`

Type: **DWORD**

The current hour. Valid values are 0 through 23.

`tod_mins`

Type: **DWORD**

The current minute. Valid values are 0 through 59.

`tod_secs`

Type: **DWORD**

The current second. Valid values are 0 through 59.

`tod_hunds`

Type: **DWORD**

The current hundredth second (0.01 second). Valid values are 0 through 99.

`tod_timezone`

Type: **LONG**

The time zone of the server. This value is calculated, in minutes, from Greenwich Mean Time (GMT). For time zones west of Greenwich, the value is positive; for time zones east of Greenwich, the value is negative. A value of –1 indicates that the time zone is undefined.

`tod_tinterval`

Type: **DWORD**

The time interval for each tick of the clock. Each integral integer represents one ten-thousandth second (0.0001 second).

`tod_day`

Type: **DWORD**

The day of the month. Valid values are 1 through 31.

`tod_month`

Type: **DWORD**

The month of the year. Valid values are 1 through 12.

`tod_year`

Type: **DWORD**

The year.

`tod_weekday`

Type: **DWORD**

The day of the week. Valid values are 0 through 6, where 0 is Sunday, 1 is Monday, and so on.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmremutl.h (include Lm.h)

See also

NetRemoteTOD

Network Management Overview

Network Management Structures

Remote Utility functions

lmserver.h header

2/1/2021 • 5 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmserver.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetServerComputerNameAdd	The NetServerComputerNameAdd function enumerates the transports on which the specified server is active, and binds the emulated server name to each of the transports.
NetServerComputerNameDel	The NetServerComputerNameDel function causes the specified server to cease supporting the emulated server name set by a previous call to the NetServerComputerNameAdd function. The function does this by unbinding network transports from the emulated name.
NetServerDiskEnum	The NetServerDiskEnum function retrieves a list of disk drives on a server. The function returns an array of three-character strings (a drive letter, a colon, and a terminating null character).
NetServerEnum	The NetServerEnum function lists all servers of the specified type that are visible in a domain.
NetServerGetInfo	The NetServerGetInfo function retrieves current configuration information for the specified server.
NetServerSetInfo	The NetServerSetInfo function sets a server's operating parameters; it can set them individually or collectively. The information is stored in a way that allows it to remain in effect after the system has been reinitialized.
NetServerTransportAdd	The NetServerTransportAdd function binds the server to the transport protocol.
NetServerTransportAddEx	The NetServerTransportAddEx function binds the specified server to the transport protocol.
NetServerTransportDel	The NetServerTransportDel function unbinds (or disconnects) the transport protocol from the server. Effectively, the server can no longer communicate with clients using the specified transport protocol (such as TCP or XNS).
NetServerTransportEnum	The NetServerTransportEnum function supplies information about transport protocols that are managed by the server.

TITLE	DESCRIPTION
SetServiceBits	Registers a service type with the service control manager and the Server service.

Structures

TITLE	DESCRIPTION
SERVER_INFO_100	The SERVER_INFO_100 structure contains information about the specified server, including the name and platform.
SERVER_INFO_1005	The SERVER_INFO_1005 structure contains a comment that describes the specified server.
SERVER_INFO_101	The SERVER_INFO_101 structure contains information about the specified server, including name, platform, type of server, and associated software.
SERVER_INFO_1010	The SERVER_INFO_1010 structure contains the auto-disconnect time associated with the specified server.
SERVER_INFO_1016	The SERVER_INFO_1016 structure contains information about whether the server is visible to other computers in the same network domain.
SERVER_INFO_1017	The SERVER_INFO_1017 structure contains the network announce rate associated with the specified server.
SERVER_INFO_1018	The SERVER_INFO_1018 structure contains information about how much the announce rate can vary for the specified server.
SERVER_INFO_102	Contains information about the specified server, including name, platform, type of server, attributes, and associated software.
SERVER_INFO_1107	The SERVER_INFO_1107 structure specifies the number of users that can simultaneously log on to the specified server.
SERVER_INFO_1501	The SERVER_INFO_1501 structure specifies the number of files that can be open in one session on the specified server.
SERVER_INFO_1502	The SERVER_INFO_1502 structure specifies the maximum number of virtual circuits per client for the specified server.
SERVER_INFO_1503	The SERVER_INFO_1503 structure specifies the number of search operations that can be carried out simultaneously.
SERVER_INFO_1506	The SERVER_INFO_1506 structure contains information about the maximum number of work items the specified server can allocate.
SERVER_INFO_1509	The SERVER_INFO_1509 structure specifies the maximum raw mode buffer size.

TITLE	DESCRIPTION
SERVER_INFO_1510	The SERVER_INFO_1510 structure specifies the maximum number of users that can be logged on to the specified server using a single virtual circuit.
SERVER_INFO_1511	The SERVER_INFO_1511 structure specifies the maximum number of tree connections that users can make with a single virtual circuit.
SERVER_INFO_1512	The SERVER_INFO_1512 structure contains the maximum size of nonpaged memory that the specified server can allocate at a particular time.
SERVER_INFO_1513	The SERVER_INFO_1513 structure contains the maximum size of pageable memory that the specified server can allocate at a particular time.
SERVER_INFO_1515	The SERVER_INFO_1515 structure specifies whether the server should force a client to disconnect once the client's logon time has expired.
SERVER_INFO_1516	The SERVER_INFO_1516 structure specifies whether the server is a reliable time source.
SERVER_INFO_1518	The SERVER_INFO_1518 structure specifies whether the server is visible to LAN Manager 2.x clients.
SERVER_INFO_1523	The SERVER_INFO_1523 structure specifies the length of time the server retains information about incomplete search operations.
SERVER_INFO_1528	The SERVER_INFO_1528 structure specifies the period of time that the scavenger remains idle before waking up to service requests.
SERVER_INFO_1529	The SERVER_INFO_1529 structure specifies the minimum number of free receive work items the server requires before it begins allocating more items.
SERVER_INFO_1530	The SERVER_INFO_1530 structure specifies the minimum number of available receive work items the server requires to begin processing a server message block.
SERVER_INFO_1533	The SERVER_INFO_1533 structure specifies the maximum number of outstanding requests a client can send to the server.
SERVER_INFO_1536	The SERVER_INFO_1536 structure specifies whether the server allows clients to use opportunistic locks (oplocks) on files.
SERVER_INFO_1538	The SERVER_INFO_1538 structure specifies whether several MS-DOS File Control Blocks (FCBs) are placed in a single location.

TITLE	DESCRIPTION
SERVER_INFO_1539	The SERVER_INFO_1539 structure specifies whether the server processes raw Server Message Blocks (SMBs).
SERVER_INFO_1540	The SERVER_INFO_1540 structure specifies whether the server allows redirected server drives to be shared.
SERVER_INFO_1541	The SERVER_INFO_1541 structure specifies the minimum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.
SERVER_INFO_1542	The SERVER_INFO_1542 structure specifies the maximum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.
SERVER_INFO_1544	The SERVER_INFO_1544 structure specifies the initial number of tree connections to be allocated in the connection table.
SERVER_INFO_1550	The SERVER_INFO_1550 structure specifies the percentage of free disk space remaining before an alert message is sent.
SERVER_INFO_1552	The SERVER_INFO_1552 structure specifies the maximum time allowed for a link delay.
SERVER_INFO_402	Contains information about a specified server.
SERVER_INFO_403	The SERVER_INFO_403 structure contains information about a specified server.
SERVER_INFO_502	The SERVER_INFO_502 structure is obsolete. The structure contains information about a specified server.
SERVER_INFO_503	The SERVER_INFO_503 structure is obsolete. The structure contains information about the specified server.
SERVER_TRANSPORT_INFO_0	The SERVER_TRANSPORT_INFO_0 structure contains information about the specified transport protocol, including name, address, and location on the network.
SERVER_TRANSPORT_INFO_1	The SERVER_TRANSPORT_INFO_1 structure contains information about the specified transport protocol, including name and address. This information level is valid only for the NetServerTransportAddEx function.
SERVER_TRANSPORT_INFO_2	The SERVER_TRANSPORT_INFO_2 structure contains information about the specified transport protocol, including the transport name and address. This information level is valid only for the NetServerTransportAddEx function.
SERVER_TRANSPORT_INFO_3	The SERVER_TRANSPORT_INFO_3 structure contains information about the specified transport protocol, including name, address and password (credentials). This information level is valid only for the NetServerTransportAddEx function.

NetServerComputerNameAdd function (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServerComputerNameAdd** function enumerates the transports on which the specified server is active, and binds the emulated server name to each of the transports.

NetServerComputerNameAdd is a utility function that combines the functionality of the NetServerTransportEnum function and the NetServerTransportAddEx function.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerComputerNameAdd(
  LMSTR ServerName,
  LMSTR EmulatedDomainName,
  LMSTR EmulatedServerName
);
```

Parameters

`ServerName`

Pointer to a string that specifies the name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`EmulatedDomainName`

Pointer to a string that contains the domain name the specified server should use when announcing its presence using the *EmulatedServerName*. This parameter is optional.

`EmulatedServerName`

Pointer to a null-terminated character string that contains the emulated name the server should begin supporting in addition to the name specified by the *ServerName* parameter.

Return value

If the function succeeds, the return value is NERR_Success. Note that **NetServerComputerNameAdd** succeeds if the emulated server name specified is added to at least one transport.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_DUP_NAME	A duplicate name exists on the network.
ERROR_INVALID_DOMAINNAME	The domain name could not be found on the network.

ERROR_INVALID_PARAMETER	The specified parameter is invalid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerComputerNameAdd** function.

The server specified by the *ServerName* parameter continues to support all names it was supporting, and additionally begins to support new names supplied by successful calls to the **NetServerComputerNameAdd** function.

Name emulation that results from a call to **NetServerComputerNameAdd** ceases when the server reboots or restarts. To discontinue name emulation set by a previous call to **NetServerComputerNameAdd** without restarting or rebooting, you can call the NetServerComputerNameDel function.

The **NetServerComputerNameAdd** function is typically used when a system administrator replaces a server, but wants to keep the conversion transparent to users.

Examples

Following is an example of a call to the **NetServerComputerNameAdd** function requesting that \Server1 also respond to requests for \Server2.

```
NetServerComputerNameAdd (Server1, NULL, Server2);
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerComputerNameDel

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportEnum

NetServerComputerNameDel function (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServerComputerNameDel** function causes the specified server to cease supporting the emulated server name set by a previous call to the NetServerComputerNameAdd function. The function does this by unbinding network transports from the emulated name.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerComputerNameDel(
  LMSTR ServerName,
  LMSTR EmulatedServerName
);
```

Parameters

`ServerName`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`EmulatedServerName`

Pointer to a null-terminated character string that contains the emulated name the server should stop supporting. The server continues to support all other server names it was supporting.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	The specified parameter is invalid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
NERR_NetNameNotFound	The share name does not exist.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerComputerNameDel** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerComputerNameAdd

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportEnum

Network Management Functions

Network Management Overview

Server Functions

NetServerDiskEnum function (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetServerDiskEnum** function retrieves a list of disk drives on a server. The function returns an array of three-character strings (a drive letter, a colon, and a terminating null character).

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerDiskEnum(
  LMSTR   servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

A pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

The level of information required. A value of zero is the only valid level.

`bufptr`

A pointer to the buffer that receives the data. The data is an array of three-character strings (a drive letter, a colon, and a terminating null character). This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

The preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

> **Note** This parameter is currently ignored.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`resume_handle`

A pointer to a value that contains a resume handle which is used to continue an existing server disk search. The handle should be zero on the first call and left unchanged for subsequent calls. If the *resume_handle* parameter is a **NULL** pointer, then no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if a remote server was specified in *servername* parameter, the remote server only supports remote RPC calls using the legacy Remote Access Protocol mechanism, and this request is not supported.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerDiskEnum** function on a remote computer.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same results you can achieve by calling the network management server functions. For more information, see the IADsComputer interface reference.

Examples

The following code sample demonstrates how to call the **NetServerDiskEnum** function to retrieve a list of disk drives on a server. The sample calls **NetServerDiskEnum**, specifying the information level 0 (required). If there are entries to return, and the user has access to the information, it prints a list of the drives, in the format of a three-character string: a drive letter, a colon, and a terminating null character. The sample also prints the total number of entries that are available and a hint about the number of entries actually enumerated. Finally, the code sample frees the memory allocated for the buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
```

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

#pragma comment(lib, "netapi32.lib")

int wmain(int argc, wchar_t *argv[])
{
   const int ENTRY_SIZE = 3; // Drive letter, colon, NULL
   LPTSTR pBuf = NULL;
   DWORD dwLevel = 0; // level must be zero
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   NET_API_STATUS nStatus;
   LPWSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Call the NetServerDiskEnum function.
   //
   nStatus = NetServerDiskEnum(pszServerName,
                               dwLevel,
                               (LPBYTE *) &pBuf,
                               dwPrefMaxLen,
                               &dwEntriesRead,
                               &dwTotalEntries,
                               NULL);
   //
   // If the call succeeds,
   //
   if (nStatus == NERR_Success)
   {
      LPTSTR pTmpBuf;

      if ((pTmpBuf = pBuf) != NULL)
      {
         DWORD i;
         DWORD dwTotalCount = 0;
         //
         // Loop through the entries.
         //
         for (i = 0; i < dwEntriesRead; i++)
         {
            assert(pTmpBuf != NULL);

            if (pTmpBuf == NULL)
            {
               // On a remote computer, only members of the
               //  Administrators or the Server Operators
               //  local group can execute NetServerDiskEnum.
               //
               fprintf(stderr, "An access violation has occurred\n");
               break;
            }
            //
            // Print drive letter, colon, NULL for each drive;
            //   the number of entries actually enumerated; and
            //   the total number of entries available.
            //
```

```
            fwprintf(stdout, L"\tDisk: %S\n", pTmpBuf);

            pTmpBuf += ENTRY_SIZE;
            dwTotalCount++;
         }

         fprintf(stderr, "\nEntries enumerated: %d\n", dwTotalCount);
      }
   }
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   //
   // Free the allocated buffer.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

IADsComputer

NetServerEnum

Network Management Functions

Network Management Overview

Server Functions

NetServerEnum function (lmserver.h)

2/1/2021 • 7 minutes to read • Edit Online

The **NetServerEnum** function lists all servers of the specified type that are visible in a domain.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerEnum(
  LMCSTR  servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  DWORD   servertype,
  LMCSTR  domain,
  LPDWORD resume_handle
);
```

Parameters

`servername`

Reserved; must be **NULL**.

`level`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
100	Return server names and platform information. The *bufptr* parameter points to an array of SERVER_INFO_100 structures.
101	Return server names, types, and associated data. The *bufptr* parameter points to an array of SERVER_INFO_101 structures.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

The preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of visible servers and workstations on the network. Note that applications should consider this value only as a hint.

`servertype`

A value that filters the server entries to return from the enumeration. This parameter can be a combination of the following values defined in the *Lmserver.h* header file.

VALUE	MEANING
SV_TYPE_WORKSTATION 0x00000001	All workstations.
SV_TYPE_SERVER 0x00000002	All computers that run the Server service.
SV_TYPE_SQLSERVER 0x00000004	Any server that runs an instance of Microsoft SQL Server.
SV_TYPE_DOMAIN_CTRL 0x00000008	A server that is primary domain controller.
SV_TYPE_DOMAIN_BAKCTRL 0x00000010	Any server that is a backup domain controller.
SV_TYPE_TIME_SOURCE 0x00000020	Any server that runs the Timesource service.
SV_TYPE_AFP 0x00000040	Any server that runs the Apple Filing Protocol (AFP) file service.
SV_TYPE_NOVELL 0x00000080	Any server that is a Novell server.
SV_TYPE_DOMAIN_MEMBER 0x00000100	Any computer that is LAN Manager 2.x domain member.
SV_TYPE_PRINTQ_SERVER 0x00000200	Any computer that shares a print queue.

SV_TYPE_DIALIN_SERVER 0x00000400	Any server that runs a dial-in service.
SV_TYPE_XENIX_SERVER 0x00000800	Any server that is a Xenix server.
SV_TYPE_SERVER_UNIX 0x00000800	Any server that is a UNIX server. This is the same as the **SV_TYPE_XENIX_SERVER**.
SV_TYPE_NT 0x00001000	A workstation or server.
SV_TYPE_WFW 0x00002000	Any computer that runs Windows for Workgroups.
SV_TYPE_SERVER_MFPN 0x00004000	Any server that runs the Microsoft File and Print for NetWare service.
SV_TYPE_SERVER_NT 0x00008000	Any server that is not a domain controller.
SV_TYPE_POTENTIAL_BROWSER 0x00010000	Any computer that can run the browser service.
SV_TYPE_BACKUP_BROWSER 0x00020000	A computer that runs a browser service as backup.
SV_TYPE_MASTER_BROWSER 0x00040000	A computer that runs the master browser service.
SV_TYPE_DOMAIN_MASTER 0x00080000	A computer that runs the domain master browser.
SV_TYPE_SERVER_OSF 0x00100000	A computer that runs OSF/1.
SV_TYPE_SERVER_VMS 0x00200000	A computer that runs Open Virtual Memory System (VMS).

SV_TYPE_WINDOWS 0x00400000	A computer that runs Windows.
SV_TYPE_DFS 0x00800000	A computer that is the root of Distributed File System (DFS) tree.
SV_TYPE_CLUSTER_NT 0x01000000	Server clusters available in the domain.
SV_TYPE_TERMINALSERVER 0x02000000	A server running the Terminal Server service.
SV_TYPE_CLUSTER_VS_NT 0x04000000	Cluster virtual servers available in the domain. **Windows 2000:** This value is not supported.
SV_TYPE_DCE 0x10000000	A computer that runs IBM Directory and Security Services (DSS) or equivalent.
SV_TYPE_ALTERNATE_XPORT 0x20000000	A computer that over an alternate transport.
SV_TYPE_LOCAL_LIST_ONLY 0x40000000	Any computer maintained in a list by the browser. See the following Remarks section.
SV_TYPE_DOMAIN_ENUM 0x80000000	The primary domain.
SV_TYPE_ALL 0xFFFFFFFF	All servers. This is a convenience that will return all possible servers.

`domain`

A pointer to a constant string that specifies the name of the domain for which a list of servers is to be returned. The domain name must be a NetBIOS domain name (for example, microsoft). The **NetServerEnum** function does not support DNS-style names (for example, microsoft.com).

If this parameter is **NULL**, the primary domain is implied.

`resume_handle`

Reserved; must be set to zero.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes:

RETURN CODE/VALUE	DESCRIPTION
ERROR_ACCESS_DENIED 5	Access was denied.
ERROR_INVALID_PARAMETER 87	The parameter is incorrect.
ERROR_MORE_DATA 234	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NO_BROWSER_SERVERS_FOUND 6118	No browser servers found.
ERROR_NOT_SUPPORTED 50	The request is not supported.
NERR_RemoteErr 2127	A remote error occurred with no data returned by the server.
NERR_ServerNotStarted 2114	The server service is not started.
NERR_ServiceNotInstalled 2184	The service has not been started.
NERR_WkstaNotStarted 2138	The Workstation service has not been started. The local workstation service is used to communicate with a downlevel remote server.

Remarks

The **NetServerEnum** function is used to list all servers of the specified type that are visible in a domain. For example, an application can call **NetServerEnum** to list all domain controllers only or all servers that run instances of SQL server only.

An application combine the bit masks for various server types in the *servertype* parameter to list several types. For example, a value of SV_TYPE_WORKSTATION | SVTYPE_SERVER (0x00000003) combines the bit masks for SV_TYPE_WORKSTATION (0x00000001) and SV_TYPE_SERVER (0x00000002).

If you require more information for a specific server, call the WNetEnumResource function.

No special group membership is required to successfully execute the **NetServerEnum** function.

If you specify the value SV_TYPE_LOCAL_LIST_ONLY, the **NetServerEnum** function returns the list of servers that the browser maintains internally. This has meaning only on the master browser (or on a computer that has been the master browser in the past). The master browser is the computer that currently has rights to determine which computers can be servers or workstations on the network.

If there are no servers found that match the types specified in the *servertype* parameter, the **NetServerEnum** function returns the *bufptr* parameter as **NULL** and DWORD values pointed to by the *entriesread* and *totalentries* parameters are set to zero.

The **NetServerEnum** function depends on the browser service being installed and running. If no browser servers are found, then **NetServerEnum** fails with ERROR_NO_BROWSER_SERVERS_FOUND.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same function you can achieve by calling the network management server functions. For more information, see IADsComputer.

Examples

The following code sample demonstrates how to list all servers that are visible in a domain with a call to the **NetServerEnum** function. The sample calls **NetServerEnum**, specifying information level 101 (SERVER_INFO_101). If any servers are found, the sample code loops through the entries and prints the retrieved data. If the server is a domain controller, it identifies the server as either a primary domain controller (PDC) or a backup domain controller (BDC). The sample also prints the total number of entries available and a hint about the number of entries actually enumerated, warning the user if all entries were not enumerated. Finally, the sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t * argv[])
{
    LPSERVER_INFO_101 pBuf = NULL;
    LPSERVER_INFO_101 pTmpBuf;
    DWORD dwLevel = 101;
    DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
    DWORD dwEntriesRead = 0;
    DWORD dwTotalEntries = 0;
    DWORD dwTotalCount = 0;
    DWORD dwServerType = SV_TYPE_SERVER;        // all servers
    DWORD dwResumeHandle = 0;
    NET_API_STATUS nStatus;
    LPWSTR pszServerName = NULL;
    LPWSTR pszDomainName = NULL;
    DWORD i;

    if (argc > 2)
    {
        fwprintf(stderr, L"Usage: %s [DomainName]\n", argv[0]);
        exit(1);
    }
    // The request is not for the primary domain.
    //
    if (argc == 2)
        pszDomainName = argv[1];
    //
    // Call the NetServerEnum function to retrieve information
    //  for all servers, specifying information level 101.
```

```
    //
    nStatus = NetServerEnum(pszServerName,
                            dwLevel,
                            (LPBYTE *) & pBuf,
                            dwPrefMaxLen,
                            &dwEntriesRead,
                            &dwTotalEntries,
                            dwServerType,
                            pszDomainName,
                            &dwResumeHandle);
    //
    // If the call succeeds,
    //
    if ((nStatus == NERR_Success) || (nStatus == ERROR_MORE_DATA)) {
        if ((pTmpBuf = pBuf) != NULL) {
            //
            // Loop through the entries and
            //  print the data for all server types.
            //
            for (i = 0; i < dwEntriesRead; i++) {
                assert(pTmpBuf != NULL);

                if (pTmpBuf == NULL) {
                    fprintf(stderr, "An access violation has occurred\n");
                    break;
                }

                printf("\tPlatform: %d\n", pTmpBuf->sv101_platform_id);
                wprintf(L"\tName:     %s\n", pTmpBuf->sv101_name);
                printf("\tVersion:  %d.%d\n",
                       pTmpBuf->sv101_version_major,
                       pTmpBuf->sv101_version_minor);
                printf("\tType:     %d", pTmpBuf->sv101_type);
                //
                // Check to see if the server is a domain controller;
                //  if so, identify it as a PDC or a BDC.
                //
                if (pTmpBuf->sv101_type & SV_TYPE_DOMAIN_CTRL)
                    wprintf(L" (PDC)");
                else if (pTmpBuf->sv101_type & SV_TYPE_DOMAIN_BAKCTRL)
                    wprintf(L" (BDC)");

                printf("\n");
                //
                // Also print the comment associated with the server.
                //
                wprintf(L"\tComment:  %s\n\n", pTmpBuf->sv101_comment);

                pTmpBuf++;
                dwTotalCount++;
            }
            // Display a warning if all available entries were
            //  not enumerated, print the number actually
            //  enumerated, and the total number available.

            if (nStatus == ERROR_MORE_DATA) {
                fprintf(stderr, "\nMore entries available!!!\n");
                fprintf(stderr, "Total entries: %d", dwTotalEntries);
            }

            printf("\nEntries enumerated: %d\n", dwTotalCount);

        } else {
            printf("No servers were found\n");
            printf("The buffer (bufptr) returned was NULL\n");
            printf("  entriesread: %d\n", dwEntriesRead);
            printf("  totalentries: %d\n", dwEntriesRead);
        }
```

```
    } else
        fprintf(stderr, "NetServerEnum failed with error: %d\n", nStatus);
    //
    // Free the allocated buffer.
    //
    if (pBuf != NULL)
        NetApiBufferFree(pBuf);

    return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetQueryDisplayInformation

NetServerDiskEnum

Network Management Functions

Network Management Overview

SERVER_INFO_100

SERVER_INFO_101

Server Functions

NetServerGetInfo function (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServerGetInfo** function retrieves current configuration information for the specified server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerGetInfo(
  LMSTR  servername,
  DWORD  level,
  LPBYTE *bufptr
);
```

Parameters

`servername`

Pointer to a string that specifies the name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
100	Return the server name and platform information. The *bufptr* parameter points to a SERVER_INFO_100 structure.
101	Return the server name, type, and associated software. The *bufptr* parameter points to a SERVER_INFO_101 structure.
102	Return the server name, type, associated software, and other attributes. The *bufptr* parameter points to a SERVER_INFO_102 structure.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter.

This buffer is allocated by the system and must be freed using the NetApiBufferFree function.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.

ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	The specified parameter is invalid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

Only the Administrators or Server Operators local group, or those with Print or Server Operator group membership, can successfully execute the **NetServerGetInfo** function at level 102. No special group membership is required for level 100 or level 101 calls.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management server functions. For more information, see IADsComputer.

Examples

The following code sample demonstrates how to retrieve current configuration information for a server using a call to the **NetServerGetInfo** function. The sample calls **NetServerGetInfo**, specifying information level 101 (SERVER_INFO_101). If the call succeeds, the code attempts to identify the type of server. Finally, the sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 101;
   LPSERVER_INFO_101 pBuf = NULL;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Call the NetServerGetInfo function, specifying level 101.
   //
   nStatus = NetServerGetInfo(pszServerName,
                              dwLevel,
                              (LPBYTE *)&pBuf);
   //
   // If the call succeeds,
   //
   if (nStatus == NERR_Success)
   {
      //
      // Check for the type of server.
      //
      if ((pBuf->sv101_type & SV_TYPE_DOMAIN_CTRL) ||
         (pBuf->sv101_type & SV_TYPE_DOMAIN_BAKCTRL) ||
         (pBuf->sv101_type & SV_TYPE_SERVER_NT))
         printf("This is a server\n");
      else
         printf("This is a workstation\n");
   }
   //
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetRemoteComputerSupports

NetServerSetInfo

Network Management Functions

Network Management Overview

SERVER_INFO_100

SERVER_INFO_101

SERVER_INFO_102

Server Functions

NetServerSetInfo function (lmserver.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetServerSetInfo** function sets a server's operating parameters; it can set them individually or collectively. The information is stored in a way that allows it to remain in effect after the system has been reinitialized.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerSetInfo(
  LMSTR   servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD ParmError
);
```

Parameters

`servername`

Pointer to a string that specifies the name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
101	Specifies the server name, type, and associated software. The *buf* parameter points to a SERVER_INFO_101 structure.
102	Specifies the server name, type, associated software, and other attributes. The *buf* parameter points to a SERVER_INFO_102 structure.
402	Specifies detailed information about the server. The *buf* parameter points to a SERVER_INFO_402 structure.
403	Specifies detailed information about the server. The *buf* parameter points to a SERVER_INFO_403 structure.

In addition, levels 1001-1006, 1009-1011, 1016-1018, 1021, 1022, 1028, 1029, 1037, and 1043 are valid based on the restrictions for LAN Manager systems.

`buf`

Pointer to a buffer that receives the server information. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`ParmError`

Pointer to a value that receives the index of the first member of the server information structure that causes the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	The specified parameter is invalid. For more information, see the following Remarks section.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerSetInfo** function.

If you are programming for Active Directory, you may be able to call certain Active Directory Service Interface (ADSI) methods to achieve the same functionality you can achieve by calling the network management server functions. For more information, see IADsComputer.

If the **NetServerSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *ParmError* parameter to indicate the first member of the server information structure that is invalid. (A server information structure begins with SERVER_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *ParmError* parameter and the corresponding structure member that is in error. (The prefix sv*_ indicates that the member can begin with multiple prefixes, for example, sv101_ or sv402_.)

VALUE	MEMBER
SV_PLATFORM_ID_PARMNUM	sv*_platform_id
SV_NAME_PARMNUM	sv*_name
SV_VERSION_MAJOR_PARMNUM	sv*_version_major
SV_VERSION_MINOR_PARMNUM	sv*_version_minor
SV_TYPE_PARMNUM	sv*_type
SV_COMMENT_PARMNUM	sv*_comment

SV_USERS_PARMNUM	sv*_users
SV_DISC_PARMNUM	sv*_disc
SV_HIDDEN_PARMNUM	sv*_hidden
SV_ANNOUNCE_PARMNUM	sv*_announce
SV_ANNDELTA_PARMNUM	sv*_anndelta
SV_USERPATH_PARMNUM	sv*_userpath
SV_ULIST_MTIME_PARMNUM	sv*_ulist_mtime
SV_GLIST_MTIME_PARMNUM	sv*_glist_mtime
SV_ALIST_MTIME_PARMNUM	sv*_alist_mtime
SV_ALERTS_PARMNUM	sv*_alerts
SV_SECURITY_PARMNUM	sv*_security
SV_NUMADMIN_PARMNUM	sv*_numadmin
SV_LANMASK_PARMNUM	sv*_lanmask
SV_GUESTACC_PARMNUM	sv*_guestacc
SV_CHDEVQ_PARMNUM	sv*_chdevq
SV_CHDEVJOBS_PARMNUM	sv*_chdevjobs
SV_CONNECTIONS_PARMNUM	sv*_connections
SV_SHARES_PARMNUM	sv*_shares
SV_OPENFILES_PARMNUM	sv*_openfiles
SV_SESSOPENS_PARMNUM	sv*_sessopens
SV_SESSVCS_PARMNUM	sv*_sessvcs
SV_SESSREQS_PARMNUM	sv*_sessreqs
SV_OPENSEARCH_PARMNUM	sv*_opensearch
SV_ACTIVELOCKS_PARMNUM	sv*_activelocks
SV_NUMREQBUF_PARMNUM	sv*_numreqbuf
SV_SIZREQBUF_PARMNUM	sv*_sizreqbuf

SV_NUMBIGBUF_PARMNUM	sv*_numbigbuf
SV_NUMFILETASKS_PARMNUM	sv*_numfiletasks
SV_ALERTSCHED_PARMNUM	sv*_alertsched
SV_ERRORALERT_PARMNUM	sv*_erroralert
SV_LOGONALERT_PARMNUM	sv*_logonalert
SV_ACCESSALERT_PARMNUM	sv*_accessalert
SV_DISKALERT_PARMNUM	sv*_diskalert
SV_NETIOALERT_PARMNUM	sv*_netioalert
SV_MAXAUDITSZ_PARMNUM	sv*_maxauditsz
SV_SRVHEURISTICS_PARMNUM	sv*_srvheuristics
SV_TIMESOURCE_PARMNUM	sv*_timesource

Examples

The following code sample demonstrates how to call the **NetServerSetInfo** function. The sample calls **NetServerSetInfo**, specifying the *level* parameter as 1005 (required) to set the **sv1005_comment** member of the SERVER_INFO_1005 structure.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 1005;
   SERVER_INFO_1005 si;
   NET_API_STATUS nStatus;

   if (argc != 3)
   {
      fwprintf(stderr, L"Usage: %s \\\\ServerName Comment\n", argv[0]);
      exit(1);
   }
   //
   // Fill in SERVER_INFO_1005 structure member.
   //
   si.sv1005_comment = (LPTSTR) argv[2];
   //
   // Call the NetServerSetInfo function,
   //  specifying level 1005.
   //
   nStatus = NetServerSetInfo(argv[1],
                              dwLevel,
                              (LPBYTE)&si,
                              NULL);
   //
   // Display the result of the call.
   //
   if (nStatus == NERR_Success)
      fwprintf(stderr, L"Comment reset\n", argv[2]);
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerGetInfo

Network Management Functions

Network Management Overview

SERVER_INFO_101

SERVER_INFO_102

SERVER_INFO_402

SERVER_INFO_403

Server Functions

NetServerTransportAdd function (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServerTransportAdd** function binds the server to the transport protocol.

The extended function NetServerTransportAddEx allows the calling application to specify the SERVER_TRANSPORT_INFO_1, SERVER_TRANSPORT_INFO_2, and SERVER_TRANSPORT_INFO_3 information levels.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerTransportAdd(
  LMSTR  servername,
  DWORD  level,
  LPBYTE bufptr
);
```

Parameters

`servername`

A pointer to a string that specifies the name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be the following value.

VALUE	MEANING
0	Specifies information about the transport protocol, including name, address, and location on the network. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_0 structure.

`bufptr`

A pointer to the buffer that contains the data.

For more information, see Network Management Function Buffers.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.

ERROR_DUP_NAME	A duplicate name exists on the network.
ERROR_INVALID_DOMAINNAME	The domain name could not be found on the network.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	A parameter is invalid. This error is returned if the **svti0_transportname** or **svti0_transportaddress** member in the SERVER_TRANSPORT_INFO_0 structure pointed to by the *bufptr* parameter is **NULL**. This error is also returned if the **svti0_transportaddresslength** member in the **SERVER_TRANSPORT_INFO_0** structure pointed to by the *bufptr* parameter is zero or larger than MAX_PATH (defined in the Windef.h header file). This error is also returned for other invalid parameters.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerTransportAdd** function.

If you add a transport protocol to a server using a call to the **NetServerTransportAdd** function, the connection will not remain after the server reboots or restarts.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerComputerNameAdd

NetServerComputerNameDel

NetServerTransportAddEx

NetServerTransportDel

NetServerTransportEnum

Network Management Functions

Network Management Overview

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

SERVER_TRANSPORT_INFO_2

SERVER_TRANSPORT_INFO_3

Server and Workstation Transport Functions

NetServerTransportAddEx function (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetServerTransportAddEx** function binds the specified server to the transport protocol. This extended function allows the calling application to specify the SERVER_TRANSPORT_INFO_0, SERVER_TRANSPORT_INFO_1, SERVER_TRANSPORT_INFO_2, or SERVER_TRANSPORT_INFO_3 information levels.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerTransportAddEx(
  LMSTR  servername,
  DWORD  level,
  LPBYTE bufptr
);
```

Parameters

`servername`

A pointer to a string that specifies the name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies a value that indicates the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies information about the transport protocol, including name, address, and location on the network. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_0 structure.
1	Specifies information about the transport protocol, including name, address, network location, and domain. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_1 structure.
2	Specifies the same information as level 1, with the addition of an **svti2_flags** member. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_2 structure.
3	Specifies the same information as level 2, with the addition of credential information. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_3 structure.

`bufptr`

A pointer to the buffer that contains the data. The format of this data depends on the value of the *level* parameter.

For more information, see Network Management Function Buffers.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_DUP_NAME	A duplicate name exists on the network.
ERROR_INVALID_DOMAINNAME	The domain name could not be found on the network.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	A parameter is invalid. This error is returned if the transport name or transport address member in the SERVER_TRANSPORT_INFO_0, SERVER_TRANSPORT_INFO_1, SERVER_TRANSPORT_INFO_2, or SERVER_TRANSPORT_INFO_3 structure pointed to by the *bufptr* parameter is **NULL**. This error is also returned if the transport address length member in the **SERVER_TRANSPORT_INFO_0**, **SERVER_TRANSPORT_INFO_1**, **SERVER_TRANSPORT_INFO_2**, or **SERVER_TRANSPORT_INFO_3** structure pointed to by the *bufptr* parameter is zero or larger than MAX_PATH (defined in the *Windef.h* header file). This error is also returned if the flags member of the **SERVER_TRANSPORT_INFO_2**, or **SERVER_TRANSPORT_INFO_3** structure pointed to by the *bufptr* parameter contains an illegal value. This error is also returned for other invalid parameters.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerTransportAddEx** function.

If you add a transport protocol to a server using a call to the **NetServerTransportAddEx** function, the connection will not remain after the server reboots or restarts.

The NetServerComputerNameAdd function is a utility function. It combines the features of the NetServerTransportEnum function and the **NetServerTransportAddEx** function, allowing you to specify an emulated server name.

On Windows Server 2008 and Windows Vista with Service Pack 1 (SP1), every name registered with the Windows remote file server (SRV) is designated as either a scoped name or a non-scoped name. Every share that is added to the system will then either be attached to all of the non-scoped names, or to a single scoped name. Applications that wish to use the scoping features are responsible for both registering the new name as a scoped endpoint and then creating the shares with an appropriate scope. In this way, legacy uses of the Network Management and Network Share Management functions are not affected in any way since they continue to register shares and names as non-scoped names.

A scoped endpoint is created by calling the **NetServerTransportAddEx** function with the *level* parameter set to 2 and the *bufptr* parameter pointed to a SERVER_TRANSPORT_INFO_2 structure with the **SVTI2_SCOPED_NAME** bit value set in **svti2_flags** member. A scoped endpoint is also created by calling the **NetServerTransportAddEx** function with the *level* parameter set to 3 and the *bufptr* parameter pointed to a SERVER_TRANSPORT_INFO_3 structure with the **SVTI2_SCOPED_NAME** bit value set in **svti3_flags** member.

When the **SVTI2_SCOPED_NAME** bit value is set for a transport, then shares can be added with a corresponding server name (the **shi503_servername** member of the SHARE_INFO_503 structure) in a scoped fashion using the NetShareAdd function. If there is no transport registered with the **SVTI2_SCOPED_NAME** bit value and the name provided in **shi503_servername** member, then the share add in a scoped fashion will not succeed.

The NetShareAdd function is used to add a scoped share on a remote server specified in the *servername* parameter. The remote server specified in the **shi503_servername** member of the SHARE_INFO_503 passed in the *bufptr* parameter must have been bound to a transport protocol using the **NetServerTransportAddEx** function as a scoped endpoint. The **SVTI2_SCOPED_NAME** flag must have been specified in the **shi503_servername** member of the SERVER_TRANSPORT_INFO_2 or SERVER_TRANSPORT_INFO_3 structure for the transport protocol. The NetShareDelEx function is used to delete a scoped share. The NetShareGetInfo and NetShareSetInfo functions are to used to get and set information on a scoped share.

Scoped endpoints are generally used by the cluster namespace.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerComputerNameAdd

NetServerComputerNameDel

NetServerTransportAdd

NetServerTransportDel

NetServerTransportEnum

NetShareAdd

NetShareDelEx

NetShareGetInfo

NetShareSetInfo

Network Management Functions

Network Management Overview

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

SERVER_TRANSPORT_INFO_2

SERVER_TRANSPORT_INFO_3

SHARE_INFO_503

Server and Workstation Transport Functions

NetServerTransportDel function (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServerTransportDel** function unbinds (or disconnects) the transport protocol from the server. Effectively, the server can no longer communicate with clients using the specified transport protocol (such as TCP or XNS).

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerTransportDel(
  LMSTR  servername,
  DWORD  level,
  LPBYTE bufptr
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies information about the transport protocol, including name, address, and location on the network. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_0 structure.
1	Specifies information about the transport protocol, including name, address, network location, and domain. The *bufptr* parameter points to a SERVER_TRANSPORT_INFO_1 structure.

`bufptr`

Pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION

ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_INVALID_PARAMETER	The specified parameter is invalid.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
NERR_NetNameNotFound	The share name does not exist.

Remarks

Only members of the Administrators or Server Operators local group can successfully execute the **NetServerTransportDel** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetServerTransportAdd

Network Management Functions

Network Management Overview

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

Server and Workstation Transport Functions

NetServerTransportEnum function (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetServerTransportEnum** function supplies information about transport protocols that are managed by the server.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServerTransportEnum(
  LMSTR   servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return information about the transport protocol, including name, address, and location on the network. The *bufptr* parameter points to an array of SERVER_TRANSPORT_INFO_0 structures.
1	Return information about the transport protocol, including name, address, network location, and domain. The *bufptr* parameter points to an array of SERVER_TRANSPORT_INFO_1 structures.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and

Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`resume_handle`

Pointer to a value that contains a resume handle which is used to continue an existing server transport search. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_LEVEL	The value specified for the *level* parameter is invalid.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory is available.
NERR_BufTooSmall	The supplied buffer is too small.

Remarks

Only Authenticated Users can successfully call this function.**Windows XP/2000:** No special group membership is required to successfully execute this function.

Examples

The following code sample demonstrates how to retrieve information about transport protocols that are managed by the server, using a call to the **NetServerTransportEnum** function. The sample calls **NetServerTransportEnum**, specifying information level 0 (SERVER_TRANSPORT_INFO_0). The sample prints the name of each transport protocol and the total number enumerated. Finally, the code sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

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

int wmain(int argc, wchar_t *argv[])
{
   LPSERVER_TRANSPORT_INFO_0 pBuf = NULL;
   LPSERVER_TRANSPORT_INFO_0 pTmpBuf;
   DWORD dwLevel = 0;
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   DWORD dwResumeHandle = 0;
   DWORD dwTotalCount = 0;
   NET_API_STATUS nStatus;
   LPTSTR pszServerName = NULL;
   DWORD i;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = (LPTSTR) argv[1];
   //
   // Call the NetServerTransportEnum function; specify level 0.
   //
   do // begin do
   {
      nStatus = NetServerTransportEnum(pszServerName,
                                       dwLevel,
                                       (LPBYTE *) &pBuf,
                                       dwPrefMaxLen,
                                       &dwEntriesRead,
                                       &dwTotalEntries,
                                       &dwResumeHandle);
      //
      // If the call succeeds,
      //
      if ((nStatus == NERR_Success) || (nStatus == ERROR_MORE_DATA))
      {
         if ((pTmpBuf = pBuf) != NULL)
         {
            //
            // Loop through the entries;
            //  process access errors.
            //
            for (i = 0; i < dwEntriesRead; i++)
            {
               assert(pTmpBuf != NULL);

               if (pTmpBuf == NULL)
               {
                  fprintf(stderr, "An access violation has occurred\n");
                  break;
               }
               //
               // Print the transport protocol name.
               //
               wprintf(L"\tTransport: %s\n", pTmpBuf->svti0_transportname);

               pTmpBuf++;
               dwTotalCount++;
            }
         }
      }
      //
```

```
      //
      // Otherwise, indicate a system error.
      //
      else
         fprintf(stderr, "A system error has occurred: %d\n", nStatus);

      //
      // Free the allocated buffer.
      //
      if (pBuf != NULL)
      {
         NetApiBufferFree(pBuf);
         pBuf = NULL;
      }
   //
   // Continue to call NetServerTransportEnum while
   //  there are more entries.
   //
   }
   while (nStatus == ERROR_MORE_DATA); // end do

   // Check again for an allocated buffer.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);
   //
   // Print the final count of transports enumerated.
   //
   fprintf(stderr, "\nTotal of %d entries enumerated\n", dwTotalCount);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmserver.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

Network Management Functions

Network Management Overview

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

Server and Workstation Transport Functions

SERVER_INFO_100 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_100** structure contains information about the specified server, including the name and platform.

Syntax

```
typedef struct _SERVER_INFO_100 {
  DWORD sv100_platform_id;
  LMSTR sv100_name;
} SERVER_INFO_100, *PSERVER_INFO_100, *LPSERVER_INFO_100;
```

Members

`sv100_platform_id`

Type: **DWORD**

The information level to use for platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.
PLATFORM_ID_VMS 700	The VMS platform.

`sv100_name`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of the server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerDiskEnum

NetServerEnum

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1005 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1005** structure contains a comment that describes the specified server.

Syntax

```
typedef struct _SERVER_INFO_1005 {
  LMSTR sv1005_comment;
} SERVER_INFO_1005, *PSERVER_INFO_1005, *LPSERVER_INFO_1005;
```

Members

`sv1005_comment`

Pointer to a string that contains a comment describing the server. The comment can be null.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_101 structure (lmserver.h)

2/1/2021 • 3 minutes to read • Edit Online

The **SERVER_INFO_101** structure contains information about the specified server, including name, platform, type of server, and associated software.

Syntax

```
typedef struct _SERVER_INFO_101 {
  DWORD sv101_platform_id;
  LMSTR sv101_name;
  DWORD sv101_version_major;
  DWORD sv101_version_minor;
  DWORD sv101_type;
  LMSTR sv101_comment;
} SERVER_INFO_101, *PSERVER_INFO_101, *LPSERVER_INFO_101;
```

Members

`sv101_platform_id`

Type: **DWORD**

The information level to use for platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.
PLATFORM_ID_VMS 700	The VMS platform.

`sv101_name`

Type: **LPWSTR**

A pointer to a Unicode string specifying the name of a server.

`sv101_version_major`

Type: **DWORD**

The major version number and the server type.

The major release version number of the operating system is specified in the least significant 4 bits. The server type is specified in the most significant 4 bits. The **MAJOR_VERSION_MASK** bitmask defined in the *Lmserver.h* header should be used by an application to obtain the major version number from this member.

`sv101_version_minor`

Type: **DWORD**

The minor release version number of the operating system.

`sv101_type`

Type: **DWORD**

The type of software the computer is running.

Possible values for this member are listed in the *Lmserver.h* header file. This member can be a combination of some of the following values.

VALUE	MEANING
SV_TYPE_WORKSTATION 0x00000001	A workstation.
SV_TYPE_SERVER 0x00000002	A server.
SV_TYPE_SQLSERVER 0x00000004	A server running with Microsoft SQL Server.
SV_TYPE_DOMAIN_CTRL 0x00000008	A primary domain controller.
SV_TYPE_DOMAIN_BAKCTRL 0x00000010	A backup domain controller.
SV_TYPE_TIME_SOURCE 0x00000020	A server running the Timesource service.
SV_TYPE_AFP 0x00000040	A server running the Apple Filing Protocol (AFP) file service.

SV_TYPE_NOVELL 0x00000080	A Novell server.
SV_TYPE_DOMAIN_MEMBER 0x00000100	A LAN Manager 2.x domain member.
SV_TYPE_PRINTQ_SERVER 0x00000200	A server that shares a print queue.
SV_TYPE_DIALIN_SERVER 0x00000400	A server that runs a dial-in service.
SV_TYPE_XENIX_SERVER 0x00000800	A Xenix or Unix server.
SV_TYPE_NT 0x00001000	A workstation or server.
SV_TYPE_WFW 0x00002000	A computer that runs Windows for Workgroups.
SV_TYPE_SERVER_MFPN 0x00004000	A server that runs the Microsoft File and Print for NetWare service.
SV_TYPE_SERVER_NT 0x00008000	Any server that is not a domain controller.
SV_TYPE_POTENTIAL_BROWSER 0x00010000	A computer that can run the browser service.
SV_TYPE_BACKUP_BROWSER 0x00020000	A server running a browser service as backup.
SV_TYPE_MASTER_BROWSER 0x00040000	A server running the master browser service.
SV_TYPE_DOMAIN_MASTER 0x00080000	A server running the domain master browser.

SV_TYPE_SERVER_OSF 0x00100000	A computer that runs OSF.
SV_TYPE_SERVER_VMS 0x00200000	A computer that runs VMS.
SV_TYPE_WINDOWS 0x00400000	A computer that runs Windows.
SV_TYPE_DFS 0x00800000	A server that is the root of a DFS tree.
SV_TYPE_CLUSTER_NT 0x01000000	A server cluster available in the domain.
SV_TYPE_TERMINALSERVER 0x02000000	A server that runs the Terminal Server service.
SV_TYPE_CLUSTER_VS_NT 0x04000000	Cluster virtual servers available in the domain. **Windows 2000:** This value is not supported.
SV_TYPE_DCE 0x10000000	A server that runs the DCE Directory and Security Services or equivalent.
SV_TYPE_ALTERNATE_XPORT 0x20000000	A server that is returned by an alternate transport.
SV_TYPE_LOCAL_LIST_ONLY 0x40000000	A server that is maintained by the browser.
SV_TYPE_DOMAIN_ENUM 0x80000000	A primary domain.

The **SV_TYPE_ALL** constant is defined to 0xFFFFFFFF in the *Lmserver.h* header file. This constant can be used to check for all server types when used with the NetServerEnumfunction.

`sv101_comment`

Type: **LPWSTR**

A pointer to a Unicode string specifying a comment describing the server. The comment can be null.

Remarks

To retrieve a value that indicates whether a share is the root volume in a Dfs tree structure, you must call the NetShareGetInfo function and specify information level 1005.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerEnum

NetServerGetInfo

NetServerSetInfo

NetShareGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1010 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1010** structure contains the auto-disconnect time associated with the specified server.

Syntax

```
typedef struct _SERVER_INFO_1010 {
  LONG sv1010_disc;
} SERVER_INFO_1010, *PSERVER_INFO_1010, *LPSERVER_INFO_1010;
```

Members

`sv1010_disc`

Specifies the auto-disconnect time, in minutes.

If a session is idle longer than the period of time specified by this member, the server disconnects the session. If the value of this member is SV_NODISC, auto-disconnect is not enabled.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1016 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1016** structure contains information about whether the server is visible to other computers in the same network domain.

Syntax

```
typedef struct _SERVER_INFO_1016 {
  BOOL sv1016_hidden;
} SERVER_INFO_1016, *PSERVER_INFO_1016, *LPSERVER_INFO_1016;
```

Members

`sv1016_hidden`

Specifies whether the server is visible to other computers in the same network domain. This member can be one of the following values.

VALUE	MEANING
SV_VISIBLE	The server is visible.
SV_HIDDEN	The server is not visible.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1017 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1017** structure contains the network announce rate associated with the specified server.

Syntax

```
typedef struct _SERVER_INFO_1017 {
  DWORD sv1017_announce;
} SERVER_INFO_1017, *PSERVER_INFO_1017, *LPSERVER_INFO_1017;
```

Members

`sv1017_announce`

Specifies the network announce rate, in seconds. This rate determines how often the server is announced to other computers on the network.

For more information about how much the announce rate can vary from the period of time specified by this member, see SERVER_INFO_1018.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

SERVER_INFO_1018

Server Functions

SERVER_INFO_1018 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1018** structure contains information about how much the announce rate can vary for the specified server.

Syntax

```
typedef struct _SERVER_INFO_1018 {
  DWORD sv1018_anndelta;
} SERVER_INFO_1018, *PSERVER_INFO_1018, *LPSERVER_INFO_1018;
```

Members

`sv1018_anndelta`

Specifies the delta value for the announce rate, in milliseconds. This value specifies how much the announce rate can vary from the period of time specified in the sv*X*_announce member.

The delta value allows randomly varied announce rates. For example, if the sv*X*_announce member has the value 10 and the sv*X*_anndelta member has the value 1, the announce rate can vary from 9.999 seconds to 10.001 seconds. For more information, see SERVER_INFO_102 and SERVER_INFO_1017.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

SERVER_INFO_1017

SERVER_INFO_102

Server Functions

SERVER_INFO_102 structure (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **SERVER_INFO_102** structure contains information about the specified server, including name, platform, type of server, attributes, and associated software.

Syntax

```
typedef struct _SERVER_INFO_102 {
  DWORD sv102_platform_id;
  LMSTR sv102_name;
  DWORD sv102_version_major;
  DWORD sv102_version_minor;
  DWORD sv102_type;
  LMSTR sv102_comment;
  DWORD sv102_users;
  LONG  sv102_disc;
  BOOL  sv102_hidden;
  DWORD sv102_announce;
  DWORD sv102_anndelta;
  DWORD sv102_licenses;
  LMSTR sv102_userpath;
} SERVER_INFO_102, *PSERVER_INFO_102, *LPSERVER_INFO_102;
```

Members

`sv102_platform_id`

Type: **DWORD**

The information level to use for platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.

PLATFORM_ID_VMS 700	The VMS platform.

`sv102_name`

Type: **LPWSTR**

A pointer to a Unicode string specifying the name of a server.

`sv102_version_major`

Type: **DWORD**

The major version number and the server type.

The major release version number of the operating system is specified in the least significant 4 bits. The server type is specified in the most significant 4 bits. The **MAJOR_VERSION_MASK** bitmask defined in the *Lmserver.h* header should be used by an application to obtain the major version number from this member.

`sv102_version_minor`

Type: **DWORD**

The minor release version number of the operating system.

`sv102_type`

Type: **DWORD**

The type of software the computer is running.

Possible values for this member are listed in the *Lmserver.h* header file. This member can be a combination of some of the following values.

VALUE	MEANING
SV_TYPE_WORKSTATION 0x00000001	A workstation.
SV_TYPE_SERVER 0x00000002	A server.
SV_TYPE_SQLSERVER 0x00000004	A server running with Microsoft SQL Server.
SV_TYPE_DOMAIN_CTRL 0x00000008	A primary domain controller.
SV_TYPE_DOMAIN_BAKCTRL 0x00000010	A backup domain controller.

SV_TYPE_TIME_SOURCE 0x00000020	A server running the Timesource service.
SV_TYPE_AFP 0x00000040	A server running the Apple Filing Protocol (AFP) file service.
SV_TYPE_NOVELL 0x00000080	A Novell server.
SV_TYPE_DOMAIN_MEMBER 0x00000100	A LAN Manager 2.x domain member.
SV_TYPE_PRINTQ_SERVER 0x00000200	A server that shares a print queue.
SV_TYPE_DIALIN_SERVER 0x00000400	A server that runs a dial-in service.
SV_TYPE_XENIX_SERVER 0x00000800	A Xenix or Unix server.
SV_TYPE_NT 0x00001000	A workstation or server.
SV_TYPE_WFW 0x00002000	A computer that runs Windows for Workgroups.
SV_TYPE_SERVER_MFPN 0x00004000	A server that runs the Microsoft File and Print for NetWare service.
SV_TYPE_SERVER_NT 0x00008000	Any server that is not a domain controller.
SV_TYPE_POTENTIAL_BROWSER 0x00010000	A computer that can run the browser service.
SV_TYPE_BACKUP_BROWSER 0x00020000	A server running a browser service as backup.

SV_TYPE_MASTER_BROWSER 0x00040000	A server running the master browser service.
SV_TYPE_DOMAIN_MASTER 0x00080000	A server running the domain master browser.
SV_TYPE_SERVER_OSF 0x00100000	A computer that runs OSF.
SV_TYPE_SERVER_VMS 0x00200000	A computer that runs VMS.
SV_TYPE_WINDOWS 0x00400000	A computer that runs Windows.
SV_TYPE_DFS 0x00800000	A server that is the root of a DFS tree.
SV_TYPE_CLUSTER_NT 0x01000000	A server cluster available in the domain.
SV_TYPE_TERMINALSERVER 0x02000000	A server that runs the Terminal Server service.
SV_TYPE_CLUSTER_VS_NT 0x04000000	Cluster virtual servers available in the domain. **Windows 2000:** This value is not supported.
SV_TYPE_DCE 0x10000000	A server that runs the DCE Directory and Security Services or equivalent.
SV_TYPE_ALTERNATE_XPORT 0x20000000	A server that is returned by an alternate transport.
SV_TYPE_LOCAL_LIST_ONLY 0x40000000	A server that is maintained by the browser.
SV_TYPE_DOMAIN_ENUM 0x80000000	A primary domain.

The **SV_TYPE_ALL** constant is defined to 0xFFFFFFFF in the *Lmserver.h* header file. This constant can be used to check for all server types when used with the NetServerEnumfunction.

`sv102_comment`

Type: **LPWSTR**

A pointer to a Unicode string specifying a comment describing the server. The comment can be null.

`sv102_users`

Type: **DWORD**

The number of users who can attempt to log on to the system server. Note that it is the license server that determines how many of these users can actually log on.

`sv102_disc`

Type: **LONG**

The auto-disconnect time, in minutes. A session is disconnected if it is idle longer than the period of time specified by the **sv102_disc** member. If the value of **sv102_disc** is SV_NODISC, auto-disconnect is not enabled.

`sv102_hidden`

Type: **BOOL**

A value that indicates whether the server is visible to other computers in the same network domain. This member can be one of the following values defined in the *Lmserver.h* header file.

VALUE	MEANING
SV_VISIBLE	The server is visible.
SV_HIDDEN	The server is not visible.

`sv102_announce`

Type: **DWORD**

The network announce rate, in seconds. This rate determines how often the server is announced to other computers on the network. For more information about how much the announce rate can vary from the period of time specified by this member, see SERVER_INFO_1018.

`sv102_anndelta`

Type: **DWORD**

The delta value for the announce rate, in milliseconds. This value specifies how much the announce rate can vary from the period of time specified in the **sv102_announce** member.

The delta value allows randomly varied announce rates. For example, if the **sv102_announce** member has the value 10 and the **sv102_anndelta** member has the value 1, the announce rate can vary from 9.999 seconds to 10.001 seconds.

`sv102_licenses`

Type: **DWORD**

The number of users per license. By default, this number is SV_USERS_PER_LICENSE.

`sv102_userpath`

Type: **LPWSTR**

A pointer to a Unicode string specifying the path to user directories.

Remarks

To retrieve a value that indicates whether a share is the root volume in a Dfs tree structure, you must call the NetShareGetInfo function and specify information level 1005.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

NetServerSetInfo

NetShareGetInfo

Network Management Overview

Network Management Structures

SERVER_INFO_1018

Server Functions

SERVER_INFO_1107 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1107** structure specifies the number of users that can simultaneously log on to the specified server.

Syntax

```
typedef struct _SERVER_INFO_1107 {
  DWORD sv1107_users;
} SERVER_INFO_1107, *PSERVER_INFO_1107, *LPSERVER_INFO_1107;
```

Members

`sv1107_users`

Specifies the number of users who can attempt to log on to the system server. Note that it is the license server that determines how many of these users can actually log on.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1501 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1501** structure specifies the number of files that can be open in one session on the specified server.

Syntax

```
typedef struct _SERVER_INFO_1501 {
  DWORD sv1501_sessopens;
} SERVER_INFO_1501, *PSERVER_INFO_1501, *LPSERVER_INFO_1501;
```

Members

`sv1501_sessopens`

Specifies the number of files that one session can open.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1502 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1502** structure specifies the maximum number of virtual circuits per client for the specified server.

Syntax

```
typedef struct _SERVER_INFO_1502 {
  DWORD sv1502_sessvcs;
} SERVER_INFO_1502, *PSERVER_INFO_1502, *LPSERVER_INFO_1502;
```

Members

`sv1502_sessvcs`

Specifies the maximum number of virtual circuits permitted per client.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1503 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1503** structure specifies the number of search operations that can be carried out simultaneously.

Syntax

```
typedef struct _SERVER_INFO_1503 {
  DWORD sv1503_opensearch;
} SERVER_INFO_1503, *PSERVER_INFO_1503, *LPSERVER_INFO_1503;
```

Members

`sv1503_opensearch`

Specifies the number of search operations that can be carried out simultaneously.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1506 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1506** structure contains information about the maximum number of work items the specified server can allocate.

Syntax

```
typedef struct _SERVER_INFO_1506 {
  DWORD sv1506_maxworkitems;
} SERVER_INFO_1506, *PSERVER_INFO_1506, *LPSERVER_INFO_1506;
```

Members

`sv1506_maxworkitems`

Specifies the maximum number of receive buffers, or work items, the server can allocate. If this limit is reached, the transport protocol must initiate flow control at a significant cost to performance.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1509 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1509** structure specifies the maximum raw mode buffer size.

Syntax

```
typedef struct _SERVER_INFO_1509 {
  DWORD sv1509_maxrawbuflen;
} SERVER_INFO_1509, *PSERVER_INFO_1509, *LPSERVER_INFO_1509;
```

Members

`sv1509_maxrawbuflen`

Specifies the maximum raw mode buffer size, in bytes.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1510 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1510** structure specifies the maximum number of users that can be logged on to the specified server using a single virtual circuit.

Syntax

```
typedef struct _SERVER_INFO_1510 {
  DWORD sv1510_sessusers;
} SERVER_INFO_1510, *PSERVER_INFO_1510, *LPSERVER_INFO_1510;
```

Members

`sv1510_sessusers`

Specifies the maximum number of users that can be logged on to a server using a single virtual circuit.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1511 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1511** structure specifies the maximum number of tree connections that users can make with a single virtual circuit.

Syntax

```
typedef struct _SERVER_INFO_1511 {
  DWORD sv1511_sessconns;
} SERVER_INFO_1511, *PSERVER_INFO_1511, *LPSERVER_INFO_1511;
```

Members

`sv1511_sessconns`

Specifies the maximum number of tree connections that users can make with a single virtual circuit.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1512 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1512** structure contains the maximum size of nonpaged memory that the specified server can allocate at a particular time.

Syntax

```
typedef struct _SERVER_INFO_1512 {
  DWORD sv1512_maxnonpagedmemoryusage;
} SERVER_INFO_1512, *PSERVER_INFO_1512, *LPSERVER_INFO_1512;
```

Members

`sv1512_maxnonpagedmemoryusage`

Specifies the maximum size of nonpaged memory that the server can allocate at any one time. Adjust this member if you want to administer memory quota control.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1513 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1513** structure contains the maximum size of pageable memory that the specified server can allocate at a particular time.

Syntax

```
typedef struct _SERVER_INFO_1513 {
  DWORD sv1513_maxpagedmemoryusage;
} SERVER_INFO_1513, *PSERVER_INFO_1513, *LPSERVER_INFO_1513;
```

Members

`sv1513_maxpagedmemoryusage`

Specifies the maximum size of pageable memory that the server allocates at any particular time. Adjust this member if you want to administer memory quota control.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1515 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1515** structure specifies whether the server should force a client to disconnect once the client's logon time has expired.

Syntax

```
typedef struct _SERVER_INFO_1515 {
  BOOL sv1515_enableforcedlogoff;
} SERVER_INFO_1515, *PSERVER_INFO_1515, *LPSERVER_INFO_1515;
```

Members

`sv1515_enableforcedlogoff`

Specifies whether the server should force a client to disconnect, even if the client has open files, once the client's logon time has expired.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1516 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1516** structure specifies whether the server is a reliable time source.

Syntax

```
typedef struct _SERVER_INFO_1516 {
  BOOL sv1516_timesource;
} SERVER_INFO_1516, *PSERVER_INFO_1516, *LPSERVER_INFO_1516;
```

Members

`sv1516_timesource`

Specifies whether the server is a reliable time source.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1518 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1518** structure specifies whether the server is visible to LAN Manager 2.*x* clients.

Syntax

```
typedef struct _SERVER_INFO_1518 {
  BOOL sv1518_lmannounce;
} SERVER_INFO_1518, *PSERVER_INFO_1518, *LPSERVER_INFO_1518;
```

Members

`sv1518_lmannounce`

Specifies whether the server is visible to LAN Manager 2.*x* clients.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1523 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1523** structure specifies the length of time the server retains information about incomplete search operations.

Syntax

```
typedef struct _SERVER_INFO_1523 {
  DWORD sv1523_maxkeepsearch;
} SERVER_INFO_1523, *PSERVER_INFO_1523, *LPSERVER_INFO_1523;
```

Members

`sv1523_maxkeepsearch`

Specifies the length of time the server retains information about incomplete search operations.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1528 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1528** structure specifies the period of time that the scavenger remains idle before waking up to service requests.

Syntax

```
typedef struct _SERVER_INFO_1528 {
  DWORD sv1528_scavtimeout;
} SERVER_INFO_1528, *PSERVER_INFO_1528, *LPSERVER_INFO_1528;
```

Members

`sv1528_scavtimeout`

Specifies the period of time, in seconds, that the scavenger remains idle before waking up to service requests. A smaller value for this member improves the response of the server to various events but costs CPU cycles.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1529 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1529** structure specifies the minimum number of free receive work items the server requires before it begins allocating more items.

Syntax

```
typedef struct _SERVER_INFO_1529 {
  DWORD sv1529_minrcvqueue;
} SERVER_INFO_1529, *PSERVER_INFO_1529, *LPSERVER_INFO_1529;
```

Members

`sv1529_minrcvqueue`

Specifies the minimum number of free receive work items the server requires before it begins allocating more. A larger value for this member helps ensure that there will always be work items available, but a value that is too large is inefficient.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1530 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1530** structure specifies the minimum number of available receive work items the server requires to begin processing a server message block.

Syntax

```
typedef struct _SERVER_INFO_1530 {
  DWORD sv1530_minfreeworkitems;
} SERVER_INFO_1530, *PSERVER_INFO_1530, *LPSERVER_INFO_1530;
```

Members

`sv1530_minfreeworkitems`

Specifies the minimum number of available receive work items that the server requires to begin processing a server message block. A larger value for this member ensures that work items are available more frequently for nonblocking requests, but it also increases the likelihood that blocking requests will be rejected.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1533 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1533** structure specifies the maximum number of outstanding requests a client can send to the server.

Syntax

```
typedef struct _SERVER_INFO_1533 {
  DWORD sv1533_maxmpxct;
} SERVER_INFO_1533, *PSERVER_INFO_1533, *LPSERVER_INFO_1533;
```

Members

`sv1533_maxmpxct`

Specifies the maximum number of outstanding requests any one client can send to the server. For example, 10 means you can have 10 unanswered requests at the server. When any single client has 10 requests queued within the server, the client must wait for a server response before sending another request.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1536 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1536** structure specifies whether the server allows clients to use opportunistic locks (oplocks) on files.

Syntax

```
typedef struct _SERVER_INFO_1536 {
  BOOL sv1536_enableoplocks;
} SERVER_INFO_1536, *PSERVER_INFO_1536, *LPSERVER_INFO_1536;
```

Members

`sv1536_enableoplocks`

Specifies whether the server allows clients to use oplocks on files. Opportunistic locks are a significant performance enhancement, but have the potential to cause lost cached data on some networks, particularly wide-area networks.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1538 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1538** structure specifies whether several MS-DOS File Control Blocks (FCBs) are placed in a single location.

Syntax

```
typedef struct _SERVER_INFO_1538 {
  BOOL sv1538_enablefcbopens;
} SERVER_INFO_1538, *PSERVER_INFO_1538, *LPSERVER_INFO_1538;
```

Members

`sv1538_enablefcbopens`

Specifies whether several MS-DOS File Control Blocks (FCBs) are placed in a single location accessible to the server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1539 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1539** structure specifies whether the server processes raw Server Message Blocks (SMBs).

Syntax

```
typedef struct _SERVER_INFO_1539 {
  BOOL sv1539_enableraw;
} SERVER_INFO_1539, *PSERVER_INFO_1539, *LPSERVER_INFO_1539;
```

Members

`sv1539_enableraw`

Specifies whether the server processes raw SMBs. If enabled, this member allows more data to be transferred per transaction and improves performance. However, it is possible that processing raw SMBs can impede performance on certain networks. The server maintains the value of this member.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1540 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1540** structure specifies whether the server allows redirected server drives to be shared.

Syntax

```
typedef struct _SERVER_INFO_1540 {
  BOOL sv1540_enablesharednetdrives;
} SERVER_INFO_1540, *PSERVER_INFO_1540, *LPSERVER_INFO_1540;
```

Members

`sv1540_enablesharednetdrives`

Specifies whether the server allows redirected server drives to be shared.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1541 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1541** structure specifies the minimum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.

Syntax

```
typedef struct _SERVER_INFO_1541 {
  BOOL sv1541_minfreeconnections;
} SERVER_INFO_1541, *PSERVER_INFO_1541, *LPSERVER_INFO_1541;
```

Members

`sv1541_minfreeconnections`

Specifies the minimum number of free connection blocks maintained per endpoint. The server sets these aside to handle bursts of requests by clients to connect to the server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1542 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1542** structure specifies the maximum number of free connection blocks the server sets aside to handle bursts of requests by clients to connect to the server.

Syntax

```
typedef struct _SERVER_INFO_1542 {
  BOOL sv1542_maxfreeconnections;
} SERVER_INFO_1542, *PSERVER_INFO_1542, *LPSERVER_INFO_1542;
```

Members

`sv1542_maxfreeconnections`

Specifies the maximum number of free connection blocks maintained per endpoint. The server sets these aside to handle bursts of requests by clients to connect to the server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1544 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1544** structure specifies the initial number of tree connections to be allocated in the connection table.

Syntax

```
typedef struct _SERVER_INFO_1544 {
  DWORD sv1544_initconntable;
} SERVER_INFO_1544, *PSERVER_INFO_1544, *LPSERVER_INFO_1544;
```

Members

`sv1544_initconntable`

Specifies the initial number of tree connections to be allocated in the connection table. The server automatically increases the table as necessary, so setting the member to a higher value is an optimization.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1550 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1550** structure specifies the percentage of free disk space remaining before an alert message is sent.

Syntax

```
typedef struct _SERVER_INFO_1550 {
  DWORD sv1550_diskspacethreshold;
} SERVER_INFO_1550, *PSERVER_INFO_1550, *LPSERVER_INFO_1550;
```

Members

`sv1550_diskspacethreshold`

Specifies the percentage of free disk space remaining before an alert message is sent.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_1552 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_1552** structure specifies the maximum time allowed for a link delay.

Syntax

```
typedef struct _SERVER_INFO_1552 {
  DWORD sv1552_maxlinkdelay;
} SERVER_INFO_1552, *PSERVER_INFO_1552, *LPSERVER_INFO_1552;
```

Members

`sv1552_maxlinkdelay`

Specifies the maximum time allowed for a link delay, in seconds. If delays exceed this number, the server disables raw I/O for this connection.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_402 structure (lmserver.h)

2/1/2021 • 3 minutes to read • Edit Online

The **SERVER_INFO_402** structure contains information about a specified server.

Syntax

```
typedef struct _SERVER_INFO_402 {
  DWORD sv402_ulist_mtime;
  DWORD sv402_glist_mtime;
  DWORD sv402_alist_mtime;
  LMSTR sv402_alerts;
  DWORD sv402_security;
  DWORD sv402_numadmin;
  DWORD sv402_lanmask;
  LMSTR sv402_guestacct;
  DWORD sv402_chdevs;
  DWORD sv402_chdevq;
  DWORD sv402_chdevjobs;
  DWORD sv402_connections;
  DWORD sv402_shares;
  DWORD sv402_openfiles;
  DWORD sv402_sessopens;
  DWORD sv402_sessvcs;
  DWORD sv402_sessreqs;
  DWORD sv402_opensearch;
  DWORD sv402_activelocks;
  DWORD sv402_numreqbuf;
  DWORD sv402_sizreqbuf;
  DWORD sv402_numbigbuf;
  DWORD sv402_numfiletasks;
  DWORD sv402_alertsched;
  DWORD sv402_erroralert;
  DWORD sv402_logonalert;
  DWORD sv402_accessalert;
  DWORD sv402_diskalert;
  DWORD sv402_netioalert;
  DWORD sv402_maxauditsz;
  LMSTR sv402_srvheuristics;
} SERVER_INFO_402, *PSERVER_INFO_402, *LPSERVER_INFO_402;
```

Members

`sv402_ulist_mtime`

Type: **DWORD**

The last time the user list was modified. The value is expressed as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv402_glist_mtime`

Type: **DWORD**

The last time the group list was modified. The value is expressed as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv402_alist_mtime`

Type: **DWORD**

The last time the access control list was modified. The value is expressed as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv402_alerts`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the list of user names on the server. Spaces separate the names.

`sv402_security`

Type: **DWORD**

The security type of the server. This member can be one of the following values. Note that Windows NT, Windows 2000, Windows XP, and Windows Server 2003 operating systems do not support share-level security.

VALUE	MEANING
SV_SHARESECURITY	Share-level security
SV_USERSECURITY	User-level security

`sv402_numadmin`

Type: **DWORD**

The number of administrators the server can accommodate at one time.

`sv402_lanmask`

Type: **DWORD**

The order in which the network device drivers are served.

`sv402_guestacct`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of a reserved account for guest users on the server. The constant UNLEN specifies the maximum number of characters in the string.

`sv402_chdevs`

Type: **DWORD**

The number of character-oriented devices that can be shared on the server.

`sv402_chdevq`

Type: **DWORD**

The number of character-oriented device queues that can coexist on the server.

`sv402_chdevjobs`

Type: **DWORD**

The number of character-oriented device jobs that can be pending at one time on the server.

`sv402_connections`

Type: **DWORD**

The number of connections allowed on the server.

`sv402_shares`

Type: **DWORD**

The number of share names the server can accommodate.

`sv402_openfiles`

Type: **DWORD**

The number of files that can be open at once on the server.

`sv402_sessopens`

Type: **DWORD**

The number of files that one session can open.

`sv402_sessvcs`

Type: **DWORD**

The maximum number of virtual circuits permitted per client.

`sv402_sessreqs`

Type: **DWORD**

The number of simultaneous requests a client can make on a single virtual circuit.

`sv402_opensearch`

Type: **DWORD**

The number of search operations that can be carried out simultaneously.

`sv402_activelocks`

Type: **DWORD**

The number of file locks that can be active at the same time.

`sv402_numreqbuf`

Type: **DWORD**

The number of server buffers provided.

`sv402_sizreqbuf`

Type: **DWORD**

The size, in bytes, of each server buffer.

`sv402_numbigbuf`

Type: **DWORD**

The number of 64K server buffers provided.

`sv402_numfiletasks`

Type: **DWORD**

The number of processes that can access the operating system at one time.

`sv402_alertsched`

Type: **DWORD**

The interval, in seconds, for notifying an administrator of a network event.

`sv402_erroralert`

Type: **DWORD**

The number of entries that can be written to the error log, in any one interval, before notifying an administrator. The interval is specified by the **sv402_alertsched** member.

`sv402_logonalert`

Type: **DWORD**

The number of invalid logon attempts to allow a user before notifying an administrator.

`sv402_accessalert`

Type: **DWORD**

The number of invalid file access attempts to allow before notifying an administrator.

`sv402_diskalert`

Type: **DWORD**

The point at which the system sends a message notifying an administrator that free space on a disk is low. This value is expressed as the number of kilobytes of free disk space remaining on the disk.

`sv402_netioalert`

Type: **DWORD**

The network I/O error ratio, in tenths of a percent, that is allowed before notifying an administrator.

`sv402_maxauditsz`

Type: **DWORD**

The maximum size, in kilobytes, of the audit file. The audit file traces user activity.

`sv402_srvheuristics`

Type: **LPWSTR**

A pointer to a Unicode string containing flags that control operations on a server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

NetServerSetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_403 structure (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **SERVER_INFO_403** structure contains information about a specified server.

Syntax

```
typedef struct _SERVER_INFO_403 {
  DWORD sv403_ulist_mtime;
  DWORD sv403_glist_mtime;
  DWORD sv403_alist_mtime;
  LMSTR sv403_alerts;
  DWORD sv403_security;
  DWORD sv403_numadmin;
  DWORD sv403_lanmask;
  LMSTR sv403_guestacct;
  DWORD sv403_chdevs;
  DWORD sv403_chdevq;
  DWORD sv403_chdevjobs;
  DWORD sv403_connections;
  DWORD sv403_shares;
  DWORD sv403_openfiles;
  DWORD sv403_sessopens;
  DWORD sv403_sessvcs;
  DWORD sv403_sessreqs;
  DWORD sv403_opensearch;
  DWORD sv403_activelocks;
  DWORD sv403_numreqbuf;
  DWORD sv403_sizreqbuf;
  DWORD sv403_numbigbuf;
  DWORD sv403_numfiletasks;
  DWORD sv403_alertsched;
  DWORD sv403_erroralert;
  DWORD sv403_logonalert;
  DWORD sv403_accessalert;
  DWORD sv403_diskalert;
  DWORD sv403_netioalert;
  DWORD sv403_maxauditsz;
  LMSTR sv403_srvheuristics;
  DWORD sv403_auditedevents;
  DWORD sv403_autoprofile;
  LMSTR sv403_autopath;
} SERVER_INFO_403, *PSERVER_INFO_403, *LPSERVER_INFO_403;
```

Members

`sv403_ulist_mtime`

Type: **DWORD**

The last time the user list was modified. The value is expressed as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv403_glist_mtime`

Type: **DWORD**

The last time the group list was modified. The value is expressed as the number of seconds that have elapsed

since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv403_alist_mtime`

Type: **DWORD**

The last time the access control list was modified. The value is expressed as the number of seconds that have elapsed since 00:00:00, January 1, 1970, GMT, and applies to servers running with user-level security.

`sv403_alerts`

Type: **LMSTR**

A pointer to a string that specifies the list of user names on the server. Spaces separate the names.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`sv403_security`

Type: **DWORD**

The security type of the server. This member can be one of the following values. Note that Windows NT, Windows 2000, Windows XP, and Windows Server 2003 operating systems do not support share-level security.

VALUE	MEANING
SV_SHARESECURITY	Share-level security
SV_USERSECURITY	User-level security

`sv403_numadmin`

Type: **DWORD**

The number of administrators the server can accommodate at one time.

`sv403_lanmask`

Type: **DWORD**

The order in which the network device drivers are served.

`sv403_guestacct`

Type: **LPWSTR**

A pointer to a Unicode string that specifies the name of a reserved account for guest users on the server. The UNLEN constant specifies the maximum number of characters in the string.

`sv403_chdevs`

Type: **DWORD**

The number of character devices that can be shared on the server.

`sv403_chdevq`

Type: **DWORD**

The number of character device queues that can coexist on the server.

`sv403_chdevjobs`

Type: **DWORD**

The number of character device jobs that can be pending at one time on the server.

`sv403_connections`

Type: **DWORD**

The number of connections allowed on the server.

`sv403_shares`

Type: **DWORD**

The number of share names the server can accommodate.

`sv403_openfiles`

Type: **DWORD**

The number of files that can be open at once on the server.

`sv403_sessopens`

Type: **DWORD**

The number of files that one session can open.

`sv403_sessvcs`

Type: **DWORD**

The maximum number of virtual circuits permitted per client.

`sv403_sessreqs`

Type: **DWORD**

The number of simultaneous requests a client can make on a single virtual circuit.

`sv403_opensearch`

Type: **DWORD**

The number of search operations that can be carried out simultaneously.

`sv403_activelocks`

Type: **DWORD**

The number of file locks that can be active at the same time.

`sv403_numreqbuf`

Type: **DWORD**

The number of server buffers that are provided.

`sv403_sizreqbuf`

Type: **DWORD**

The size, in bytes, of each server buffer.

`sv403_numbigbuf`

Type: **DWORD**

The number of 64K server buffers provided.

`sv403_numfiletasks`

Type: **DWORD**

The number of processes that can access the operating system at the same time.

`sv403_alertsched`

Type: **DWORD**

The alert interval, in seconds, for notifying an administrator of a network event.

`sv403_erroralert`

Type: **DWORD**

The number of entries that can be written to the error log, in any one interval, before notifying an administrator. The interval is specified by the **sv403_alertsched** member.

`sv403_logonalert`

Type: **DWORD**

The number of invalid attempts that a user tries to logon before notifying an administrator.

`sv403_accessalert`

Type: **DWORD**

The number of invalid file access attempts to allow before notifying an administrator.

`sv403_diskalert`

Type: **DWORD**

The amount of free disk space at which the system sends a message notifying an administrator that free space on a disk is low. This value is expressed as the number of kilobytes of free disk space remaining on the disk.

`sv403_netioalert`

Type: **DWORD**

The network I/O error ratio, in tenths of a percent, that is allowed before notifying an administrator.

`sv403_maxauditsz`

Type: **DWORD**

The maximum audit file size in kilobytes. The audit file traces user activity.

`sv403_srvheuristics`

Type: **LPWSTR**

A pointer to a Unicode string that contains flags that are used to control operations on a server.

`sv403_auditedevents`

Type: **DWORD**

The audit event control mask.

`sv403_autoprofile`

Type: **DWORD**

A value that controls the action of the server on the profile. The following values are predefined.

VALUE	MEANING
SW_AUTOPROF_LOAD_MASK	The server loads the profile.
SW_AUTOPROF_SAVE_MASK	The server saves the profile.

`sv403_autopath`

Type: **LPWSTR**

A pointer to a Unicode string that contains the path for the profile.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

NetServerSetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_502 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_INFO_502** structure is obsolete. The structure contains information about a specified server.

Syntax

```
typedef struct _SERVER_INFO_502 {
  DWORD sv502_sessopens;
  DWORD sv502_sessvcs;
  DWORD sv502_opensearch;
  DWORD sv502_sizreqbuf;
  DWORD sv502_initworkitems;
  DWORD sv502_maxworkitems;
  DWORD sv502_rawworkitems;
  DWORD sv502_irpstacksize;
  DWORD sv502_maxrawbuflen;
  DWORD sv502_sessusers;
  DWORD sv502_sessconns;
  DWORD sv502_maxpagedmemoryusage;
  DWORD sv502_maxnonpagedmemoryusage;
  BOOL  sv502_enablesoftcompat;
  BOOL  sv502_enableforcedlogoff;
  BOOL  sv502_timesource;
  BOOL  sv502_acceptdownlevelapis;
  BOOL  sv502_lmannounce;
} SERVER_INFO_502, *PSERVER_INFO_502, *LPSERVER_INFO_502;
```

Members

`sv502_sessopens`

Type: **DWORD**

The number of files that can be open in one session.

`sv502_sessvcs`

Type: **DWORD**

T he maximum number of virtual circuits permitted per client.

`sv502_opensearch`

Type: **DWORD**

The number of search operations that can be carried out simultaneously.

`sv502_sizreqbuf`

Type: **DWORD**

The size, in bytes, of each server buffer.

`sv502_initworkitems`

Type: **DWORD**

The initial number of receive buffers, or work items, used by the server.

`sv502_maxworkitems`

Type: **DWORD**

The maximum number of receive buffers, or work items, the server can allocate. If this limit is reached, the transport must initiate flow control at a significant performance cost.

`sv502_rawworkitems`

Type: **DWORD**

The number of special work items the server uses for raw mode I/O. A large value for this member can increase performance, but it requires more memory.

`sv502_irpstacksize`

Type: **DWORD**

The number of stack locations that the server allocated in I/O request packets (IRPs).

`sv502_maxrawbuflen`

Type: **DWORD**

The maximum raw mode buffer size, in bytes.

`sv502_sessusers`

Type: **DWORD**

The maximum number of users that can be logged on to the server using a single virtual circuit.

`sv502_sessconns`

Type: **DWORD**

The maximum number of tree connections that can be made on the server using a single virtual circuit.

`sv502_maxpagedmemoryusage`

Type: **DWORD**

The maximum size, in bytes, of pageable memory that the server can allocate at any one time.

`sv502_maxnonpagedmemoryusage`

Type: **DWORD**

The maximum size, in bytes, of nonpaged memory that the server can allocate at any one time.

`sv502_enablesoftcompat`

Type: **BOOL**

A value that indicates whether the server maps a request to a normal open request with shared-read access when the server receives a compatibility open request with read access. Mapping such requests allows several MS-DOS computers to open a single file for read access.

`sv502_enableforcedlogoff`

Type: **BOOL**

A value that indicates whether the server should force a client to disconnect, even if the client has open files,

once the client's logon time has expired.

`sv502_timesource`

Type: **BOOL**

A value that indicates whether the server is a reliable time source.

`sv502_acceptdownlevelapis`

Type: **BOOL**

A value that indicates whether the server accepts function calls from previous-generation LAN Manager clients.

`sv502_lmannounce`

Type: **BOOL**

A value that indicates whether the server is visible to LAN Manager 2.x clients.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

NetServerSetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_INFO_503 structure (lmserver.h)

2/1/2021 • 5 minutes to read • Edit Online

The **SERVER_INFO_503** structure is obsolete. The structure contains information about the specified server.

Syntax

```
typedef struct _SERVER_INFO_503 {
  DWORD sv503_sessopens;
  DWORD sv503_sessvcs;
  DWORD sv503_opensearch;
  DWORD sv503_sizreqbuf;
  DWORD sv503_initworkitems;
  DWORD sv503_maxworkitems;
  DWORD sv503_rawworkitems;
  DWORD sv503_irpstacksize;
  DWORD sv503_maxrawbuflen;
  DWORD sv503_sessusers;
  DWORD sv503_sessconns;
  DWORD sv503_maxpagedmemoryusage;
  DWORD sv503_maxnonpagedmemoryusage;
  BOOL  sv503_enablesoftcompat;
  BOOL  sv503_enableforcedlogoff;
  BOOL  sv503_timesource;
  BOOL  sv503_acceptdownlevelapis;
  BOOL  sv503_lmannounce;
  LMSTR sv503_domain;
  DWORD sv503_maxcopyreadlen;
  DWORD sv503_maxcopywritelen;
  DWORD sv503_minkeepsearch;
  DWORD sv503_maxkeepsearch;
  DWORD sv503_minkeepcomplsearch;
  DWORD sv503_maxkeepcomplsearch;
  DWORD sv503_threadcountadd;
  DWORD sv503_numblockthreads;
  DWORD sv503_scavtimeout;
  DWORD sv503_minrcvqueue;
  DWORD sv503_minfreeworkitems;
  DWORD sv503_xactmemsize;
  DWORD sv503_threadpriority;
  DWORD sv503_maxmpxct;
  DWORD sv503_oplockbreakwait;
  DWORD sv503_oplockbreakresponsewait;
  BOOL  sv503_enableoplocks;
  BOOL  sv503_enableoplockforceclose;
  BOOL  sv503_enablefcbopens;
  BOOL  sv503_enableraw;
  BOOL  sv503_enablesharednetdrives;
  DWORD sv503_minfreeconnections;
  DWORD sv503_maxfreeconnections;
} SERVER_INFO_503, *PSERVER_INFO_503, *LPSERVER_INFO_503;
```

Members

`sv503_sessopens`

Type: **DWORD**

The number of files that can be open in one session.

`sv503_sessvcs`

Type: **DWORD**

The maximum number of sessions or virtual circuits permitted per client.

`sv503_opensearch`

Type: **DWORD**

The number of search operations that can be carried out simultaneously.

`sv503_sizreqbuf`

Type: **DWORD**

The size, in bytes, of each server buffer.

`sv503_initworkitems`

Type: **DWORD**

The initial number of receive buffers, or work items, used by the server.

`sv503_maxworkitems`

Type: **DWORD**

The maximum number of receive buffers, or work items, the server can allocate. If this limit is reached, the transport must initiate flow control at a significant performance cost.

`sv503_rawworkitems`

Type: **DWORD**

The number of special work items the server uses for raw mode I/O. A larger value for this member can increase performance but it requires more memory.

`sv503_irpstacksize`

Type: **DWORD**

The number of stack locations that the server allocated in I/O request packets (IRPs).

`sv503_maxrawbuflen`

Type: **DWORD**

The maximum raw mode buffer size, in bytes.

`sv503_sessusers`

Type: **DWORD**

The maximum number of users that can be logged on to the server using a single session or virtual circuit.

`sv503_sessconns`

Type: **DWORD**

The maximum number of tree connections that can be made on the server using a single session or virtual circuit.

`sv503_maxpagedmemoryusage`

Type: **DWORD**

The maximum size, in bytes, of pageable memory that the server can allocate at any one time.

`sv503_maxnonpagedmemoryusage`

`sv503_enablesoftcompat`

Type: **BOOL**

A value that indicates whether the server maps a request to a normal open request with shared-read access when the server receives a compatibility open request with read access. Mapping such requests allows several MS-DOS computers to open a single file for read access. This member is unused.

`sv503_enableforcedlogoff`

Type: **BOOL**

A value that indicates whether the server should force a client to disconnect, even if the client has open files, once the client's logon time has expired.

`sv503_timesource`

Type: **BOOL**

A value that indicates whether the server is a reliable time source.

`sv503_acceptdownlevelapis`

Type: **BOOL**

A value that indicates whether the server accepts function calls from previous-generation LAN Manager clients.

`sv503_lmannounce`

Type: **BOOL**

A value that indicates whether the server is visible to LAN Manager 2.x clients.

`sv503_domain`

Type: **LPWSTR**

A pointer to a Unicode character string that specifies the name of the server's domain.

`sv503_maxcopyreadlen`

Type: **DWORD**

The maximum length, in bytes, of copy reads on the server.

This member is unused.

`sv503_maxcopywritelen`

Type: **DWORD**

The maximum length, in bytes, of copy writes on the server.

This member is unused.

`sv503_minkeepsearch`

Type: **DWORD**

The minimum length of time the server retains information about incomplete search operations. This member is unused.

`sv503_maxkeepsearch`

Type: **DWORD**

The maximum length of time, in seconds, the server retains information about incomplete search operations.

`sv503_minkeepcomplsearch`

Type: **DWORD**

The minimum length of time, in seconds, the server retains information about complete search operations. This member is unused.

`sv503_maxkeepcomplsearch`

Type: **DWORD**

The maximum length of time, in seconds, the server retains information about complete search operations. This member is unused.

`sv503_threadcountadd`

Type: **DWORD**

The number of additional threads the server should use in addition to one worker thread per processor it already uses. This member is unused.

`sv503_numblockthreads`

Type: **DWORD**

The number of threads set aside by the server to service requests that can block the thread for a significant amount of time. This member is unused.

`sv503_scavtimeout`

Type: **DWORD**

The period of time, in seconds, that the scavenger remains idle before waking up to service requests.

`sv503_minrcvqueue`

Type: **DWORD**

The minimum number of free receive work items the server requires before it begins to allocate more.

`sv503_minfreeworkitems`

Type: **DWORD**

The minimum number of available receive work items that the server requires to begin processing a server message block.

`sv503_xactmemsize`

Type: **DWORD**

The size, in bytes, of the shared memory region used to process server functions.

`sv503_threadpriority`

Type: **DWORD**

The priority of all server threads in relation to the base priority of the process.

`sv503_maxmpxct`

Type: **DWORD**

The maximum number of outstanding requests that any one client can send to the server. For example, 10 means you can have 10 unanswered requests at the server. When any single client has 10 requests queued within the server, the client must wait for a server response before sending another request.

`sv503_oplockbreakwait`

Type: **DWORD**

The period of time, in seconds, to wait before timing out an opportunistic lock break request.

`sv503_oplockbreakresponsewait`

Type: **DWORD**

The period of time, in seconds, the server waits for a client to respond to an oplock break request from the server.

`sv503_enableoplocks`

Type: **BOOL**

A value that indicates whether the server allows clients to use opportunistic locks on files. Opportunistic locks are a significant performance enhancement, but have the potential to cause lost cached data on some networks, particularly wide-area networks.

`sv503_enableoplockforceclose`

Type: **BOOL**

A value that indicates how the server should behave if a client has an opportunistic lock (oplock) and does not respond to an oplock break. This member indicates whether the server will fail the second open (value of 0), or force close the open instance of a client that has an oplock (value equal to 1). This member is unused.

`sv503_enablefcbopens`

Type: **BOOL**

A value that indicates whether several MS-DOS File Control Blocks (FCBs) are placed in a single location accessible to the server. If enabled, this can save resources on the server.

`sv503_enableraw`

Type: **BOOL**

A value that indicates whether the server processes raw Server Message Blocks (SMBs). If enabled, this allows more data to transfer per transaction and also improves performance. However, it is possible that processing raw SMBs can impede performance on certain networks. The server maintains the value of this member.

`sv503_enablesharednetdrives`

Type: **BOOL**

A value that indicates whether the server allows redirected server drives to be shared.

`sv503_minfreeconnections`

Type: **DWORD**

The minimum number of free connection blocks maintained per endpoint. The server sets these aside to handle bursts of requests by clients to connect to the server.

`sv503_maxfreeconnections`

Type: **DWORD**

The maximum number of free connection blocks maintained per endpoint. The server sets these aside to handle bursts of requests by clients to connect to the server.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerGetInfo

Network Management Overview

Network Management Structures

Server Functions

SERVER_TRANSPORT_INFO_0 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_TRANSPORT_INFO_0** structure contains information about the specified transport protocol, including name, address, and location on the network.

Syntax

```
typedef struct _SERVER_TRANSPORT_INFO_0 {
  DWORD  svti0_numberofvcs;
  LMSTR  svti0_transportname;
  LPBYTE svti0_transportaddress;
  DWORD  svti0_transportaddresslength;
  LMSTR  svti0_networkaddress;
} SERVER_TRANSPORT_INFO_0, *PSERVER_TRANSPORT_INFO_0, *LPSERVER_TRANSPORT_INFO_0;
```

Members

`svti0_numberofvcs`

Type: **DWORD**

The number of clients connected to the server that are using the transport protocol specified by the **svti0_transportname** member.

`svti0_transportname`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of a transport device; for example,

```
\Device\NetBT_Tcpip_{2C9725F4-151A-11D3-AEEC-C3B211BD350B}
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti0_transportaddress`

Type: **LPBYTE**

A pointer to a variable that contains the address the server is using on the transport device specified by the **svti0_transportname** member.

This member is usually the NetBIOS name that the server is using. In these instances, the name must be 16 characters long, and the last character must be a blank character (0x20).

`svti0_transportaddresslength`

Type: **DWORD**

The length, in bytes, of the **svti0_transportaddress** member. For NetBIOS names, the value of this member is 16 (decimal).

`svti0_networkaddress`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the address the network adapter is using. The string is transport-specific.

You can retrieve this value only with a call to the NetServerTransportEnum function. You cannot set this value with a call to the NetServerTransportAdd function or the NetServerTransportAddEx function.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Remarks

The **SERVER_TRANSPORT_INFO_0** structure is used by the NetServerTransportAdd or NetServerTransportAddEx function to bind the specified server to the transport protocol.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportDel

NetServerTransportEnum

Network Management Overview

Network Management Structures

SERVER_TRANSPORT_INFO_1

SERVER_TRANSPORT_INFO_2

SERVER_TRANSPORT_INFO_3

Server and Workstation Transport Functions

SERVER_TRANSPORT_INFO_1 structure (lmserver.h)

2/1/2021 • 2 minutes to read • Edit Online

The **SERVER_TRANSPORT_INFO_1** structure contains information about the specified transport protocol, including name and address. This information level is valid only for the NetServerTransportAddEx function.

Syntax

```
typedef struct _SERVER_TRANSPORT_INFO_1 {
  DWORD  svti1_numberofvcs;
  LMSTR  svti1_transportname;
  LPBYTE svti1_transportaddress;
  DWORD  svti1_transportaddresslength;
  LMSTR  svti1_networkaddress;
  LMSTR  svti1_domain;
} SERVER_TRANSPORT_INFO_1, *PSERVER_TRANSPORT_INFO_1, *LPSERVER_TRANSPORT_INFO_1;
```

Members

`svti1_numberofvcs`

Type: **DWORD**

The number of clients connected to the server that are using the transport protocol specified by the **svti1_transportname** member.

`svti1_transportname`

Type: **LMSTR**

A pointer to a null-terminated character string that contains the name of a transport device; for example,

```
\Device\NetBT_Tcpip_{2C9725F4-151A-11D3-AEEC-C3B211BD350B}
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti1_transportaddress`

Type: **LPBYTE**

A pointer to a variable that contains the address the server is using on the transport device specified by the **svti1_transportname** member.

This member is usually the NetBIOS name that the server is using. In these instances, the name must be 16 characters long, and the last character must be a blank character (0x20).

`svti1_transportaddresslength`

Type: **DWORD**

The length, in bytes, of the **svti1_transportaddress** member. For NetBIOS names, the value of this member is 16 (decimal).

`svti1_networkaddress`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the address the network adapter is using. The string is transport-specific.

You can retrieve this value only with a call to the NetServerTransportEnum function. You cannot set this value with a call to the NetServerTransportAdd function or the NetServerTransportAddEx function.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti1_domain`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of the domain to which the server should announce its presence. (When you call NetServerTransportEnum, this member is the name of the domain to which the server is announcing its presence.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Remarks

The **SERVER_TRANSPORT_INFO_1** structure is used by the NetServerTransportAddEx function to bind the specified server to the transport protocol.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerComputerNameAdd

NetServerComputerNameDel

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportDel

NetServerTransportEnum

Network Management Overview

Network Management Structures

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_2

SERVER_TRANSPORT_INFO_3

Server and Workstation Transport Functions

SERVER_TRANSPORT_INFO_2 structure (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **SERVER_TRANSPORT_INFO_2** structure contains information about the specified transport protocol, including the transport name and address. This information level is valid only for the NetServerTransportAddEx function.

Syntax

```
typedef struct _SERVER_TRANSPORT_INFO_2 {
  DWORD  svti2_numberofvcs;
  LMSTR  svti2_transportname;
  LPBYTE svti2_transportaddress;
  DWORD  svti2_transportaddresslength;
  LMSTR  svti2_networkaddress;
  LMSTR  svti2_domain;
  ULONG  svti2_flags;
} SERVER_TRANSPORT_INFO_2, *PSERVER_TRANSPORT_INFO_2, *LPSERVER_TRANSPORT_INFO_2;
```

Members

`svti2_numberofvcs`

Type: **DWORD**

The number of clients connected to the server that are using the transport protocol specified by the **svti2_transportname** member.

`svti2_transportname`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of a transport device; for example,

```
\Device\NetBT_Tcpip_{2C9725F4-151A-11D3-AEEC-C3B211BD350B}
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti2_transportaddress`

Type: **LPBYTE**

A pointer to a variable that contains the address the server is using on the transport device specified by the **svti2_transportname** member.

This member is usually the NetBIOS name that the server is using. In these instances, the name must be 16 characters long, and the last character must be a blank character (0x20).

`svti2_transportaddresslength`

Type: **DWORD**

The length, in bytes, of the **svti2_transportaddress** member. For NetBIOS names, the value of this member is 16 (decimal).

`svti2_networkaddress`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the address the network adapter is using. The string is transport-specific.

You can retrieve this value only with a call to the NetServerTransportEnum function. You cannot set this value with a call to the NetServerTransportAdd function or the NetServerTransportAddEx function.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti2_domain`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of the domain to which the server should announce its presence. (When you call NetServerTransportEnum, this member is the name of the domain to which the server is announcing its presence.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti2_flags`

Type: **ULONG**

This member can be a combination of the following bit values defined in the *Lmserver.h* header file.

VALUE	MEANING
SVTI2_REMAP_PIPE_NAMES	If this value is set for an endpoint, client requests arriving over the transport to open a named pipe are rerouted (remapped) to the following local pipe name: **$$\ServerName\PipeName** For more information on the use of this value, see the Remarks section.
SVTI2_SCOPED_NAME	If this value is set for an endpoint and there is an attempt to create a second transport with the same network address but a different transport name and conflicting settings for the SCOPED flag, this transport creation will fail. Thus, every registered transport for a given network address must have the same scoped setting. For more information on the use of this value, see the Remarks section. This value is defined on Windows Server 2008 and Windows Vista with SP1.

Remarks

The **SERVER_TRANSPORT_INFO_2** structure is used by the NetServerTransportAddEx function to bind the specified server to the transport protocol.

An example of the use of the SVTI2_REMAP_PIPE_NAMES value follows. Call the NetServerTransportAddEx function to add a transport to the server, specifying the address of "MyServer" in the **svti2_transportaddress** member, and **SVTI2_REMAP_PIPE_NAMES** in the **svti2_flags** member. When a client attempts to open "Pipe" on "\MyServer" the client will actually open $$MyServer\Pipe instead.

On Windows Server 2008 and Windows Vista with SP1, every name registered with the Windows remote file server (SRV) is designated as either a scoped name or a non-scoped name. Every share that is added to the system will then either be attached to all of the non-scoped names, or to a single scoped name. Applications that wish to use the scoping features are responsible for both registering the new name as a scoped endpoint and then creating the shares with an appropriate scope. In this way, legacy uses of the Network Management and Network Share Management functions are not affected in any way since they continue to register shares and names as non-scoped names.

A scoped endpoint is created by calling the NetServerTransportAddEx function with the *level* parameter set to 2 and the *bufptr* parameter pointed to a **SERVER_TRANSPORT_INFO_2** structure with the **SVTI2_SCOPED_NAME** bit value set in **svti2_flags** member. A scoped endpoint is also created by calling the **NetServerTransportAddEx** function with the *level* parameter set to 3 and the *bufptr* parameter pointed to a SERVER_TRANSPORT_INFO_3 structure with the **SVTI2_SCOPED_NAME** bit value set in **svti3_flags** member.

When the **SVTI2_SCOPED_NAME** bit value is set for a transport, then shares can be added with a corresponding server name (the **shi503_servername** member of the SHARE_INFO_503 structure) in a scoped fashion using the NetShareAdd function. If there is no transport registered with the **SVTI2_SCOPED_NAME** bit value and the name provided in **shi503_servername** member, then the share add in a scoped fashion will not succeed.

The NetShareAdd function is used to add a scoped share on a remote server specified in the *servername* parameter. The remote server specified in the **shi503_servername** member of the SHARE_INFO_503 passed in the *bufptr* parameter must have been bound to a transport protocol using the NetServerTransportAddEx function as a scoped endpoint. The **SVTI2_SCOPED_NAME** flag must have been specified in the **shi503_servername** member of the **SERVER_TRANSPORT_INFO_2** or SERVER_TRANSPORT_INFO_3 structure for the transport protocol. The NetShareDelEx function is used to delete a scoped share. The NetShareGetInfo and NetShareSetInfo functions are to used to get and set information on a scoped share.

Scoped endpoints are generally used by the cluster namespace.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmserver.h (include Lm.h)

See also

NetServerComputerNameAdd

NetServerComputerNameDel

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportDel

NetServerTransportEnum

NetShareAdd

NetShareDelEx

NetShareGetInfo

NetShareSetInfo

Network Management Overview

Network Management Structures

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

SERVER_TRANSPORT_INFO_3

SHARE_INFO_503

Server and Workstation Transport Functions

SERVER_TRANSPORT_INFO_3 structure (lmserver.h)

2/1/2021 • 4 minutes to read • Edit Online

The **SERVER_TRANSPORT_INFO_3** structure contains information about the specified transport protocol, including name, address and password (credentials). This information level is valid only for the NetServerTransportAddEx function.

Syntax

```
typedef struct _SERVER_TRANSPORT_INFO_3 {
  DWORD  svti3_numberofvcs;
  LMSTR  svti3_transportname;
  LPBYTE svti3_transportaddress;
  DWORD  svti3_transportaddresslength;
  LMSTR  svti3_networkaddress;
  LMSTR  svti3_domain;
  ULONG  svti3_flags;
  DWORD  svti3_passwordlength;
  BYTE   svti3_password[256];
} SERVER_TRANSPORT_INFO_3, *PSERVER_TRANSPORT_INFO_3, *LPSERVER_TRANSPORT_INFO_3;
```

Members

`svti3_numberofvcs`

Type: **DWORD**

The number of clients connected to the server that are using the transport protocol specified by the **svti3_transportname** member.

`svti3_transportname`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of a transport device; for example,

```
\Device\NetBT_Tcpip_{2C9725F4-151A-11D3-AEEC-C3B211BD350B}
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti3_transportaddress`

Type: **LPBYTE**

A pointer to a variable that contains the address the server is using on the transport device specified by the **svti3_transportname** member.

This member is usually the NetBIOS name that the server is using. In these instances, the name must be 16 characters long, and the last character must be a blank character (0x20).

`svti3_transportaddresslength`

Type: **DWORD**

The length, in bytes, of the **svti3_transportaddress** member. For NetBIOS names, the value of this member is

16 (decimal).

`svti3_networkaddress`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the address the network adapter is using. The string is transport-specific.

You can retrieve this value only with a call to the NetServerTransportEnum function. You cannot set this value with a call to the NetServerTransportAdd function or the NetServerTransportAddEx function.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti3_domain`

Type: **LMSTR**

A pointer to a NULL-terminated character string that contains the name of the domain to which the server should announce its presence. (When you call NetServerTransportEnum, this member is the name of the domain to which the server is announcing its presence.)

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`svti3_flags`

Type: **ULONG**

This member can be a combination of the following bit values defined in the *Lmserver.h* header file.

VALUE	MEANING
SVTI2_REMAP_PIPE_NAMES	If this value is set for an endpoint, client requests arriving over the transport to open a named pipe are rerouted (remapped) to the following local pipe name: **$$\ServerName\PipeName** For more information on the use of this value, see the Remarks section.
SVTI2_SCOPED_NAME	If this value is set for an endpoint and there is an attempt to create a second transport with the same network address but a different transport name and conflicting settings for the SCOPED flag, this transport creation will fail. Thus, every registered transport for a given network address must have the same scoped setting. For more information on the use of this value, see the Remarks section. This value is defined on Windows Server 2008 and Windows Vista with SP1.

`svti3_passwordlength`

Type: **DWORD**

The number of valid bytes in the **svti3_password** member.

`svti3_password`

Type: **BYTE[256]**

The credentials to use for the new transport address. If the **svti3_passwordlength** member is zero, the credentials for the server are used.

Remarks

The **SERVER_TRANSPORT_INFO_3** structure is used by the NetServerTransportAddEx function to bind the specified server to the transport protocol.

An example of the use of the SVTI2_REMAP_PIPE_NAMES value follows. Call the NetServerTransportAddEx function to add a transport to the server, specifying the address of "MyServer" in the **svti3_transportaddress** member, and **SVTI2_REMAP_PIPE_NAMES** in the **svti3_flags** member. When a client attempts to open "Pipe" on "\MyServer" the client will actually open $$MyServer\Pipe instead.

The **svti3_passwordlength** and **svti3_password** members are necessary for a client and server to perform mutual authentication.

On Windows Server 2008 and Windows Vista with SP1, every name registered with the Windows remote file server (SRV) is designated as either a scoped name or a non-scoped name. Every share that is added to the system will then either be attached to all of the non-scoped names, or to a single scoped name. Applications that wish to use the scoping features are responsible for both registering the new name as a scoped endpoint and then creating the shares with an appropriate scope. In this way, legacy uses of the Network Management and Network Share Management functions are not affected in any way since they continue to register shares and names as non-scoped names.

A scoped endpoint is created by calling the NetServerTransportAddEx function with the *level* parameter set to 2 and the *bufptr* parameter pointed to a SERVER_TRANSPORT_INFO_2 structure with the **SVTI2_SCOPED_NAME** bit value set in **svti2_flags** member. A scoped endpoint is also created by calling the **NetServerTransportAddEx** function with the *level* parameter set to 3 and the *bufptr* parameter pointed to a **SERVER_TRANSPORT_INFO_3** structure with the **SVTI2_SCOPED_NAME** bit value set in **svti3_flags** member.

When the **SVTI2_SCOPED_NAME** bit value is set for a transport, then shares can be added with a corresponding server name (the **shi503_servername** member of the SHARE_INFO_503 structure) in a scoped fashion using the NetShareAdd function. If there is no transport registered with the **SVTI2_SCOPED_NAME** bit value and the name provided in **shi503_servername** member, then the share add in a scoped fashion will not succeed.

The NetShareAdd function is used to add a scoped share on a remote server specified in the *servername* parameter. The remote server specified in the **shi503_servername** member of the SHARE_INFO_503 passed in the *bufptr* parameter must have been bound to a transport protocol using the NetServerTransportAddEx function as a scoped endpoint. The **SVTI2_SCOPED_NAME** flag must have been specified in the **shi503_servername** member of the SERVER_TRANSPORT_INFO_2 or **SERVER_TRANSPORT_INFO_3** structure for the transport protocol. The NetShareDelEx function is used to delete a scoped share. The NetShareGetInfo and NetShareSetInfo functions are to used to get and set information on a scoped share.

Scoped endpoints are generally used by the cluster namespace.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmserver.h (include Lm.h)

See also

NetServerTransportAdd

NetServerTransportAddEx

NetServerTransportDel

NetServerTransportEnum

NetShareAdd

NetShareDelEx

NetShareGetInfo

NetShareSetInfo

Network Management Overview

Network Management Structures

SERVER_TRANSPORT_INFO_0

SERVER_TRANSPORT_INFO_1

SERVER_TRANSPORT_INFO_2

SHARE_INFO_503

Server and Workstation Transport Functions

lmsvc.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmsvc.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetServiceControl	The NetServiceControl function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceEnum	The NetServiceEnum function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceGetInfo	The NetServiceGetInfo function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.
NetServiceInstall	The NetServiceInstall function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.

NetServiceControl function (lmsvc.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServiceControl** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServiceControl(
  LPCWSTR servername,
  LPCWSTR service,
  DWORD   opcode,
  DWORD   arg,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

TBD

`service`

TBD

`opcode`

TBD

`arg`

TBD

`bufptr`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmsvc.h

NetServiceEnum function (lmsvc.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServiceEnum** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServiceEnum(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

TBD

`level`

TBD

`bufptr`

TBD

`prefmaxlen`

TBD

`entriesread`

TBD

`totalentries`

TBD

`resume_handle`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmsvc.h

NetServiceGetInfo function (lmsvc.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServiceGetInfo** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServiceGetInfo(
  LPCWSTR servername,
  LPCWSTR service,
  DWORD   level,
  LPBYTE  *bufptr
);
```

Parameters

`servername`

TBD

`service`

TBD

`level`

TBD

`bufptr`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmsvc.h

NetServiceInstall function (lmsvc.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetServiceInstall** function is obsolete. It is included for compatibility with 16-bit versions of Windows. Other applications should use the service functions.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetServiceInstall(
  LPCWSTR    servername,
  LPCWSTR    service,
  DWORD      argc,
  LPCWSTR [] argv,
  LPBYTE     *bufptr
);
```

Parameters

`servername`

TBD

`service`

TBD

`argc`

TBD

`argv`

TBD

`bufptr`

TBD

Return value

None

Requirements

Target Platform	Windows
Header	lmsvc.h

lmuse.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmuse.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetUseAdd	The NetUseAdd function establishes a connection between the local computer and a remote server.
NetUseDel	The NetUseDel function ends a connection to a shared resource.
NetUseEnum	The NetUseEnum function lists all current connections between the local computer and resources on remote servers.
NetUseGetInfo	The NetUseGetInfo function retrieves information about a connection to a shared resource.

Structures

TITLE	DESCRIPTION
USE_INFO_0	The USE_INFO_0 structure contains the name of a shared resource and the local device redirected to it.
USE_INFO_1	Contains information about the connection between a local device and a shared resource.
USE_INFO_2	The USE_INFO_2 structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, and domain name.
USE_INFO_3	The USE_INFO_3 structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, domain name, and specific flags that describe connection behavior.

NetUseAdd function (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetUseAdd** function establishes a connection between the local computer and a remote server. You can specify a local drive letter or a printer device to connect. If you do not specify a local drive letter or printer device, the function authenticates the client with the server for future connections.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUseAdd(
  LPTSTR  servername,
  DWORD   LevelFlags,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

The UNC name of the computer on which to execute this function. If this parameter is **NULL**, then the local computer is used. If the *UncServerName* parameter specified is a remote computer, then the remote computer must support remote RPC calls using the legacy Remote Access Protocol mechanism.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`LevelFlags`

A value that specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
1	Specifies information about the connection between a local device and a shared resource. Information includes the connection status and type. The *Buf* parameter is a pointer to a USE_INFO_1 structure.
2	Specifies information about the connection between a local device and a shared resource. Information includes the connection status and type, and a user name and domain name. The *Buf* parameter is a pointer to a USE_INFO_2 structure.

`buf`

A pointer to the buffer that specifies the data. The format of this data depends on the value of the *Level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

A pointer to a value that receives the index of the first member of the information structure in error when the ERROR_INVALID_PARAMETER error is returned. If this parameter is **NULL**, the index is not returned on error. For more information, see the following Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

You can also use the WNetAddConnection2 and WNetAddConnection3 functions to redirect a local device to a network resource.

No special group membership is required to call the **NetUseAdd** function. This function cannot be executed on a remote server except in cases of downlevel compatibility.

This function applies only to the Server Message Block (LAN Manager Workstation) client. The **NetUseAdd** function does not support Distributed File System (DFS) shares. To add a share using a different network provider (WebDAV or a DFS share, for example), use the WNetAddConnection2 or WNetAddConnection3 function.

If the **NetUseAdd** function returns ERROR_INVALID_PARAMETER, you can use the *ParmError* parameter to indicate the first member of the information structure that is invalid. (The information structure begins with USE_INFO_ and its format is specified by the *Level* parameter.) The following table lists the values that can be returned in the *ParmError* parameter and the corresponding structure member that is in error. (The prefix ui*_ indicates that the member can begin with multiple prefixes, for example, ui1_ or ui2_.)

CONSTANT	VALUE	MEMBER
USE_LOCAL_PARMNUM	1	ui*_local
USE_REMOTE_PARMNUM	2	ui*_remote
USE_PASSWORD_PARMNUM	3	ui*_password
USE_ASGTYPE_PARMNUM	4	ui*_asg_type
USE_USERNAME_PARMNUM	5	ui*_username
USE_DOMAINNAME_PARMNUM	6	ui*_domainname

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmuse.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUseDel

Network Management Functions

Network Management Overview

USE_INFO_1

USE_INFO_2

Use Functions

WNetAddConnection2

WNetAddConnection3

NetUseDel function (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetUseDel** function ends a connection to a shared resource.

You can also use the WNetCancelConnection2 function to terminate a network connection.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUseDel(
  LMSTR UncServerName,
  LMSTR UseName,
  DWORD ForceLevelFlags
);
```

Parameters

`UncServerName`

The UNC name of the computer on which to execute this function. If this is parameter is **NULL**, then the local computer is used.

If the *UncServerName* parameter specified is a remote computer, then the remote computer must support remote RPC calls using the legacy Remote Access Protocol mechanism.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`UseName`

A pointer to a string that specifies the path of the connection to delete.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ForceLevelFlags`

The level of force to use in deleting the connection.

This parameter can be one of the following values defined in the *lmuseflg.h* header file.

VALUE	MEANING
USE_NOFORCE	Fail the disconnection if open files exist on the connection.
USE_FORCE	Do not fail the disconnection if open files exist on the connection.
USE_LOTS_OF_FORCE	Close any open files and delete the connection.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

The **NetUseDel** function applies only to the Server Message Block (LAN Manager Workstation) client. The **NetUseDel** function does not support Distributed File System (DFS) shares or other network file systems. To terminate a connection to a share using a different network provider (WebDAV or a DFS share, for example), use the WNetCancelConnection2 function.

No special group membership is required to call the **NetUseDel** function. This function cannot be executed on a remote server except in cases of downlevel compatibility.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmuse.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUseAdd

Network Management Functions

Network Management Overview

Use Functions

WNetCancelConnection2

NetUseEnum function (lmuse.h)

2/1/2021 • 3 minutes to read • Edit Online

The **NetUseEnum** function lists all current connections between the local computer and resources on remote servers.

You can also use the WNetOpenEnum and the WNetEnumResource functions to enumerate network resources or connections.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUseEnum(
  LMSTR   UncServerName,
  DWORD   LevelFlags,
  LPBYTE  *BufPtr,
  DWORD   PreferedMaximumSize,
  LPDWORD EntriesRead,
  LPDWORD TotalEntries,
  LPDWORD ResumeHandle
);
```

Parameters

`UncServerName`

The UNC name of the computer on which to execute this function. If this is parameter is **NULL**, then the local computer is used. If the *UncServerName* parameter specified is a remote computer, then the remote computer must support remote RPC calls using the legacy Remote Access Protocol mechanism.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`LevelFlags`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies a local device name and the share name of a remote resource. The *BufPtr* parameter points to an array of USE_INFO_0 structures.
1	Specifies information about the connection between a local device and a shared resource, including connection status and type. The *BufPtr* parameter points to an array of USE_INFO_1 structures.
2	Specifies information about the connection between a local device and a shared resource. Information includes the connection status, connection type, user name, and domain name. The *BufPtr* parameter points to an array of USE_INFO_2 structures.

`BufPtr`

A pointer to the buffer that receives the information structures. The format of this data depends on the value of the *Level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function when the information is no longer needed. Note that you must free the buffer even if the function fails with **ERROR_MORE_DATA**.

`PreferedMaximumSize`

The preferred maximum length, in bytes, of the data to return. If **MAX_PREFERRED_LENGTH** is specified, the function allocates the amount of memory required for the data. If another value is specified in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns **ERROR_MORE_DATA**. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`EntriesRead`

A pointer to a value that receives the count of elements actually enumerated.

`TotalEntries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`ResumeHandle`

A pointer to a value that contains a resume handle which is used to continue the search. The handle should be zero on the first call and left unchanged for subsequent calls. If *ResumeHandle* is **NULL**, then no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

RETURN CODE	DESCRIPTION
ERROR_INVALID_PARAMETER	An invalid parameter was passed to the function. This error is returned if a **NULL** pointer is passed in the *BufPtr* or *entriesread* parameters.
ERROR_MORE_DATA	There is more data to return. This error is returned if the buffer size is insufficient to hold all entries.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if the *UncServerName* parameter was not **NULL** and the remote server does not support remote RPC calls using the legacy Remote Access Protocol mechanism.
Other	Use FormatMessage to obtain the message string for the returned error.

Remarks

No special group membership is required to call the **NetUseEnum** function. This function cannot be executed on a remote server except in cases of downlevel compatibility using the legacy Remote Access Protocol.

To retrieve information about one network connection, you can call the NetUseGetInfo function.

This function applies only to the Server Message Block (LAN Manager Workstation) client. The **NetUseEnum** function does not support Distributed File System (DFS) shares. To enumerate shares using a different network provider (WebDAV or a DFS share, for example), use the WNetOpenEnum, WNetEnumResource, and WNetCloseEnum functions.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmuse.h (include Lm.h, Lmcons.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUseGetInfo

Network Management Functions

Network Management Overview

USE_INFO_0

USE_INFO_1

USE_INFO_2

Use Functions

WNetCloseEnum

WNetEnumResource

WNetOpenEnum

NetUseGetInfo function (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetUseGetInfo** function retrieves information about a connection to a shared resource.

You can also use the WNetGetConnection function to retrieve the name of a network resource associated with a local device.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetUseGetInfo(
  LMSTR  UncServerName,
  LMSTR  UseName,
  DWORD  LevelFlags,
  LPBYTE *bufptr
);
```

Parameters

`UncServerName`

The UNC name of computer on which to execute this function. If this is parameter is **NULL**, then the local computer is used. If the *UncServerName* parameter specified is a remote computer, then the remote computer must support remote RPC calls using the legacy Remote Access Protocol mechanism.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`UseName`

A pointer to a string that specifies the name of the connection for which to return information.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`LevelFlags`

The information level of the data requested. This parameter can be one of the following values.

VALUE	MEANING
0	Specifies a local device name and the share name of a remote resource. The *BufPtr* parameter is a pointer to a USE_INFO_0 structure.
1	Specifies information about the connection between a local device and a shared resource, including connection status and type. The *BufPtr* parameter is a pointer to a USE_INFO_1 structure.
2	Specifies information about the connection between a local device and a shared resource. Information includes the connection status, connection type, user name, and domain name. The *BufPtr* parameter is a pointer to a USE_INFO_2 structure.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *Level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value is a system error code. For a list of error codes, see System Error Codes.

Remarks

No special group membership is required to call the **NetUseGetInfo** function. This function cannot be executed on a remote server except in cases of downlevel compatibility.

To list all current connections between the local computer and resources on remote servers, you can call the NetUseEnum function.

This function applies only to the Server Message Block (LAN Manager Workstation) client. The **NetUseGetInfo** function does not support Distributed File System (DFS) shares. To retrieve information for a share using a different network provider (WebDAV or a DFS share, for example), use the WNetGetConnection function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmuse.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetUseEnum

Network Management Functions

Network Management Overview

USE_INFO_0

USE_INFO_1

USE_INFO_2

Use Functions

WNetGetConnection

USE_INFO_0 structure (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USE_INFO_0** structure contains the name of a shared resource and the local device redirected to it.

Syntax

```
typedef struct _USE_INFO_0 {
  LMSTR ui0_local;
  LMSTR ui0_remote;
} USE_INFO_0, *PUSE_INFO_0, *LPUSE_INFO_0;
```

Members

`ui0_local`

Pointer to a Unicode string that specifies the local device name (for example, drive E or LPT1) being redirected to the shared resource. The constant DEVLEN specifies the maximum number of characters in the string.

`ui0_remote`

Pointer to a Unicode string that specifies the share name of the remote resource being accessed. The string is in the form:

```
\\servername\sharename
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmuse.h (include Lm.h)

See also

NetUseEnum

NetUseGetInfo

Network Management Overview

Network Management Structures

Use Functions

USE_INFO_1 structure (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USE_INFO_1** structure contains information about the connection between a local device and a shared resource. The information includes connection status and connection type.

Syntax

```
typedef struct _USE_INFO_1 {
  LMSTR ui1_local;
  LMSTR ui1_remote;
  LMSTR ui1_password;
  DWORD ui1_status;
  DWORD ui1_asg_type;
  DWORD ui1_refcount;
  DWORD ui1_usecount;
} USE_INFO_1, *PUSE_INFO_1, *LPUSE_INFO_1;
```

Members

`ui1_local`

Type: **LMSTR**

A pointer to a string that contains the local device name (for example, drive E or LPT1) being redirected to the shared resource. The constant DEVLEN specifies the maximum number of characters in the string. This member can be **NULL**. For more information, see the following Remarks section.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui1_remote`

Type: **LMSTR**

A pointer to a string that contains the share name of the remote resource being accessed. The string is in the form:

```
\\servername\sharename
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui1_password`

Type: **LMSTR**

A pointer to a string that contains the password needed to establish a session between a specific workstation and a server.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui1_status`

Type: **DWORD**

The status of the connection. This element is not used by the NetUseAdd function. The following values are defined.

VALUE	MEANING
USE_OK	The connection is valid.
USE_PAUSED	Paused by local workstation.
USE_SESSLOST	Disconnected.
USE_DISCONN	An error occurred.
USE_NETERR	A network error occurred.
USE_CONN	The connection is being made.
USE_RECONN	Reconnecting.

`ui1_asg_type`

Type: **DWORD**

The type of remote resource being accessed. This member can be one of the following values.

VALUE	MEANING
USE_WILDCARD	Matches the type of the server's shared resources. Wildcards can be used only with the NetUseAdd function, and only when the **ui1_local** member is **NULL**. For more information, see the following Remarks section.
USE_DISKDEV	Disk device.
USE_SPOOLDEV	Spooled printer.
USE_IPC	Interprocess communication (IPC).

`ui1_refcount`

Type: **DWORD**

The number of files, directories, and other processes that are open on the remote resource. This element is not used by the NetUseAdd function.

`ui1_usecount`

Type: **DWORD**

The number of explicit connections (redirection with a local device name) or implicit UNC connections (redirection without a local device name) that are established with the resource.

Remarks

Specifying a **ui1_local** member that is **NULL** requests authentication with the server without redirecting a drive letter or a device. Future redirections involving the server while the same connection is in effect use the password specified by the **ui1_password** member in the initial call to the NetUseAdd function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmuse.h (include Lm.h)

See also

NetUseAdd

NetUseEnum

NetUseGetInfo

Network Management Overview

Network Management Structures

Use Functions

USE_INFO_2 structure (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USE_INFO_2** structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, and domain name.

Syntax

```
typedef struct _USE_INFO_2 {
  LMSTR ui2_local;
  LMSTR ui2_remote;
  LMSTR ui2_password;
  DWORD ui2_status;
  DWORD ui2_asg_type;
  DWORD ui2_refcount;
  DWORD ui2_usecount;
  LMSTR ui2_username;
  LMSTR ui2_domainname;
} USE_INFO_2, *PUSE_INFO_2, *LPUSE_INFO_2;
```

Members

`ui2_local`

Type: **LMSTR**

A pointer to a string that contains the local device name (for example, drive E or LPT1) being redirected to the shared resource. The constant DEVLEN specifies the maximum number of characters in the string. This member can be **NULL**. For more information, see the following Remarks section.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui2_remote`

Type: **LMSTR**

A pointer to a string that contains the share name of the remote resource. The string is in the form

```
\\servername\sharename
```

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui2_password`

Type: **LMSTR**

A pointer to a string that contains the password needed to establish a session with a specific workstation.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui2_status`

Type: **DWORD**

The status of the connection. This element is not used by the NetUseAdd function. The following values are

defined.

VALUE	MEANING
USE_OK	The connection is successful.
USE_PAUSED	Paused by a local workstation.
USE_SESSLOST	Disconnected.
USE_DISCONN	An error occurred.
USE_NETERR	A network error occurred.
USE_CONN	The connection is being made.
USE_RECONN	Reconnecting.

`ui2_asg_type`

Type: **DWORD**

The type of remote resource being accessed. This member can be one of the following values.

VALUE	MEANING
USE_WILDCARD	Matches the type of the server's shared resources. Wildcards can be used only with the NetUseAdd function, and only when the **ui2_local** member is a **NULL** string. For more information, see the following Remarks section.
USE_DISKDEV	Disk device.
USE_SPOOLDEV	Spooled printer.
USE_IPC	Interprocess communication (IPC).

`ui2_refcount`

Type: **DWORD**

The number of files, directories, and other processes that are open on the remote resource. This element is not used by the **NetUseAdd** function.

`ui2_usecount`

Type: **DWORD**

The number of explicit connections (redirection with a local device name) or implicit UNC connections (redirection without a local device name) that are established with the resource.

`ui2_username`

Type: **LPWSTR**

A pointer to a string that contains the name of the user who initiated the connection.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`ui2_domainname`

Type: **LMSTR**

A pointer to a string that contains the domain name of the remote resource.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Remarks

Specifying a **ui2_local** member that is **NULL** requests authentication with the server without redirecting a drive letter or a device. Future redirections involving the server while the same connection is in effect use the authentication information specified in the initial call to the NetUseAdd function. This information includes the combination of the **ui2_password**, **ui2_username**, and **ui2_domainname** members.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmuse.h (include Lm.h)

See also

NetUseAdd

NetUseEnum

NetUseGetInfo

Network Management Overview

Network Management Structures

Use Functions

USE_INFO_3 structure (lmuse.h)

2/1/2021 • 2 minutes to read • Edit Online

The **USE_INFO_3** structure contains information about a connection between a local computer and a shared resource, including connection type, connection status, user name, domain name, and specific flags that describe connection behavior.

Syntax

```
typedef struct _USE_INFO_3 {
  USE_INFO_2 ui3_ui2;
  ULONG      ui3_flags;
} USE_INFO_3, *PUSE_INFO_3, *LPUSE_INFO_3;
```

Members

`ui3_ui2`

USE_INFO_2 structure that contains

`ui3_flags`

A set of bit flags that describe connection behavior and credential handling.

VALUE	MEANING
CREATE_NO_CONNECT	Do not connect to the server.
CREATE_BYPASS_CSC	Force a connection to the server, bypassing the CSC.
USE_DEFAULT_CREDENTIALS	No explicit credentials are supplied in the call to NetUseAdd.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmuse.h (include Lm.h)

See also

NetUseEnum

NetUseGetInfo

Network Management Overview

Network Management Structures

Use functions

lmwksta.h header

2/1/2021 • 2 minutes to read • Edit Online

This header is used by Network Management. For more information, see:

- Network Management lmwksta.h contains the following programming interfaces:

Functions

TITLE	DESCRIPTION
NetWkstaGetInfo	The NetWkstaGetInfo function returns information about the configuration of a workstation.
NetWkstaSetInfo	The NetWkstaSetInfo function configures a workstation with information that remains in effect after the system has been reinitialized.
NetWkstaTransportAdd	Not supported.
NetWkstaTransportDel	Not supported.
NetWkstaTransportEnum	The NetWkstaTransportEnum function supplies information about transport protocols that are managed by the redirector, which is the software on the client computer that generates file requests to the server computer.
NetWkstaUserEnum	The NetWkstaUserEnum function lists information about all users currently logged on to the workstation. This list includes interactive, service and batch logons.
NetWkstaUserGetInfo	The NetWkstaUserGetInfo function returns information about the currently logged-on user. This function must be called in the context of the logged-on user.
NetWkstaUserSetInfo	The NetWkstaUserSetInfo function sets the user-specific information about the configuration elements for a workstation.

Structures

TITLE	DESCRIPTION
WKSTA_INFO_100	Contains information about a workstation environment, including platform-specific information, the names of the domain and the local computer, and information concerning the operating system.
WKSTA_INFO_101	Contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.

TITLE	DESCRIPTION
WKSTA_INFO_102	Contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.
WKSTA_INFO_502	The WKSTA_INFO_502 structure is obsolete. The structure contains information about a workstation environment.
WKSTA_TRANSPORT_INFO_0	The WKSTA_TRANSPORT_INFO_0 structure contains information about the workstation transport protocol, such as Wide Area Network (WAN) or NetBIOS.
WKSTA_USER_INFO_0	The WKSTA_USER_INFO_0 structure contains the name of the user on a specified workstation.
WKSTA_USER_INFO_1	The WKSTA_USER_INFO_1 structure contains user information as it pertains to a specific workstation. The information includes the name of the current user and the domains accessed by the workstation.
WKSTA_USER_INFO_1101	The WKSTA_USER_INFO_1101 structure contains information about the domains accessed by a workstation.

NetWkstaGetInfo function (lmwksta.h)

2/1/2021 • 4 minutes to read • Edit Online

The **NetWkstaGetInfo** function returns information about the configuration of a workstation.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaGetInfo(
  LMSTR  servername,
  DWORD  level,
  LPBYTE *bufptr
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
100	Return information about the workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system. The *bufptr* parameter points to a WKSTA_INFO_100 structure.
101	In addition to level 100 information, return the path to the LANMAN directory. The *bufptr* parameter points to a WKSTA_INFO_101 structure.
102	In addition to level 101 information, return the number of users who are logged on to the local computer. The *bufptr* parameter points to a WKSTA_INFO_102 structure.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION

ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The *level* parameter is invalid.

Remarks

Windows Server 2003 and Windows XP: If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the ACL for the securable object. To enable anonymous access, the user Anonymous must be a member of the "Pre-Windows 2000 compatible access" group. This is because anonymous tokens do not include the Everyone group SID by default. If you call this function on a member server or workstation, all authenticated users can view the information. Anonymous access is also permitted if the EveryoneIncludesAnonymous policy setting allows anonymous access. Anonymous access is always permitted for level 100. If you call this function at level 101, authenticated users can view the information. Members of the Administrators, and the Server, System and Print Operator local groups can view information at levels 102 and 502. For more information about restricting anonymous access, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

Windows 2000: If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the " Pre-Windows 2000 compatible access" group to view the information. By default, the "Pre-Windows 2000 compatible access" group includes Everyone as a member. This enables anonymous access to the information if the system allows anonymous access. If you call this function on a member server or workstation, all authenticated users can view the information. Anonymous access is also permitted if the RestrictAnonymous policy setting allows anonymous access.

To compile an application that uses this function, define the _WIN32_WINNT macro as 0x0400 or later. For more information,see Using the Windows Headers.

Examples

The following code sample demonstrates how to retrieve information about the configuration elements for a workstation using a call to the **NetWkstaGetInfo** function. The sample calls **NetWkstaGetInfo**, specifying information level 102 (WKSTA_INFO_102). If the call succeeds, the sample prints information about the workstation. Finally, the code sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 102;
   LPWKSTA_INFO_102 pBuf = NULL;
   NET_API_STATUS nStatus;
   LPWSTR pszServerName = NULL;
   //
   // Check command line arguments.
   //
   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = argv[1];
   //
   // Call the NetWkstaGetInfo function, specifying level 102.
   //
   nStatus = NetWkstaGetInfo(pszServerName,
                             dwLevel,
                             (LPBYTE *)&pBuf);
   //
   // If the call is successful,
   //  print the workstation data.
   //
   if (nStatus == NERR_Success)
   {
      printf("\n\tPlatform: %d\n", pBuf->wki102_platform_id);
      wprintf(L"\tName:     %s\n", pBuf->wki102_computername);
      printf("\tVersion:  %d.%d\n", pBuf->wki102_ver_major,
                                    pBuf->wki102_ver_minor);
      wprintf(L"\tDomain:   %s\n", pBuf->wki102_langroup);
      wprintf(L"\tLan Root: %s\n", pBuf->wki102_lanroot);
      wprintf(L"\t# Logged On Users: %d\n", pBuf->wki102_logged_on_users);
   }
   //
   // Otherwise, indicate the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaSetInfo

Network Management Functions

Network Management Overview

WKSTA_INFO_100

WKSTA_INFO_101

WKSTA_INFO_102

Workstation and Workstation User Functions

NetWkstaSetInfo function (lmwksta.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetWkstaSetInfo** function configures a workstation with information that remains in effect after the system has been reinitialized.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaSetInfo(
  LMSTR   servername,
  DWORD   level,
  LPBYTE  buffer,
  LPDWORD parm_err
);
```

Parameters

`servername`

A pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

The information level of the data. This parameter can be one of the following values.

VALUE	MEANING
100	**Windows NT:** Specifies information about a workstation environment, including platform-specific information, the names of the domain and the local computer, and information concerning the operating system. The *buffer* parameter points to a WKSTA_INFO_100 structure. The **wk100_computername** and **wk100_langroup** fields of this structure cannot be set by calling this function. To set these values, call SetComputerName/SetComputerNameEx or NetJoinDomain, respectively.
101	**Windows NT:** In addition to level 100 information, specifies the path to the LANMAN directory. The *buffer* parameter points to a WKSTA_INFO_101 structure. The **wk101_computername** and **wk101_langroup** fields of this structure cannot be set by calling this function. To set these values, call SetComputerName/SetComputerNameEx or **NetJoinDomain**, respectively.
102	**Windows NT:** In addition to level 101 information, specifies the number of users who are logged on to the local computer. The *buffer* parameter points to a WKSTA_INFO_102 structure. The **wk102_computername** and **wk102_langroup** fields of this structure cannot be set by calling this function. To set these values, call SetComputerName/SetComputerNameEx or **NetJoinDomain**, respectively.

502	**Windows NT:** The *buffer* parameter points to a WKSTA_INFO_502 structure that contains information about the workstation environment.

Do not set levels 1010-1013, 1018, 1023, 1027, 1028, 1032, 1033, 1035, or 1041-1062.

`buffer`

A pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

A pointer to a value that receives the index of the first member of the workstation information structure that causes the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error. For more information, see the Remarks section.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid. For more information, see the following Remarks section.

Remarks

Only members of the Administrators group can successfully execute the **NetWkstaSetInfo** function on a remote server.

The **NetWkstaSetInfo** function calls the workstation service on the local system or a remote system. Only a limited number of members of the WKSTA_INFO_502 structure can actually be changed using the **NetWkstaSetInfo** function. No errors are returned if a member is set that is ignored by the workstation service. The workstation service is primarily configured using settings in the registry.

The NetWkstaUserSetInfo function can be used instead of the **NetWkstaSetInfo** function to set configuration information on the local system. The **NetWkstaUserSetInfo** function calls the Local Security Authority (LSA).

If the **NetWkstaSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the first member of the workstation information structure that is invalid. (A workstation information structure begins with WKSTA_INFO_ and its format is specified by the *level* parameter.) The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix wki*_ indicates that the member can begin with multiple prefixes, for example, wki100_ or wki402_.)

VALUE	MEMBER
WKSTA_PLATFORM_ID_PARMNUM	wki*_platform_id

WKSTA_COMPUTERNAME_PARMNUM	wki*_computername
WKSTA_LANGROUP_PARMNUM	wki*_langroup
WKSTA_VER_MAJOR_PARMNUM	wki*_ver_major
WKSTA_VER_MINOR_PARMNUM	wki*_ver_minor
WKSTA_LOGGED_ON_USERS_PARMNUM	wki*_logged_on_users
WKSTA_LANROOT_PARMNUM	wki*_lanroot
WKSTA_LOGON_DOMAIN_PARMNUM	wki*_logon_domain
WKSTA_LOGON_SERVER_PARMNUM	wki*_logon_server
WKSTA_CHARWAIT_PARMNUM	wki*_char_wait
WKSTA_CHARTIME_PARMNUM	wki*_collection_time
WKSTA_CHARCOUNT_PARMNUM	wki*_maximum_collection_count
WKSTA_KEEPCONN_PARMNUM	wki*_keep_conn
WKSTA_KEEPSEARCH_PARMNUM	wki*_keep_search
WKSTA_MAXCMDS_PARMNUM	wki*_max_cmds
WKSTA_NUMWORKBUF_PARMNUM	wki*_num_work_buf
WKSTA_MAXWRKCACHE_PARMNUM	wki*_max_wrk_cache
WKSTA_SESSTIMEOUT_PARMNUM	wki*_sess_timeout
WKSTA_SIZERROR_PARMNUM	wki*_siz_error
WKSTA_NUMALERTS_PARMNUM	wki*_num_alerts
WKSTA_NUMSERVICES_PARMNUM	wki*_num_services
WKSTA_ERRLOGSZ_PARMNUM	wki*_errlog_sz
WKSTA_PRINTBUFTIME_PARMNUM	wki*_print_buf_time
WKSTA_NUMCHARBUF_PARMNU	wki*_num_char_buf
WKSTA_SIZCHARBUF_PARMNUM	wki*_siz_char_buf
WKSTA_WRKHEURISTICS_PARMNUM	wki*_wrk_heuristics
WKSTA_MAILSLOTS_PARMNUM	wki*_mailslots

WKSTA_MAXTHREADS_PARMNUM	wki*_max_threads
WKSTA_SIZWORKBUF_PARMNUM	wki*_siz_work_buf
WKSTA_NUMDGRAMBUF_PARMNUM	wki*_num_dgram_buf

The workstation service parameter settings are stored in the registry, not in the LanMan.ini file used prveiously by LAN Manager. The **NetWkstaSetInfo** function does not change the values in the LanMan.ini file. When the workstation service is stopped and restarted, workstation parameters are reset to the default values specified in the registry (unless they are overwritten by command-line parameters). Values set by previous calls to **NetWkstaSetInfo** can be overwritten when workstation parameters are reset.

Examples

The following code sample demonstrates how to set the session time-out value associated with a workstation using a call to the **NetServerSetInfo** function. (The session time-out is the number of seconds the server waits before disconnecting an inactive session.) The code specifies information level 502 (WKSTA_INFO_502).

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   LPWKSTA_INFO_502 pBuf = NULL;
   WKSTA_INFO_502 wi;
   DWORD dwLevel = 502;
   NET_API_STATUS nStatus;
   LPWSTR pszServerName = NULL;

   if ((argc < 2) || (argc > 3))
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName] SessionTimeOut\n", argv[0]);
      exit(1);
   }

   if (argc == 3)
      pszServerName = argv[1];
   //
   // Retrieve the current settings.
   //
   nStatus = NetWkstaGetInfo(pszServerName,
                             dwLevel,
                             (LPBYTE *)&pBuf);

   if (nStatus != NERR_Success)
   {
      fprintf(stderr, "A system error has occurred (NetWkstaGetInfo): %d\n", nStatus);
      return -1;
   }

   if (pBuf != NULL)
   {
      //
      // Copy the existing settings to the new structure,
      //   and free the buffer.
      //
      CopyMemory(&wi, pBuf, sizeof(wi));
```

```
        NetApiBufferFree(pBuf);
    }
    else
    {
        fprintf(stderr, "Memory invalid!\n");
        return -1;
    }
    //
    // Set a new session time-out value by
    //   calling the NetWkstaSetInfo function.
    //
    wi.wki502_sess_timeout = _wtoi(argv[argc-1]);
    nStatus = NetWkstaSetInfo(pszServerName,
                              dwLevel,
                              (LPBYTE)&wi,
                              NULL);
    //
    // Display the result of the call.
    //
    if (nStatus == NERR_Success)
        fwprintf(stderr, L"Workstation information reset: session time-out = %d\n", _wtoi(argv[argc-1]));
    else
        fprintf(stderr, "A system error has occurred (NetWkstaSetInfo): %d\n", nStatus);

    return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaGetInfo

NetWkstaUserGetInfo

NetWkstaUserSetInfo

Network Management Functions

Network Management Overview

Workstation and Workstation User Functions

NetWkstaTransportAdd function (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. To change the default settings for transport protocols manually, use the **Local Area Connection Properties** dialog box in the **Network and Dial-Up Connections** folder.]

Not supported.

The **NetWkstaTransportAdd** function binds (or connects) the redirector to the transport. The redirector is the software on the client computer which generates file requests to the server computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaTransportAdd(
  LPTSTR  servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

This string must begin with \.

`level`

Specifies the information level of the data. This parameter can be the following value.

VALUE	MEANING
0	Specifies workstation transport protocol information. The *buf* parameter points to a WKSTA_TRANSPORT_INFO_0 structure.

`buf`

Pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first parameter that causes the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_LEVEL	The level parameter, which indicates what level of data structure information is available, is invalid.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid.

Remarks

Only members of the Administrators local group can successfully execute the **NetWkstaTransportAdd** function.

If the **NetWkstaTransportAdd** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the member of the **WKSTA_TRANSPORT_INFO_0** structure that is invalid. The following table lists the values that can be returned in the *parm_err* parameter and the corresponding structure member that is in error.

VALUE	MEMBER
TRANSPORT_QUALITYOFSERVICE_PARMNUM	**wkti0_quality_of_service**
TRANSPORT_NAME_PARMNUM	**wkti0_transport_name**

Requirements

Target Platform	Windows
Header	lmwksta.h (include Lm.h, Lmwksta.h)
Library	Netapi32.lib
DLL	Netapi32.dll

NetWkstaTransportDel function (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

[This function is obsolete. To change the default settings for transport protocols manually, use the **Local Area Connection Properties** dialog box in the **Network and Dial-Up Connections** folder.]

Not supported.

The **NetWkstaTransportDel** function unbinds the transport protocol from the redirector. (The redirector is the software on the client computer that generates file requests to the server computer.)

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaTransportDel(
  LMSTR servername,
  LMSTR transportname,
  DWORD ucond
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

This string must begin with \.

`transportname`

Pointer to a string that specifies the name of the transport protocol to disconnect from the redirector.

`ucond`

Specifies the level of force to use when disconnecting the transport protocol from the redirector. This parameter can be one of the following values.

VALUE	MEANING
USE_NOFORCE	Fail the disconnection if open files exist on the connection.
USE_FORCE	Fail the disconnection if open files exist on the connection.
USE_LOTS_OF_FORCE	Close any open files and delete the connection.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid.
NERR_UseNotFound	The network connection does not exist.

Remarks

Only members of the Administrators local group can successfully execute the **NetWkstaTransportDel** function.

Requirements

Target Platform	Windows
Header	lmwksta.h (include Lm.h, Lmwksta.h)
Library	Netapi32.lib
DLL	Netapi32.dll

NetWkstaTransportEnum function (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetWkstaTransportEnum** function supplies information about transport protocols that are managed by the redirector, which is the software on the client computer that generates file requests to the server computer.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaTransportEnum(
  LPTSTR  servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resume_handle
);
```

Parameters

`servername`

A pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

The level of information requested for the data. This parameter can be the following value.

VALUE	MEANING
0	Return workstation transport protocol information. The *bufptr* parameter points to an array of WKSTA_TRANSPORT_INFO_0 structures.

`bufptr`

A pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with **ERROR_MORE_DATA** or **NERR_BufTooSmall**.

`prefmaxlen`

The preferred maximum length of returned data, in bytes. If you specify **MAX_PREFERRED_LENGTH**, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns **ERROR_MORE_DATA** or **NERR_BufTooSmall**. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

A pointer to a value that receives the count of elements actually enumerated.

`totalentries`

A pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`resume_handle`

A pointer to a value that contains a resume handle which is used to continue an existing workstation transport search. The handle should be zero on the first call and left unchanged for subsequent calls. If the *resumehandle* parameter is a **NULL** pointer, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_INVALID_LEVEL	The level parameter, which indicates what level of data structure information is available, is invalid. This error is returned if the *level* parameter is specified as a value other than zero.
ERROR_INVALID_PARAMETER	One or more parameters was invalid. This error is returned if the *bufptr* or the *entriesread* parameters are **NULL** pointers.
ERROR_NOT_ENOUGH_MEMORY	Insufficient memory was available to process the request.
ERROR_NOT_SUPPORTED	The request is not supported. This error is returned if a remote server was specified in *servername* parameter, and this request is not supported on the remote server.
NERR_BufTooSmall	More entries are available. Specify a large enough buffer to receive all entries. This error code is defined in the *Lmerr.h* header file.

Remarks

No special group membership is required to successfully execute the **NetWkstaTransportEnum** function.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)

Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaTransportAdd

NetWkstaTransportDel

Network Management Functions

Network Management Overview

Server and Workstation Transport Functions

WKSTA_TRANSPORT_INFO_0

NetWkstaUserEnum function (lmwksta.h)

2/1/2021 • 5 minutes to read • Edit Online

The **NetWkstaUserEnum** function lists information about all users currently logged on to the workstation. This list includes interactive, service and batch logons.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaUserEnum(
  LMSTR   servername,
  DWORD   level,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  LPDWORD resumehandle
);
```

Parameters

`servername`

Pointer to a string that specifies the DNS or NetBIOS name of the remote server on which the function is to execute. If this parameter is **NULL**, the local computer is used.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the names of users currently logged on to the workstation. The *bufptr* parameter points to an array of WKSTA_USER_INFO_0 structures.
1	Return the names of the current users and the domains accessed by the workstation. The *bufptr* parameter points to an array of WKSTA_USER_INFO_1 structures.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *level* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. Note that you must free the buffer even if the function fails with ERROR_MORE_DATA.

`prefmaxlen`

Specifies the preferred maximum length of returned data, in bytes. If you specify MAX_PREFERRED_LENGTH, the function allocates the amount of memory required for the data. If you specify another value in this parameter, it can restrict the number of bytes that the function returns. If the buffer size is insufficient to hold all entries, the function returns ERROR_MORE_DATA. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

`entriesread`

Pointer to a value that receives the count of elements actually enumerated.

`totalentries`

Pointer to a value that receives the total number of entries that could have been enumerated from the current resume position. Note that applications should consider this value only as a hint.

`resumehandle`

Pointer to a value that contains a resume handle which is used to continue an existing search. The handle should be zero on the first call and left unchanged for subsequent calls. If this parameter is **NULL**, no resume handle is stored.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION
ERROR_ACCESS_DENIED	The user does not have access to the requested information.
ERROR_MORE_DATA	More entries are available. Specify a large enough buffer to receive all entries.
ERROR_INVALID_LEVEL	The *level* parameter is invalid.

Remarks

Note that since the **NetWkstaUserEnum** function lists entries for service and batch logons, as well as for interactive logons, the function can return entries for users who have logged off a workstation. This can occur, for example, when a user calls a service that impersonates the user. In this instance, **NetWkstaUserEnum** returns an entry for the user until the service stops impersonating the user.

Windows Server 2003 and Windows XP: If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the ACL for the securable object. To enable anonymous access, the user Anonymous must be a member of the "Pre-Windows 2000 compatible access" group. This is because anonymous tokens do not include the Everyone group SID by default. If you call this function on a member server or workstation, all authenticated users can view the information. Anonymous access is also permitted if the RestrictAnonymous policy setting permits anonymous access. If the RestrictAnonymous policy setting does not permit anonymous access, only an administrator can successfully execute the function. Members of the Administrators, and the Server, System and Print Operator local groups can also view information. For more information about restricting anonymous access, see Security Requirements for the Network Management Functions. For more information on ACLs, ACEs, and access tokens, see Access Control Model.

Windows 2000: If you call this function on a domain controller that is running Active Directory, access is allowed or denied based on the access control list (ACL) for the securable object. The default ACL permits all authenticated users and members of the " Pre-Windows 2000 compatible access" group to view the information. By default, the "Pre-Windows 2000 compatible access" group includes Everyone as a member. This

enables anonymous access to the information if the system allows anonymous access. If you call this function on a member server or workstation, all authenticated users can view the information. Anonymous access is also permitted if the RestrictAnonymous policy setting allows anonymous access.

To compile an application that uses this function, define the _WIN32_WINNT macro as 0x0400 or later. For more information,see Using the Windows Headers.

Examples

The following code sample demonstrates how to list information about all users currently logged on to a workstation using a call to the **NetWkstaUserEnum** function. The sample calls **NetWkstaUserEnum**, specifying information level 0 (WKSTA_USER_INFO_0). The sample loops through the entries and prints the names of the users logged on to a workstation. Finally, the code sample frees the memory allocated for the information buffer, and prints the total number of users enumerated.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

#include <stdio.h>
#include <assert.h>
#include <windows.h>
#include <lm.h>

int wmain(int argc, wchar_t *argv[])
{
   LPWKSTA_USER_INFO_0 pBuf = NULL;
   LPWKSTA_USER_INFO_0 pTmpBuf;
   DWORD dwLevel = 0;
   DWORD dwPrefMaxLen = MAX_PREFERRED_LENGTH;
   DWORD dwEntriesRead = 0;
   DWORD dwTotalEntries = 0;
   DWORD dwResumeHandle = 0;
   DWORD i;
   DWORD dwTotalCount = 0;
   NET_API_STATUS nStatus;
   LPWSTR pszServerName = NULL;

   if (argc > 2)
   {
      fwprintf(stderr, L"Usage: %s [\\\\ServerName]\n", argv[0]);
      exit(1);
   }
   // The server is not the default local computer.
   //
   if (argc == 2)
      pszServerName = argv[1];
   fwprintf(stderr, L"\nUsers currently logged on %s:\n", pszServerName);
   //
   // Call the NetWkstaUserEnum function, specifying level 0.
   //
   do // begin do
   {
      nStatus = NetWkstaUserEnum( pszServerName,
                                  dwLevel,
                                  (LPBYTE*)&pBuf,
                                  dwPrefMaxLen,
                                  &dwEntriesRead,
                                  &dwTotalEntries,
                                  &dwResumeHandle);
      //
      // If the call succeeds,
      //
      if ((nStatus == NERR_Success) || (nStatus == ERROR_MORE_DATA))
      {
```

```
         if ((pTmpBuf = pBuf) != NULL)
         {
            //
            // Loop through the entries.
            //
            for (i = 0; (i < dwEntriesRead); i++)
            {
               assert(pTmpBuf != NULL);

               if (pTmpBuf == NULL)
               {
                  //
                  // Only members of the Administrators local group
                  //  can successfully execute NetWkstaUserEnum
                  //  locally and on a remote server.
                  //
                  fprintf(stderr, "An access violation has occurred\n");
                  break;
               }
               //
               // Print the user logged on to the workstation.
               //
               wprintf(L"\t-- %s\n", pTmpBuf->wkui0_username);

               pTmpBuf++;
               dwTotalCount++;
            }
         }
      }
      //
      // Otherwise, indicate a system error.
      //
      else
         fprintf(stderr, "A system error has occurred: %d\n", nStatus);
      //
      // Free the allocated memory.
      //
      if (pBuf != NULL)
      {
         NetApiBufferFree(pBuf);
         pBuf = NULL;
      }
   }
   //
   // Continue to call NetWkstaUserEnum while
   //  there are more entries.
   //
   while (nStatus == ERROR_MORE_DATA); // end do
   //
   // Check again for allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);
   //
   // Print the final count of workstation users.
   //
   fprintf(stderr, "\nTotal of %d entries enumerated\n", dwTotalCount);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaGetInfo

NetWkstaSetInfo

Network Management Functions

Network Management Overview

WKSTA_USER_INFO_0

WKSTA_USER_INFO_1

Workstation and Workstation User Functions

NetWkstaUserGetInfo function (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetWkstaUserGetInfo** function returns information about the currently logged-on user. This function must be called in the context of the logged-on user.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaUserGetInfo(
  LMSTR  reserved,
  DWORD  level,
  LPBYTE *bufptr
);
```

Parameters

`reserved`

This parameter must be set to **NULL**.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
0	Return the name of the user currently logged on to the workstation. The *bufptr* parameter points to a WKSTA_USER_INFO_0 structure.
1	Return information about the workstation, including the name of the current user and the domains accessed by the workstation. The *bufptr* parameter points to a WKSTA_USER_INFO_1 structure.
1101	Return domains browsed by the workstation. The *bufptr* parameter points to a WKSTA_USER_INFO_1101 structure.

`bufptr`

Pointer to the buffer that receives the data. The format of this data depends on the value of the *bufptr* parameter. This buffer is allocated by the system and must be freed using the NetApiBufferFree function. For more information, see Network Management Function Buffers and Network Management Function Buffer Lengths.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION

ERROR_NOT_ENOUGH_MEMORY	The system ran out of memory resources. Either the network manager configuration is incorrect, or the program is running on a system with insufficient memory.
ERROR_INVALID_LEVEL	The *level* parameter is invalid.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid.

Remarks

The **NetWkstaUserGetInfo** function only works locally.

Examples

The following code sample demonstrates how to retrieve information about the currently logged-on user using a call to the **NetWkstaUserGetInfo** function. The sample calls **NetWkstaUserGetInfo**, specifying information level 1 (WKSTA_USER_INFO_1). If the call succeeds, the sample prints information about the logged-on user. Finally, the sample frees the memory allocated for the information buffer.

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(void)
{
   DWORD dwLevel = 1;
   LPWKSTA_USER_INFO_1 pBuf = NULL;
   NET_API_STATUS nStatus;
   //
   // Call the NetWkstaUserGetInfo function;
   //  specify level 1.
   //
   nStatus = NetWkstaUserGetInfo(NULL,
                                 dwLevel,
                                 (LPBYTE *)&pBuf);
   //
   // If the call succeeds, print the information
   //  about the logged-on user.
   //
   if (nStatus == NERR_Success)
   {
      if (pBuf != NULL)
      {
         wprintf(L"\n\tUser:          %s\n", pBuf->wkui1_username);
         wprintf(L"\tDomain:        %s\n", pBuf->wkui1_logon_domain);
         wprintf(L"\tOther Domains: %s\n", pBuf->wkui1_oth_domains);
         wprintf(L"\tLogon Server:  %s\n", pBuf->wkui1_logon_server);
      }
   }
   // Otherwise, print the system error.
   //
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);
   //
   // Free the allocated memory.
   //
   if (pBuf != NULL)
      NetApiBufferFree(pBuf);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)

Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaSetInfo

Network Management Functions

Network Management Overview

WKSTA_USER_INFO_0

WKSTA_USER_INFO_1

WKSTA_USER_INFO_1101

Workstation and Workstation User Functions

NetWkstaUserSetInfo function (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **NetWkstaUserSetInfo** function sets the user-specific information about the configuration elements for a workstation.

Syntax

```
NET_API_STATUS NET_API_FUNCTION NetWkstaUserSetInfo(
  LMSTR   reserved,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

Parameters

`reserved`

This parameter must be set to zero.

`level`

Specifies the information level of the data. This parameter can be one of the following values.

VALUE	MEANING
1	Specifies information about the workstation, including the name of the current user and the domains accessed by the workstation. The *buf* parameter points to a WKSTA_USER_INFO_1 structure.
1101	Specifies domains browsed by the workstation. The *buf* parameter points to a WKSTA_USER_INFO_1101 structure.

`buf`

Pointer to the buffer that specifies the data. The format of this data depends on the value of the *level* parameter. For more information, see Network Management Function Buffers.

`parm_err`

Pointer to a value that receives the index of the first parameter that causes the ERROR_INVALID_PARAMETER error. If this parameter is **NULL**, the index is not returned on error.

Return value

If the function succeeds, the return value is NERR_Success.

If the function fails, the return value can be one of the following error codes.

RETURN CODE	DESCRIPTION

ERROR_INVALID_LEVEL	The *level* parameter is invalid.
ERROR_INVALID_PARAMETER	One of the function parameters is invalid.

Remarks

The **NetWkstaUserSetInfo** function only works locally. Administrator group membership is required.

Domain names in the **wkui1101_oth_domains** member of the WKSTA_USER_INFO_1101 structure are separated by spaces. An empty list is valid. A **NULL** pointer means to leave the member unmodified. The **wkui1101_oth_domains** member cannot be set with MS-DOS. When setting this element, **NetWkstaUserSetInfo** rejects the request if the name list was invalid or if a name could not be added to one or more of the network adapters managed by the system.

If the **NetWkstaUserSetInfo** function returns ERROR_INVALID_PARAMETER, you can use the *parm_err* parameter to indicate the member of the workstation user information structure that is invalid. (A workstation user information structure begins with WKSTA_USER_INFO_ and its format is specified by the *level* parameter.) The following table lists the value that can be returned in the *parm_err* parameter and the corresponding structure member that is in error. (The prefix wkui*_ indicates that the member can begin with multiple prefixes, for example, wkui0_ or wkui1_.)

VALUE	MEMBER
WKSTA_OTH_DOMAINS_PARMNUM	wkui*_oth_domains

Examples

The following code sample demonstrates how to set user-specific information for a workstation using a call to the **NetWkstaUserSetInfo** function, specifying information level 1101 (WKSTA_USER_INFO_1101).

```
#ifndef UNICODE
#define UNICODE
#endif
#pragma comment(lib, "netapi32.lib")

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

int wmain(int argc, wchar_t *argv[])
{
   DWORD dwLevel = 1101;
   WKSTA_USER_INFO_1101 wui;
   NET_API_STATUS nStatus;

   if (argc != 2)
   {
      fwprintf(stderr, L"Usage: %s OtherDomains\n", argv[0]);
      exit(1);
   }
   //
   // Fill in the WKSTA_USER_INFO_1101 structure member.
   //
   wui.wkui1101_oth_domains = argv[1];
   //
   // Call the NetWkstaUserSetInfo function
   //  to change the list of domains browsed by
   //  the workstation; specify level 1101.
   //
   nStatus = NetWkstaUserSetInfo(NULL,
                                 dwLevel,
                                 (LPBYTE)&wui,
                                 NULL);
   //
   // Display the result of the call.
   //
   if (nStatus == NERR_Success)
      fprintf(stderr, "Workstation user information has been changed\n");
   else
      fprintf(stderr, "A system error has occurred: %d\n", nStatus);

   return 0;
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	lmwksta.h (include Lm.h)
Library	Netapi32.lib
DLL	Netapi32.dll

See also

NetWkstaUserGetInfo

Network Management Functions

Network Management Overview

WKSTA_USER_INFO_1

WKSTA_USER_INFO_1101

Workstation and Workstation User Functions

WKSTA_INFO_100 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_INFO_100** structure contains information about a workstation environment, including platform-specific information, the names of the domain and the local computer, and information concerning the operating system.

Syntax

```
typedef struct _WKSTA_INFO_100 {
  DWORD wki100_platform_id;
  LMSTR wki100_computername;
  LMSTR wki100_langroup;
  DWORD wki100_ver_major;
  DWORD wki100_ver_minor;
} WKSTA_INFO_100, *PWKSTA_INFO_100, *LPWKSTA_INFO_100;
```

Members

`wki100_platform_id`

Type: **DWORD**

The information level to use to retrieve platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.
PLATFORM_ID_VMS 700	The VMS platform.

`wki100_computername`

Type: **LMSTR**

A pointer to a string specifying the name of the local computer.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki100_langroup`

Type: **LMSTR**

A pointer to a string specifying the name of the domain to which the computer belongs.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki100_ver_major`

Type: **DWORD**

The major version number of the operating system running on the computer.

`wki100_ver_minor`

Type: **DWORD**

The minor version number of the operating system running on the computer.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaGetInfo

NetWkstaSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_INFO_101 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_INFO_101** structure contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.

Syntax

```
typedef struct _WKSTA_INFO_101 {
  DWORD wki101_platform_id;
  LMSTR wki101_computername;
  LMSTR wki101_langroup;
  DWORD wki101_ver_major;
  DWORD wki101_ver_minor;
  LMSTR wki101_lanroot;
} WKSTA_INFO_101, *PWKSTA_INFO_101, *LPWKSTA_INFO_101;
```

Members

`wki101_platform_id`

Type: **DWORD**

The information level to use to retrieve platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.
PLATFORM_ID_VMS 700	The VMS platform.

`wki101_computername`

Type: **LMSTR**

A pointer to a string specifying the name of the local computer.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki101_langroup`

Type: **LMSTR**

A pointer to a string specifying the name of the domain to which the computer belongs.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki101_ver_major`

Type: **DWORD**

The major version number of the operating system running on the computer.

`wki101_ver_minor`

Type: **DWORD**

The minor version number of the operating system running on the computer.

`wki101_lanroot`

Type: **LMSTR**

A pointer to a string that contains the path to the LANMAN directory.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaGetInfo

NetWkstaSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_INFO_102 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_INFO_102** structure contains information about a workstation environment, including platform-specific information, the name of the domain and the local computer, and information concerning the operating system.

Syntax

```
typedef struct _WKSTA_INFO_102 {
  DWORD wki102_platform_id;
  LMSTR wki102_computername;
  LMSTR wki102_langroup;
  DWORD wki102_ver_major;
  DWORD wki102_ver_minor;
  LMSTR wki102_lanroot;
  DWORD wki102_logged_on_users;
} WKSTA_INFO_102, *PWKSTA_INFO_102, *LPWKSTA_INFO_102;
```

Members

`wki102_platform_id`

Type: **DWORD**

The information level to use to retrieve platform-specific information.

Possible values for this member are listed in the *Lmcons.h* header file.

VALUE	MEANING
PLATFORM_ID_DOS 300	The MS-DOS platform.
PLATFORM_ID_OS2 400	The OS/2 platform.
PLATFORM_ID_NT 500	The Windows NT platform.
PLATFORM_ID_OSF 600	The OSF platform.
PLATFORM_ID_VMS 700	The VMS platform.

`wki102_computername`

Type: **LMSTR**

A pointer to a string specifying the name of the local computer.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki102_langroup`

Type: **LMSTR**

A pointer to a string specifying the name of the domain to which the computer belongs.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki102_ver_major`

Type: **DWORD**

The major version number of the operating system running on the computer.

`wki102_ver_minor`

Type: **DWORD**

The minor version number of the operating system running on the computer.

`wki102_lanroot`

Type: **LMSTR**

A pointer to a string that contains the path to the LANMAN directory.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wki102_logged_on_users`

Type: **DWORD**

The number of users who are logged on to the local computer.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaGetInfo

NetWkstaSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_INFO_502 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_INFO_502** structure is obsolete. The structure contains information about a workstation environment.

Syntax

```
typedef struct _WKSTA_INFO_502 {
  DWORD wki502_char_wait;
  DWORD wki502_collection_time;
  DWORD wki502_maximum_collection_count;
  DWORD wki502_keep_conn;
  DWORD wki502_max_cmds;
  DWORD wki502_sess_timeout;
  DWORD wki502_siz_char_buf;
  DWORD wki502_max_threads;
  DWORD wki502_lock_quota;
  DWORD wki502_lock_increment;
  DWORD wki502_lock_maximum;
  DWORD wki502_pipe_increment;
  DWORD wki502_pipe_maximum;
  DWORD wki502_cache_file_timeout;
  DWORD wki502_dormant_file_limit;
  DWORD wki502_read_ahead_throughput;
  DWORD wki502_num_mailslot_buffers;
  DWORD wki502_num_srv_announce_buffers;
  DWORD wki502_max_illegal_datagram_events;
  DWORD wki502_illegal_datagram_event_reset_frequency;
  BOOL  wki502_log_election_packets;
  BOOL  wki502_use_opportunistic_locking;
  BOOL  wki502_use_unlock_behind;
  BOOL  wki502_use_close_behind;
  BOOL  wki502_buf_named_pipes;
  BOOL  wki502_use_lock_read_unlock;
  BOOL  wki502_utilize_nt_caching;
  BOOL  wki502_use_raw_read;
  BOOL  wki502_use_raw_write;
  BOOL  wki502_use_write_raw_data;
  BOOL  wki502_use_encryption;
  BOOL  wki502_buf_files_deny_write;
  BOOL  wki502_buf_read_only_files;
  BOOL  wki502_force_core_create_mode;
  BOOL  wki502_use_512_byte_max_transfer;
} WKSTA_INFO_502, *PWKSTA_INFO_502, *LPWKSTA_INFO_502;
```

Members

`wki502_char_wait`

Type: **DWORD**

The number of seconds the computer waits for a remote resource to become available.

`wki502_collection_time`

Type: **DWORD**

The number of milliseconds the computer collects data before sending the data to a character device resource. The workstation waits the specified time or collects the number of characters specified by the **wki502_maximum_collection_count** member, whichever comes first.

`wki502_maximum_collection_count`

Type: **DWORD**

The number of bytes of information the computer collects before sending the data to a character device resource. The workstation collects the specified number of bytes or waits the period of time specified by the **wki502_collection_time** member, whichever comes first.

`wki502_keep_conn`

Type: **DWORD**

The number of seconds the server maintains an inactive connection to a server's resource.

`wki502_max_cmds`

Type: **DWORD**

The number of simultaneous network device driver commands that can be sent to the network.

`wki502_sess_timeout`

Type: **DWORD**

The number of seconds the server waits before disconnecting an inactive session.

`wki502_siz_char_buf`

Type: **DWORD**

The maximum size, in bytes, of a character pipe buffer and device buffer.

`wki502_max_threads`

Type: **DWORD**

The number of threads the computer can dedicate to the network.

`wki502_lock_quota`

Type: **DWORD**

Reserved.

`wki502_lock_increment`

Type: **DWORD**

Reserved.

`wki502_lock_maximum`

Type: **DWORD**

Reserved.

`wki502_pipe_increment`

Type: **DWORD**

Reserved.

`wki502_pipe_maximum`

Type: **DWORD**

Reserved.

`wki502_cache_file_timeout`

Type: **DWORD**

Reserved.

`wki502_dormant_file_limit`

Type: **DWORD**

Reserved.

`wki502_read_ahead_throughput`

Type: **DWORD**

Reserved.

`wki502_num_mailslot_buffers`

Type: **DWORD**

Reserved.

`wki502_num_srv_announce_buffers`

Type: **DWORD**

Reserved.

`wki502_max_illegal_datagram_events`

Type: **DWORD**

Reserved.

`wki502_illegal_datagram_event_reset_frequency`

Type: **DWORD**

Reserved.

`wki502_log_election_packets`

Type: **BOOL**

Reserved.

`wki502_use_opportunistic_locking`

Type: **BOOL**

Reserved.

`wki502_use_unlock_behind`

Type: **BOOL**

Reserved.

`wki502_use_close_behind`

Type: **BOOL**

Reserved.

`wki502_buf_named_pipes`

Type: **BOOL**

Reserved.

`wki502_use_lock_read_unlock`

Type: **BOOL**

Reserved.

`wki502_utilize_nt_caching`

Type: **BOOL**

Reserved.

`wki502_use_raw_read`

Type: **BOOL**

Reserved.

`wki502_use_raw_write`

Type: **BOOL**

Reserved.

`wki502_use_write_raw_data`

Type: **BOOL**

Reserved.

`wki502_use_encryption`

Type: **BOOL**

Reserved.

`wki502_buf_files_deny_write`

Type: **BOOL**

Reserved.

`wki502_buf_read_only_files`

Type: **BOOL**

Reserved.

`wki502_force_core_create_mode`

Type: **BOOL**

Reserved.

`wki502_use_512_byte_max_transfer`

Type: **BOOL**

Reserved.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaGetInfo

NetWkstaSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_TRANSPORT_INFO_0 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_TRANSPORT_INFO_0** structure contains information about the workstation transport protocol, such as Wide Area Network (WAN) or NetBIOS.

Syntax

```
typedef struct _WKSTA_TRANSPORT_INFO_0 {
  DWORD wkti0_quality_of_service;
  DWORD wkti0_number_of_vcs;
  LMSTR wkti0_transport_name;
  LMSTR wkti0_transport_address;
  BOOL  wkti0_wan_ish;
} WKSTA_TRANSPORT_INFO_0, *PWKSTA_TRANSPORT_INFO_0, *LPWKSTA_TRANSPORT_INFO_0;
```

Members

`wkti0_quality_of_service`

Specifies a value that determines the search order of the transport protocol with respect to other transport protocols. The highest value is searched first.

`wkti0_number_of_vcs`

Specifies the number of clients communicating with the server using this transport protocol.

`wkti0_transport_name`

Specifies the device name of the transport protocol.

`wkti0_transport_address`

Specifies the address of the server on this transport protocol.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wkti0_wan_ish`

This member is ignored by the NetWkstaTransportAdd function. For the NetWkstaTransportEnum function, this member indicates whether the transport protocol is a WAN transport protocol. This member is set to **TRUE** for NetBIOS/TCIP; it is set to **FALSE** for NetBEUI and NetBIOS/IPX.

Certain legacy networking protocols, including NetBEUI, will no longer be supported. For more information, see Network Protocol Support in Windows.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Header	lmwksta.h (include Lm.h)

See also

NetWkstaTransportAdd

NetWkstaTransportEnum

Network Management Overview

Network Management Structures

Server and Workstation Transport Functions

WKSTA_USER_INFO_0 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_USER_INFO_0** structure contains the name of the user on a specified workstation.

Syntax

```
typedef struct _WKSTA_USER_INFO_0 {
  LMSTR wkui0_username;
} WKSTA_USER_INFO_0, *PWKSTA_USER_INFO_0, *LPWKSTA_USER_INFO_0;
```

Members

`wkui0_username`

Specifies the name of the user currently logged on to the workstation.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaUserEnum

NetWkstaUserGetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_USER_INFO_1 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_USER_INFO_1** structure contains user information as it pertains to a specific workstation. The information includes the name of the current user and the domains accessed by the workstation.

Syntax

```
typedef struct _WKSTA_USER_INFO_1 {
  LMSTR wkui1_username;
  LMSTR wkui1_logon_domain;
  LMSTR wkui1_oth_domains;
  LMSTR wkui1_logon_server;
} WKSTA_USER_INFO_1, *PWKSTA_USER_INFO_1, *LPWKSTA_USER_INFO_1;
```

Members

`wkui1_username`

Specifies the name of the user currently logged on to the workstation.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wkui1_logon_domain`

Specifies the name of the domain in which the user is currently logged on.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wkui1_oth_domains`

Specifies the list of operating system domains browsed by the workstation. The domain names are separated by blanks.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

`wkui1_logon_server`

Specifies the name of the server that authenticated the user.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaUserEnum

NetWkstaUserGetInfo

NetWkstaUserSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions

WKSTA_USER_INFO_1101 structure (lmwksta.h)

2/1/2021 • 2 minutes to read • Edit Online

The **WKSTA_USER_INFO_1101** structure contains information about the domains accessed by a workstation.

Syntax

```
typedef struct _WKSTA_USER_INFO_1101 {
  LMSTR wkui1101_oth_domains;
} WKSTA_USER_INFO_1101, *PWKSTA_USER_INFO_1101, *LPWKSTA_USER_INFO_1101;
```

Members

`wkui1101_oth_domains`

Specifies the list of operating system domains browsed by the workstation. The domain names are separated by blanks.

This string is Unicode if **_WIN32_WINNT** or **FORCE_UNICODE** are defined.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	lmwksta.h (include Lm.h)

See also

NetWkstaUserGetInfo

NetWkstaUserSetInfo

Network Management Overview

Network Management Structures

Workstation and Workstation User Functions