yeswin32netyesTRUENg yesyesyesyesWin32 Network Help FileWin32netyesyes24/05/01

About WIL Extenders
Using WIL Extenders
Compiling with WIL Extenders

Contacting Wilson WindowWare
How to get Technical Support
About this Help File

AddExtender(filename)
LastError( )
Net101
NetInfo(requestcode)

Win32 Network Errors

netAddDrive(user-id, pswd, net-resource, local drive, persist)
netAddPrinter(user-id, pswd, net-resource, local device, persist)
netCancelCon(local drive, persist, forceflag)
netDirDialog(flag)
netGetCon(local drive)
netGetUser(netname)
netResources(net-resource, scope, type, usage)
netVersion( )

wntAccessAdd(server-name, resource/share-name, user-name, object-type, access-string)
wntAccessDel(server-name, resource/share-name, user-name, object-type)
wntAccessGet(server-name, resource/share-name, user-name, object-type)
wntAccessList
wntAcctInfo(server-name, account-name, request)
wntAcctPolGet(server-name, request)
wntAcctPolSet(server-name, request, value)
wntAddDrive(user-id, pswd, net-resource, local drive, persist)
wntAddPrinter(user-id, pswd, net-resource, local device, persist)
wntAuditAdd(server-name, resource/share-name, user/group name, object-type, access-string)
wntAuditDel(server-name, resource/share-name, user/group name, object-type)
wntAuditGet(server-name, resource/share-name, user/group name, object-type)
wntAuditList(server-name, resource/share-name, object-type, flag)
wntCancelCon(local drive, persist, forceflag)
wntChgPswd(server/domain, user-name, old-password, new-password)
wntCurrUsers(server-name, flags)
wntDirDialog(flag)
wntDomainSync(server-name, request)
wntEventWrite(server-name, source-name, type/category, event-id, description)
wntFileClose
wntFileUsers(server-name, file-pathname)
wntFilesOpen(server-name)
wntGetCon(local drive)
wntGetDc(server-name, domain-name, flag)
wntGetDrive(net-resource)
wntGetUser(netname)
wntGroupAdd(server-name, group-name, group-type, comment)
wntGroupDel(server-name, group-name, group-type)
wntGroupInfo(server-name, group, group-type, request)
wntListGroups(server-name, group-type)
wntMemberDel(server-name, group-name, user-name, group-type)

wntMemberGet(server-name, group-name, user-name,group-type)
wntMemberGrps(server-name, user-name, group-type, flags)
wntMemberList(server-name, group-name, group-type [,get-domain-flag])
wntMemberSet(server-name, group-name, user-name, group-type)
wntOwnerGet(server-name, reg-key, resource-name, object-type, flag)
wntOwnerSet(server-name, reg-key, resource-name, object-type, user/group name)
wntPrivAdd
wntPrivDel
wntPrivGet(server-name, user/group name, privilege)
wntPrivList(server-name, user/group name)
wntRasUserGet (server-name, user-name, request)
wntRasUserSet(server-name, user-name, privilege, phone-number)
wntRemoteTime(server-name, format)
wntResources(net-resource, scope, type, usage)
wntRunAsUser
wntServerList(server-name, domain-name, server-type)
wntServerType(server-name)
wntServiceAt(server, domain, server-type, service-name, flags)
wntServiceInf (server-name)
wntShareAdd(server-name, resource, share-name,share-type, max-users)
wntShareDel(server-name, resource/share-name, share-type)
wntShareInfo(server-name, resource/share-name, share-type, request)
wntShareSet(server-name, resource/share-name, share-type, comment, description)
wntShareUsers(server-name, share/computer-name, share-type, format, flags)
wntShutdown(computer-name, message, timeout, force, reboot)
wntSvcCfgGet(server, service-name, flags, request)
wntSvcCfgSet(server, service-name, flags, request, value)
wntSvcControl(server-name, service-name, flags, control-code)
wntSvcCreate(server-name, string-values, numeric-values, dependencies, reserved)
wntSvcDelete(server, service-name, flags)
wntSvcStart(server, service-name, flags, params, delimiter)
wntSvcStatus(server, service-name, flags, request)
wntUserAdd(server-name)
wntUserAddDat(element, value)
wntUserDel(server-name, user-name)
wntUserExist(server-name, user-name)
wntUserGetDat(server-name, user-name, element)
wntUserInfo(request)
wntUserProps
wntUserRename
wntUserSetDat
wntVersion( )
wntServerInfo(server-name, request)
wntShareList(server-name, share-type, flags)
wntUserList(server-name, account-type)
wntWtsUserGet
wntWtsUserSet

w9xAccessAdd
w9xAcceesDel
w9xAccessGet
w9xAccessList
w9xGroupAdd
w9xGroupDel
w9xGroupInfo
w9xListGroups
w9xMemberDel
w9xMemberGet

w9xMemberGrps
w9xMemberList
w9xMemberSet
w9xOwnerGet(server-name, reg-key, resource-name, object-type, flag)
w9xServiceAt
w9xServerList(server, domain, server-type)
w9xShareAdd
w9xShareDel
w9xShareInfo
w9xShareSet
w9xUserAdd
w9xUserAddDat
w9xUserDel
w9xUserExist
w9xUserGetDat
w9xUserInfo
w9xUserList
w9xUserProps
w9xUserRename
w9xUserSetDat
w9xFileUsers
w9xGetDc(server-name, domain-name, flag)
w9xVersion

w9xRemoteTime(server-name, format)
wntGroupEdit(server-name, group, group-type, request, value)
wntLsaPolGet
wntLsaPolSet
wntUserSidChk
wntEventLog
wntGroupRen
wntResources2
wntAccessMod
wntAuditMod

w95AccessDel(server-name, resource, user-name)
w95AccessAdd(server-name, resource, user-name, access-rights, flags)
w95AddDrive(user-id, pswd, net-resource, local-drive, persist)
w95AddPrinter(user-id, pswd, net-resource, local device, persist)
w95CancelCon(local drive, persist, forceflag)
w95DirDialog(flag)
w95FileClose
w95FileUsers(server-name, file-pathname)
w95GetCon(local name)
w95GetDrive(net-resource)
w95GetUser(netname)
w95Resources(net-resource, scope, type, usage)
w95ServerInfo(server-name, request)
w95ServerType
w95ServiceAt
w95ServiceInf
w95ShareAdd(server-name, resource, share-name,share-type, flags)
w95ShareDel(server-name, share-name)

w95ShareInfo(server-name, share-name, request)
w95ShareList(server-name, share-type, flags)
w95ShareSet(server-name, share-name, comment, full-password, read-password)
w95ShareUsers(server-name, share/computer-name, share-type, format, flags)
w95Version( )

Win32 Basic Functions
Windows NT Functions
Windows 95 Functions
Windows 95/RADMIN Functions

Win32 Network Extenders

Suggested Approaches
Determining your platform

Win 32 Network Extender Help File

Windows 95/98 Functions

For use on Windows 95/98/Me workstations. Can control Windows 95/98/Me servers.

Windows NT/2000 Functions

For use on Windows NT workstations. Can control Windows NT servers.

Windows 9x Functions

For use on Windows 95/98/Me workstations. Can control Windows NT servers. May be used in addition to the standard Windows 95/98/Me extender.

Windows Win32 Functions

General feature-limited extender. For backwards compatibility the functions are still supported.

About WIL Extenders
Using WIL Extenders
Compiling with WIL Extenders

Win32 Network Extenders
Suggested Approaches
Determining your platform
Errors

Contacting Wilson WindowWare
How to get Technical Support
About this Help File

These Windows Interface Language Network extenders provide standard support for computers running 32 bit versions of Windows, such as Windows NT and Windows 95/98/Me. They may be used in conjunction with other 32 bit Intel extenders.

The Win32 Network extenders

have been re-designed for the specific 32 bit operating systems. This might result in some minor changes, depending on how you choose to proceed with new scripts.

There are now separate extenders and additional functions for the following:

Windows 95

For use on Windows 95/98/Me workstations. Can control Windows 95/98/Me servers.

Note: All the functions have kept the same naming convention of w95[ ], however, they are also designed to run on Windows 98.

Windows NT.

For use on Windows NT workstations. Can control Windows NT servers.

Windows 95/RADMIN (9X).

For use on Windows 95/98/Me workstations. Can control Windows NT servers. May be used in addition to the standard Windows 95/98/Me extender.

Note: In order to use the functions in this extender, you must have an NT domain server, and you must have RADMIN32.DLL and RLOCAL32.DLL present. These DLL's

can be found in the Microsoft Windows 95 Service Pack 1. (location on disk admin\tools\servmgmt\)

Win32 Basic functions

General feature-limited extender. For backwards compatibility the functions are still supported.

These functions can be used within WIL scripts or can be compiled into WIL executables

Technical Support

is available for registered users. If you can't find what you're looking for, or you're having problems with your WIL scripts, be sure to look at the suggested approaches section, which will answer questions on the best direction to proceed in your development of 32 bit Networking scripts.

WIN 32 NETWORK EXTENDER HELP FILE

About WIL Extenders

WIL extender Dlls are special Dlls designed to extend the built-in function set of the WIL processor. These Dlls typically add functions not provided in the basic WIL set, such as network commands for particular networks (Novell, Windows for WorkGroups, LAN Manager and others), MAPI, TAPI, and other important Application Program Interface functions as may be defined by the various players in the computer industry from time to time. These Dlls may also include custom built function libraries either by the original authors, or by independent third party developers. (An Extender SDK is available). Custom extender Dlls may add nearly any sort of function to the WIL language, from the mundane network math or database extensions, to items that can control fancy peripherals, including laboratory or manufacturing equipment.

WIL extenders must be installed separately. Up to 10 extender Dlls may be added. The total number of added items may not exceed 200 functions and constants. The **AddExtender** function must be executed before attempting to use any functions in the extender library. The AddExtender function should be only executed once in each WIL script that requires it.

- **Using WIL Extenders**
- **Compiling with WIL Extenders**

WIN 32 NETWORK EXTENDER HELP FILE

Using WIL Extenders

- About WIL Extenders
- Compiling with WIL Extenders

Accessing the additional functionality available in WIL Extender Dlls is simple. At the top of each script in which WIL Extender commands are to be used add the appropriate extender with the **AddExtender** command.

```
AddExtender(extender filename)
```

The WIL interpreter will search for the the extender DLL's. If no path is specified in the **AddExtender** statement, the WIL interpreter will search the current directory, the windows directory and on directories on the path. The extender DLL's must be available or the **AddExtender** line will return an error. In general, when you run a large exe with embedded extenders, it will extract the extenders to the same directory the compiled exe is in.

The **AddExtender** function should only be executed once for each extender dll in each WIL script that requires it. Per script you can add up to 10 extender Dlls or a combined total of 200 functions.

For example to use both of the Win32 Network Extenders, two AddExtender lines would appear in the script.

```
AddExtender("WWW9534i.DLL")    ;for Windows 95/98/Me
AddExtender("WWWNT34i.DLL")    ;for Windows NT
```

WIN 32 NETWORK EXTENDER HELP FILE

Compiling with WIL Extenders

- About WIL Extenders
- Using WIL Extenders

The WinBatch+Compiler has two options for compiling scripts into executables, Large EXE for Standalone PC's and Small EXE for Networked PC's. When any extender functions are used in a script, the corresponding extender must be compiled into the executable, or placed where the executable can access it.

The Large Standalone EXE option of the Compiler has an additional button. The EXTENDERS button displays a list of extenders which can be chosen and compiled into a Standalone EXE option. More than one extender may be chosen. When a Standalone EXE is launched on a PC it looks for the necessary DLL's in the current directory, on the path and in the Windows directory. If the DLL's are not found, they are automatically written into the current directory. If for some reason, they cannot be written to that directory (perhaps the directory is set to be Read Only), the large compiled file will not run.

The DLLs can also be copied into a directory on a computers PATH and the compiled EXE will find them there and run. The Compiler has a Small EXE for Networked PC's option that takes advantage of this.

The DLLs need to be placed on the PATH only once. Subsequent EXE files installed on this same machine can be compiled under the Small EXE option.

It is generally recommended to use the filename without including the path in an AddExtender statement. However, if it is preferable to point to the current directory, (directory in which the WIL executable resides), it can be done.

```
AddExtender( StrCat( DirHome( ), "wwn3x34i.dll" ))
```

WIN 32 NETWORK EXTENDER HELP FILE

Contacting Wilson WindowWare

Wilson WindowWare, Inc.
5421 California Ave. SW
Seattle, WA 98136 USA

Orders: (800) 762-8383
Voice: (206) 938-1740
Fax: (206) 935-7129

Email: info@windowware.com

Registered users of our software get manuals, technical support, use of Wilson WindowWare on-line information services, and special offers on new versions of Wilson WindowWare products.

- Technical Support
- About WIL Extenders

How to get Technical Support

The Wilson WindowWare website is an excellent technical resource. Access to the entire Technical Support Database is at your fingertips. In the Technical Support area use the keyword search to find answers to common problems, alternate scripting methods, and sample code. Or join the Wilson WindowWare Web BBS, a new Web forum. The BBS provides an outlet for registered users to share their experiences with other users.

See the information on registering your copy (found in the WinBatch.hlp file) if you haven't done so yet.

The latest versions of our software are available on-line. The places here may change at any time -- check your installation sheet for the most recent addresses.

Internet Web page: http://www.windowware.com

Internet Technical Support Articles & Web BBS: http://techsupt.windowware.com

Internet FTP: ftp.windowware.com in /wwwftp/wilson

- **Contacting Wilson WindowWare**
- **About WIL Extenders**

WIN 32 NETWORK EXTENDER HELP FILE

About this Help File

This extender adds certain network capability to the Windows Interface Language (WIL) processing engine. Please refer to the **WIL Reference Manual** for an introduction to WIL, as well as for complete documentation of the many functions available in WIL and the programs that use it. This help file includes only topics and functions which are exclusive to this particular WIL Extender.

- About WIL Extenders
- Using WIL Extenders
- Compiling with WIL Extenders

Notational Conventions

Throughout this manual, we use the following conventions to distinguish elements of text:

ALL-CAPS
: Used for filenames.

Boldface
: Used for important points, programs, function names, and parts of syntax that must appear as shown.

`system`
: Used for items in menus and dialogs, as they appear to the user.

`Small fixed-width`
: Used for WIL sample code.

Italics
: Used for emphasis, and to liven up the documentation just a bit.

Acknowledgments

This network extender developed by Morrie Wilson and Richard Merit.

Documentation written by Tina Browning,and Deana Dahley.

WIN 32 NETWORK EXTENDER HELP FILE

Win32 Network Extenders

These Windows Interface Language Network extenders provide standard support for computers running 32 bit versions of Windows, such as Windows NT and Windows 95/98/Me.

The Windows 32 Network extender (WWNET34i.DLL) has been split into separate extenders for Windows 95/98/Me (WWW9534i.DLL) and Windows NT (WWWNT34i.DLL).

For Windows 95/98/Me use…
AddExtender("WWW9534i.DLL")
Other required DLL's: none

For Windows 95/98/Me to NT server use
AddExtender("WWW9X34i.DLL")
Other required DLL's: RADMIN32.DLL and RLOCAL.DLL

For Windows NT use…
AddExtender("WWWNT34i.DLL")
Other required DLL's: none

The WWNET34i.DLL, Win32 Basic Extender DLL, which supports the existing net[..] functions will still be distributed for backwards compatibility.

However, this DLL is a "wrapper" or "shell" which calls the appropriate platform-specific extender and must be used in combination with WWW9534i.DLL or WWWNT34i.DLL. One of these DLL's (depending on the operating system being used) will need to be available, either in the current directory, in the same directory as WWNET34i.DLL, or in a directory on the path.

The Win32 Basic extender remains the same.
AddExtender("WWNET34i.DLL")

This extender requires either WWW9534i.DLL or WWWNT34i.DLL, depending on the operating system.

The existing net[..] functions have been renamed to w95[..] and wnt[..], respectively: Their parameters have not changed.

- Windows 95 Functions
- Windows 9x Functions
- Windows NT Functions
- Win32 Basic Functions
- Determining your platform
- About WIL Extenders
- Using WIL Extenders
- Compiling with WIL Extenders

Old Windows 32	New Windows 95	New Windows NT	New Windows 9x
netAddDrive	w95AccessDel	wntAccessAdd	w9xAccessAdd
netAddPrinter	w95AccessAdd	wntAccessDel	w9xAccessDel
netCancelCon	w95AddDrive	wntAccessGet	w9xAccessGet
netDirDialog	w95AddPrinter	wntAccessList	w9xAccessList
netGetCon	w95CancelCon	wntAcctInfo	w9xGroupAdd
netGetUser	w95DirDialog	wntAcctPolGet	w9xGroupDel
netResources	w95FileClose	wntAcctPolSet	w9xGroupInfo
netVersion	w95FileUsers	wntAddPrinter	w9xListGroups
	w95GetCon	wntAuditAdd	w9xMemberDel
	w95GetDrive	wntAuditDel	w9xMemberGet
	w95GetUser	wntAuditGet	w9xMemberGrps
	w95Resources	wntAuditList	w9xMemberList
	w95ServiceAt	wntCancelCon	w9xMemberSet
	w95ServiceInf	wntChgPswd	w9xOwnerGet
	w95ServerType	wntCurrUsers	w9xRemoteTime
	w95ShareAdd	wntDirDialog	w9xServerList
	w95ShareDel	wntDomainSync	w9xServiceAt
	w95ShareInfo	wntEventLog	w9xShareAdd
	w95ShareSet	wntEventWrite	w9xShareDel
	w95ShareUsers	wntFileClose	w9xShareInfo
	w95Version	wntFilesOpen	w9xShareSet
		wntGetCon	w9xUserAdd
		wntGetDC	w9xUserAddDat
		wntGetDrive	w9xUserDel
		wntGetUser	w9xUserExist
		wntFileUsers	w9xUserGetDat
		wntGroupAdd	w9xUserList
		wntGroupDel	w9xUserInfo
		wntGroupEdit	w9xUserProps
		wntGroupInfo	w9xUserRename
		wntGroupRen	w9xUserSetDat
		wntListGroups	w9xVersion
		wntLsaPolGet	
		wntLsaPolSet	
		wntMemberDel	
		wntMemberGet	
		wntMemberGrps	
		wntMemberList	
		wntMemberSet	
		wntOwnerGet	
		wntOwnerSet	
		wntPrivAdd	
		wntPrivDel	
		wntPrivGet	
		wntPrivList	
		wntRasUserGet	

wntRasUserSet
wntRemoteTime
wntResources
wntResources2
wntRunAsUser
wntServerList
wntServiceAt
wntServiceInf
wntShareAdd
wntShareDel
wntShareInfo
wntShareList
wntShareSet
wntShareUsers
wntShutdown
wntSvcCfgGet
wntSvcCfgSet
wntSvcControl
wntSvcCreate
wntSvcDelete
wntSvcStart
wntSvcStatus
wntUserAdd
wntUserAddDat
wntUserDel
wntUserExist
wntUserGetDat
wntUserInfo
wntUserList
wntUserProps
wntuserRename
wntUserSetDat
wntUserSidChk
wntVersion
wntWtsUserGet
wntWtsUserSet

How do these changes affect my scripts?

WIN 32 NETWORK EXTENDER HELP FILE

Suggested Approaches

The additional extenders add a slight twist to the mix. Here are our suggestions for how to proceed.

New Scripts

If you're writing a new script, use a new platform specific extender with the new function names, **w95AddDrive** or **wntAddDrive** instead of **NetAddDrive**.

AddExtender("WWW9534i.DLL")
for Windows 95/98/Me

or…

AddExtender("WWWNT34i.DLL")
for Windows NT.

or…

AddExtender("WWW9X34i.DLL")
For Windows 95/98/Me and NT Servers.

If you don't know which platform you will be running on, you can add code to your script to **determine the platform** and conditionally execute the corresponding functions. On the other hand, you might choose to use the generic functions in the Win32 Basic Extender. In which case, all three of the Win32 extenders will need to be provided.

Existing scripts

Scripts which contain old style **net[ ]** functions will require special consideration.

If you're working with uncompiled WIL scripts, you have two options.

 # 1

Replace the first three characters of the net[ ] functions with w95 or wnt and use the new platform specific extender. The function parameters are the same and will be unaffected.

NetAddDrive becomes either **w95AddDrive** or

- **Windows 95/98/Me Functions**
- **Windows 9x Functions**
- **Windows NT Functions**
- **Win32 Basic Functions**
- **Determining your platform**
- **About WIL Extenders**
- **Using WIL Extenders**
- **Compiling with WIL Extenders**

wntAddDrive.

 # 2

If you prefer to use the old net[ ] functions, **NetAddDrive** etc., they are still supported by the Win32 Basic extender, WWNET34i.DLL. As long as this extender is available, old functions like **netAddDrive** and **netCancelCon** will work exactly as they did before.

The only catch is…

You must remember to include the platform specific extender.

"WWW9534i.DLL" for Windows 95/98/Me

"WWWNT34i.DLL" for Windows NT

Compiled Exe's

Scripts compiled with an older version of the WWNET34i.DLL will still work as long as you do not replace the dll.

For new executables, the extenders can be selected as usual from the Extender button in the Large for Standalone Exe's option or placed on the path or search drive for Small Exe's.

As usual, when a Large executable is launched it will look for copies of the DLL's. If an older version of WWNET34i.DLL is found in the current directory, on the path or in the Windows directory, it will not be replaced. This will not affect new scripts using old net[ ] commands.

However, if the Win32 Basic extender dll is manually replaced with the new version, old scripts will fail. Remember, the new WWNET34i.DLL points to the Windows 95/98/Me or Windows NT Extender DLL. It can no longer run on its own.

•

Determining your platform

If you need to write scripts for both Windows NT and Windows 95/98/Me, your script can decide on the fly which platform you are using.

You'll need to have two sections of code. One with wnt[ ] functions and one with w95[ ] functions. The functions differ only in the first three characters. For example, **wntAddDrive** has the same exact parameters as **w95AddDrive**. Simply copy your code and replace the first three characters.

When you compile and distribute the code, remember to include both the Windows 95/98/Me Network extender, WWW9534i.DLL and the Windows NT extender, WWWNT34i.DLL.

- Windows 95/98/Me Functions
- Windows 9x Functions
- Windows NT Functions
- Win32 Basic Functions
- About WIL Extenders
- Using WIL Extenders
- Compiling with WIL Extenders

Example:

```
;Verify version of Windows so that we know what we are doing
;Returns from WinMetrics
;0=??? 1=Win 2=Win4Wkg 3=Win32S  4=WinNT 5=Win95
;;
;;
platform=WinMetrics(-4)
if platform == 4
  gosub WinNTdo
else
  if platform == 5
     gosub Win95do
  else
     message("ooops", "WinMetrics returning %platform%")
  endif
endif
exit

:WinNTdo
 AddExtender("WWWNT34i.DLL")

 return
```

```
:Win95do
 AddExtender("WWW9534i.DLL")

 return
```

•

WIN 32 BASIC FUNCTIONS

The following WIL functions are useful when using Network extenders.

AddExtender(filename)
LastError( )
Net101
NetInfo(requestcode)

The following generic functions for Windows 95/98/Me and NT networks can be added with the WWNET34i.DLL.

Additionally, the platform specific extender DLLs (WWW9534i.DLL for Windows 95/98/Me and WWWNT34i.DLL for Windows NT) are required.

For Windows 95/98/Me workstations accessing an NT Server use the 9x extender WWW9X34i.DLL. Requires addtional DLL's: RADMIN.DLL
and RLOCAL.DLL. These DLL's can be found in the Microsoft Windows 95 Service Pack 1.

- **Windows 95/98/Me Functions**
- **Windows 9x Functions**
- **Windows NT Functions**
- **Errors**
- **Using WIL Extenders**
- **Compiling with WIL Extenders**

netAddDrive(user-id, pswd, net-resource, local drive, persist)
netAddPrinter(user-id, pswd, net-resource, local device, persist)
netCancelCon(local drive, persist, forceflag)
netDirDialog(flag)
netGetCon(local drive)
netGetUser(netname)
netResources(net-resource, scope, type, usage)
netVersion( )

•

WINDOWS 95/98 FUNCTIONS

The following WIL functions are useful when using Network extenders.

AddExtender(filename)
LastError( )
Net101
NetInfo(requestcode)

- **Win32 Basic Functions**
- **Windows NT Functions**
- **Windows 9x Functions**
- **Errors**
- **Using WIL Extenders**
- **Compiling with WIL Extenders**

The following functions for Windows 95 & Windows 98 networking can be added with the WWW9534i.DLL.

Disk and Printer Control

w95AddDrive(user-id, pswd, net-resource, local-drive, persist)
w95AddPrinter(user-id, pswd, net-resource, local device, persist)
w95CancelCon(local drive, persist, forceflag)
w95DirDialog(flag)
w95GetCon(local name)
w95GetDrive(net-resource)
w95ShareAdd(server-name, resource, share-name,share-type, flags)
w95ShareInfo(server-name, share-name, request)
w95ShareList(server-name, share-type, flags)
w95ShareSet(server-name, share-name, comment, full-password, read-password)
w95ShareUsers(server-name, share/computer-name, share-type, format, flags)
w95ShareDel(server-name, share-name)

Miscellaneous

w95FileClose(Server-name, path-name)
w95FileUsers(server-name, file-pathname)
w95Resources(net-resource, scope, type, usage)
w95Version( )

NT Security

w95AccessAdd(server-name, resource, user-name, access-rights, flags)

w95AccessDel(server-name, resource, user-name)

Server Info

w95ServerInfo(server-name, request)
w95ServerType(server-name)
w95ServiceAt(server, domain, server-type, service-name, flags)
w95ServiceInf(server-name)

User Account Administration

w95GetUser(netname)

Complete List: Alphabetical Order

w95AccessAdd(server-name, resource, user-name, access-rights, flags)
w95AccessDel(server-name, resource, user-name)
w95AddDrive(user-id, pswd, net-resource, local-drive, persist)
w95AddPrinter(user-id, pswd, net-resource, local device, persist)
w95CancelCon(local drive, persist, forceflag)
w95DirDialog(flag)
w95FileClose(Server-name, path-name)
w95FileUsers(server-name, file-pathname)
w95GetCon(local name)
w95GetDrive(net-resource)
w95GetUser(netname)
w95Resources(net-resource, scope, type, usage)
w95ServerInfo(server-name, request)
w95ServerType(server-name)
w95ServiceAt(server, domain, server-type, service-name, flags)
w95ServiceInf(server-name)
w95ShareAdd(server-name, resource, share-name,share-type, flags)
w95ShareDel(server-name, share-name)
w95ShareInfo(server-name, share-name, request)
w95ShareList(server-name, share-type, flags)
w95ShareSet(server-name, share-name, comment, full-password, read-password)
w95ShareUsers(server-name, share/computer-name, share-type, format, flags)
w95Version( )

•

WINDOWS NT FUNCTIONS

The following WIL functions are useful when using Network extenders.

AddExtender(filename)
LastError( )
Net101
NetInfo(requestcode)

- **Win32 Basic Functions**
- **Windows 95/98/Me Functions**
- **Windows 9x Functions**
- **Errors**
- **Using WIL Extenders**
- **Compiling with WIL Extenders**

The following functions for Windows NT networking can be added with the WWWNT34i.DLL.

Disk and Printer Control

wntAddDrive(user-id, pswd, net-resource, local drive, persist)
wntAddPrinter(user-id, pswd, net-resource, local device, persist)
wntCancelCon(local drive, persist, forceflag)
wntDirDialog(flag)
wntGetCon(local drive)
wntGetDrive(net-resource)
wntShareAdd(server-name, resource, share-name,share-type, max-users)
wntShareDel(server-name, resource/share-name, share-type)
wntShareInfo(server-name, resource/share-name, share-type, request)
wntShareList(server-name, share-type, flags)
wntShareSet(server-name, resource/share-name, share-type, comment, description)
wntShareUsers(server-name, share/computer-name, share-type, format, flags)

Miscellaneous

wntDomainSync(server-name, request)
wntEventLog(server-name, text)
wntEventWrite(server-name, source-name, type/category, event-id, description)
wntFileClose(server-name, file-pathname)
wntFilesOpen(server-name)
wntFileUsers(server-name, file-pathname)
wntResources(net-resource, scope, type, usage)
wntResources2(net-resource, scope, type, usage, provider)

wntRunAsUser(domain/server, user-name, password, login-type, flags)
wntShutdown(computer-name, message, timeout, force, reboot)
wntVersion( )

NT Security

wntAccessAdd(server-name, resource/share-name, user-name, object-type, access-string)
wntAccessDel(server-name, resource/share-name, user-name, object-type)
wntAccessGet(server-name, resource/share-name, user-name, object-type)
wntAccessList(server-name, resource/share-name, object-type, flags)
wntAccessMod(server-name, resource, object-type, request, flag)
wntAuditAdd(server-name, resource/share-name, user/group name, object-type, access-string)
wntAuditDel(server-name, resource/share-name, user/group name, object-type)
wntAuditGet(server-name, resource/share-name, user/group name, object-type)
wntAuditList(s:server-name, s:resource/share-name, i:object-type, i:flag)
wntAuditMod(server-name, resource, object-type, request, flag)
wntChgPswd(server/domain, user-name, old-password, new-password)

Owner Object Info

wntOwnerGet(server-name, reg-key, resource-name, object-type, flag)
wntOwnerSet(server-name, reg-key, resource-name, object-type, user/group name)

Server Info

wntGetDc(server-name, domain-name, flag)
wntRemoteTime(server-name, format)
wntServerInfo(server-name, request)
wntServerList(server-name, domain-name, server-type)
wntServerType(server-name)
wntServiceAt(server, domain, server-type, service-name, flags)
wntServiceInf(server-name)
wntSvcCfgGet(server, service-name, flags, request)
wntSvcCfgSet(server, service-name, flags, request, value)
wntSvcControl(server, service-name, flags, control-code)
wntSvcCreate(server-name, string-values, numeric-values, dependencies, reserved)
wntSvcDelete(server, service-name, flags)
wntSvcStart(server, service-name, flags, params, delimiter)
wntSvcStatus(server, service-name, flags, request)

User Account Administration

wntAcctInfo(server-name, account-name, request)
wntAcctPolGet(server-name, request)
wntAcctPolSet(server-name, request, value)
wntCurrUsers(server-name, flags)

wntGetUser(netname)
wntGroupAdd(server-name, group-name, group-type, comment)
wntGroupDel(server-name, group-name, group-type)
wntGroupEdit(server-name, group, group-type, request, value)
wntGroupInfo(server-name, group, group-type, request)
wntGroupRen(server-name, group, group-type, new-name)
wntListGroups(server-name, group-type)
wntLsaPolGet(server-name, class, element)
wntLsaPolSet(server-name, class, element, value)
wntMemberDel(server-name, group-name, user-name, group-type)
wntMemberGet(server-name, group-name, user-name, group-type)
wntMemberGrps(server-name, user-name, group-type, flags)
wntMemberList(server-name, group-name, group-type[,get-domain-flag])
wntMemberSet(server-name, group-name, user-name, group-type)
wntPrivAdd(server-name, user/group name, privilege)
wntPrivDel(server-name, user/group name, privilege)
wntPrivGet(server-name, user/group name, privilege)
wntPrivList(server-name, user/group name)
wntRasUserGet(server-name, user-name, request)
wntRasUserSet(server-name, user-name, privilege, phone-number)
wntUserAdd(server-name)
wntUserAddDat(element, value)
wntUserDel(server-name, user-name)
wntUserExist(server-name, user-name)
wntUserGetDat(server-name, user-name, element)
wntUserInfo(request)
wntUserList(server-name, account-type)
wntUserProps(server-name, user-name, request)
wntUserSetDat(server-name, user-name, element, value)
wntUserSidChk(request, SID, reserved-1, reserved-2, reserved-3)
wntUserRename(server-name, old-username, new-username)

NT Terminal Server

wntWtsUserGet(server-name, user-name, request)
wntWtsUserSet(server-name, user-name, request, value)

Complete List: Alphbetical Order

wntAccessAdd(server-name, resource/share-name, user-name, object-type, access-string)
wntAccessDel(server-name, resource/share-name, user-name, object-type)
wntAccessGet(server-name, resource/share-name, user-name, object-type)
wntAccessList(server-name, resource/share-name, object-type, flags)
wntAccessMod(server-name, resource, object-type, request, flag)
wntAcctInfo(server-name, account-name, request)
wntAcctPolGet(server-name, request)
wntAcctPolSet(server-name, request, value)
wntAddDrive(user-id, pswd, net-resource, local drive, persist)
wntAddPrinter(user-id, pswd, net-resource, local device, persist)

wntAuditAdd(server-name, resource/share-name, user/group name, object-type, access-string)
wntAuditDel(server-name, resource/share-name, user/group name, object-type)
wntAuditGet(server-name, resource/share-name, user/group name, object-type)
wntAuditList(s:server-name, s:resource/share-name, i:object-type, i:flag)
wntAuditMod(server-name, resource, object-type, request, flag)
wntCancelCon(local drive, persist, forceflag)
wntChgPswd(server/domain, user-name, old-password, new-password)
wntCurrUsers(server-name, flags)
wntDirDialog(flag)
wntDomainSync(server-name, request)
wntEventLog(server-name, text)
wntEventWrite(server-name, source-name, type/category, event-id, description)
wntFileClose(server-name, file-pathname)
wntFilesOpen(server-name)
wntFileUsers(server-name, file-pathname)
wntGetCon(local drive)
wntGetDc(server-name, domain-name, flag)
wntGetDrive(net-resource)
wntGetUser(netname)
wntGroupAdd(server-name, group-name, group-type, comment)
wntGroupDel(server-name, group-name, group-type)
wntGroupEdit(server-name, group, group-type, request, value)
wntGroupInfo(server-name, group, group-type, request)
wntGroupRen(server-name, group, group-type, new-name)
wntListGroups(server-name, group-type)
wntLsaPolGet(server-name, class, element)
wntLsaPolSet(server-name, class, element, value)
wntMemberDel(server-name, group-name, user-name, group-type)
wntMemberGet(server-name, group-name, user-name, group-type)
wntMemberGrps(server-name, user-name, group-type, flags)
wntMemberList(server-name, group-name, group-type [,get-domain-flag])
wntMemberSet(server-name, group-name, user-name, group-type)
wntOwnerGet(server-name, reg-key, resource-name, object-type, flag)
wntOwnerSet(server-name, reg-key, resource-name, object-type, user/group name)
wntPrivAdd(server-name, user/group name, privilege)
wntPrivDel(server-name, user/group name, privilege)
wntPrivGet(server-name, user/group name, privilege)
wntPrivList(server-name, user/group name)
wntRasUserGet(server-name, user-name, request)
wntRasUserSet(server-name, user-name, privilege, phone-number)
wntRemoteTime(server-name, format)
wntResources(net-resource, scope, type, usage)
wntResources2(net-resource, scope, type, usage, provider)
wntRunAsUser(domain/server, user-name, password, login-type, flags)
wntServerInfo(server-name, request)
wntServerList(server-name, domain-name, server-type)
wntServerType(server-name)
wntServiceAt(server, domain, server-type, service-name, flags)

wntServiceInf(server-name)
wntShareAdd(server-name, resource, share-name,share-type, max-users)
wntShareDel(server-name, resource/share-name, share-type)
wntShareinfo(server-name, resource/share-name, share-type, request)
wntShareList(server-name, share-type, flags)
wntShareSet(server-name, resource/share-name, share-type, comment, description)
wntShareUsers(server-name, share/computer-name, share-type, format, flags)
wntShutdown(computer-name, message, timeout, force, reboot)
wntSvcCfgGet(server, service-name, flags, request)
wntSvcCfgSet(server, service-name, flags, request, value)
wntSvcControl(server, service-name, flags, control-code)
wntSvcCreate(server-name, string-values, numeric-values, dependencies, reserved)
wntSvcDelete(server, service-name, flags)
wntSvcStart(server, service-name, flags, params, delimiter)
wntSvcStatus(server, service-name, flags, request)
wntUserAdd(server-name)
wntUserAddDat(element, value)
wntUserDel(server-name, user-name)
wntUserExist(server-name, user-name)
wntUserGetDat(server-name, user-name, element)
wntUserInfo(request)
wntUserList(server-name, account-type)
wntUserProps(server-name, user-name, request)
wntUserSetDat(server-name, user-name, element, value)
wntUserSidChk(request, SID, reserved-1, reserved-2, reserved-3)
wntUserRename(server-name, old-username, new-username)
wntVersion( )
wntWtsUserGet(server-name, user-name, request)
wntWtsUserSet(server-name, user-name, request, value)

•

WINDOWS 9X FUNCTIONS

The following WIL functions are useful when using Network extenders.

AddExtender(filename)
LastError( )
Net101
NetInfo(requestcode)

- **Win32 Basic Functions**
- **Windows 95/98/Me Functions**
- **Errors**
- **Using WIL Extenders**
- **Compiling with WIL Extenders**

The following functions for Windows NT networking can be added with the WWW9x34i.DLL.

Disk and Printer Control

w9xShareAdd(server-name, resource, share-name, share-type, max-users)
w9xShareDel(server-name, resource/share-name, share-type)
w9xShareInfo(server-name, resource/share-name, share-type, request)
w9xShareSet(server-name, resource/share-name, share-type, comment, s:location)

Miscellaneous

w9xVersion()
w9xFileUsers(server-name, file-pathname)

NT Security

w9xAccessAdd(server-name, resource/share-name, user/group name, object-type, access-string)
w9xAccessDel(server-name, resource/share-name, user/group name, object-type)
w9xAccessGet(server-name, resource/share-name, user/group name, object-type)
w9xAccessList(server-name, resource/share-name, object-type, flags)

Owner Object Info

w9xOwnerGet(server-name, reg-key, resource-name, object-type, flag)

Server Info

w9xGetDc(server-name, domain-name, flag)
w9xRemoteTime(server-name, format)
w9xServerList(server, domain, server-type)
w9xServiceAt(server, domain, server-type, service-name, flags)

User Account Administration

w9xUserAdd(server-name)
w9xUserAddDat(element, value)
w9xUserDel(server-name, user-name)
w9xUserExist(server-name, user-name)
w9xUserGetDat(server-name, user-name, element)
w9xUserInfo(request)
w9xUserList(server-name, account-type)
w9xUserProps(server-name, user-name, request)
w9xUserRename(server-name, old-username, new-username)
w9xUserSetDat(server-name, user-name, element, value)
w9xGroupAdd(server-name, group-name, group-type, comment)
w9xGroupDel(server-name, group-name, group-type)
w9xGroupInfo(server-name, group, group-type, request)
w9xListGroups(server-name, group-type)
w9xMemberDel(server-name, group-name, user-name, group-type)
w9xMemberGet(server-name, group-name, user-name, group-type)
w9xMemberGrps(server-name, user-name, group-type, flags)
w9xMemberList(server-name, group-name, group-type)
w9xMemberSet(server-name, group-name, user-name, group-type)

Complete List: Alphbetical Order

w9xAccessAdd(server-name, resource/share-name, user/group name, object-type, access-string)
w9xAccessDel(server-name, resource/share-name, user/group name, object-type)
w9xAccessGet(server-name, resource/share-name, user/group name, object-type)
w9xAccessList(server-name, resource/share-name, object-type, flags)
w9xFileUsers(server-name, file-pathname)
w9xGetDc(server-name, domain-name, flag)
w9xGroupAdd(server-name, group-name, group-type, comment)
w9xGroupDel(server-name, group-name, group-type)
w9xGroupInfo(server-name, group, group-type, request)
w9xListGroups(server-name, group-type)
w9xMemberDel(server-name, group-name, user-name, group-type)
w9xMemberGet(server-name, group-name, user-name, group-type)
w9xMemberGrps(server-name, user-name, group-type, flags)
w9xMemberList(server-name, group-name, group-type)
w9xMemberSet(server-name, group-name, user-name, group-type)

w9xOwnerGet(server-name, reg-key, resource-name, object-type, flag)
w9xRemoteTime(server-name, format)
w9xServiceAt(server, domain, server-type, service-name, flags)
w9xServerList(server, domain, server-type)
w9xShareAdd(server-name, resource, share-name, share-type, max-users)
w9xShareDel(server-name, resource/share-name, share-type)
w9xShareInfo(server-name, resource/share-name, share-type, request)
w9xShareSet(server-name, resource/share-name, share-type, comment, s:location)
w9xUserAdd(server-name)
w9xUserAddDat(element, value)
w9xUserDel(server-name, user-name)
w9xUserExist(server-name, user-name)
w9xUserGetDat(server-name, user-name, element)
w9xUserInfo(request)
w9xUserList(server-name, account-type)
w9xUserProps(server-name, user-name, request)
w9xUserRename(server-name, old-username, new-username)
w9xUserSetDat(server-name, user-name, element, value)
w9xVersion()

•

Win32 Network Errors

185: Bad name for local drive or printer
499: Unrecognized network error #
500: Access is denied.
501: LocalName is already connected.
502: Device and resource do not match.
503: LocalName is invalid.
504: ResourceName is not valid or cannot be located.
505: The user profile is in an incorrect format.
506: Unable to open the user profile to process persistent connections.
507: LocalName is already in the user profile.
508: Password is invalid.
509: Network component is not started or invalid
510: The network is not present.
511: Device in use.
512: Device not currently connected.
513: Open files on device and FORCE=@FALSE.
514: Device not currently available.
515: Invalid Password.
516: Insufficient memory.
517: Not supported in current NT version (Invalid Parameter).
518: Invalid 'net-resource'.
519: Invalid 'scope'
520: Invalid 'type'
521: Invalid 'usage'
522: Unable to allocate (or lock) memory
523: Error enumerating network resources
524: Unable to access specified server
525: Invalid resource
526: Invalid share name
527: Invalid 'max-users'
528: Error creating share
529: Error deleting share
530: Access denied
531: Invalid parameter
532: Error looking up specified user/group name
533: Share already exists with the specified name
534: Invalid share type
535: Invalid printer name
536: Error accessing share information
537: Error setting share information
538: Unable to access specified printer
539: Error accessing printer information

540: Error setting printer information
541: Invalid resource
542: Invalid user/group name
543: Invalid 'object-type'
544: Invalid resource/share-name
545: Invalid 'access-string'
546: Invalid file/directory name
547: Error accessing file/directory information
548: Error setting file/directory information
549: Invalid registry key handle
550: Unable to open registry key
551: Error accessing registry key information
552: Error setting registry key information
553: Error accessing object's ACL
554: Error adding new access record
555: Error setting new ACL
556: Error initializing new ACL
557: Error initializing new SD
558: Error accessing ACL information
559: Error reading ACL
560: Error updating ACL
561: Invalid group name
562: Invalid user name
563: Invalid 'group-type'
564: Error getting group information
565: Error adding user to group
566: User has an invalid account type
567: Error removing user from group
568: Error itemizing groups
569: Invalid flags
570: Operation allowed only on primary domain controller
571: Operation not allowed on this special group
572: Specified 'net-resource' is not a container
573: Access denied, or invalid provider
574: Invalid provider
575: Function failed
576: Invalid server type
577: Service name not specified
578: 'flags' does not include a service type
579: 'flags' does not include a service state
580: No browser servers found
581: Error enumerating servers
582: Invalid access rights
583: Error reading access information
584: Error creating access list
585: System not set up for user-level access control
586: Unable to access specified server or domain
587: 'old-pass' is incorrect
588: 'new-pass' is too short

589: Error changing password
590: Invalid server name
591: Function not avail (RADMIN32.DLL and RLOCAL32.DLL required)
592: Operation not supported on workstations
593: Error accessing object's owner information
594: Error changing object's owner
595: Specified user/group may not be assigned as the owner
596: Invalid file name
597: Error enumerating open files
598: Error closing file
599: Invalid level (client and server are different platforms)
600: Invalid request number
601: Error getting user information
602: Invalid delimiter
603: Service control manager database does not exist
604: Invalid service name
605: Specified service does not exist
606: Service binary file could not be found
607: Service is already running
608: Service database is locked
609: A dependent service doesn't exist or is marked for deletion
610: A dependent service has failed to start
611: Service has been disabled
612: Service could not be logged on
613: Service has been marked for deletion
614: Thread could not be created for Win32 service
615: Service request timed out
616: Service is a dependency of other running services
617: Invalid or unacceptable control code
618: Service cannot accept control code in its current state
619: Service has not been started
620: Logon failure
621: 'group-type' must be @GLOBALGROUP
622: Error adding user
623: Error deleting user
624: Group already exists
625: User account already exists
626: Password is too short
627: Invalid element
628: Operation not allowed on special groups
629: Operation not allowed on last administrative account
630: Error setting user information
631: Invalid 'old-username'
632: Invalid 'new-username'
633: 'new-username' already exists
634: Error renaming user
635: Invalid privilege
636: Invalid phone number
637: Privilege 'Act as part of the operating system' not held

638: This logon type has not been granted to the specified user
639: Logon failure (invalid user-name or password)
640: Invalid level (probably non-NT server)
641: Unable to logon as specified user
642: Unable to impersonate specified user
643: Invalid account name
644: Error looking up specified account name
645: The local group already exists
646: Error adding group
647: Invalid service handle
648: Circular service dependency
649: Service display name already exists
650: Invalid service account
651: Invalid type/category
652: Invalid event ID
653: Error opening event log
654: Error writing to event log
655: Invalid source-name
656: Function not available in this version of NT
657: Function not available (RASSAPI.DLL required)
658: Function not available (WTSAPI32.DLL required)
659: Invalid account-type
660: Error enumerating users
661: Error deleting group
662: Could not find domain controller for the domain
663: Invalid name
664: No domain controllers could be found
665: Specified domain is not a trusted domain
666: The trust relationship is broken
667: Responding server is not a proper DC of the specified domain
668: Error getting domain controller
669: Error getting account policy information
670: Error accessing policy object on server
671: Error enumerating account rights
672: Specified privilege does not exist
673: Error adding account rights
674: Error removing account rights
675: 'new-pass' is too long
676: Extended network error (see WWWBATCH.INI for details)
677: 'new-pass' is unacceptable
678: 'new-pass' is unacceptable, probably because it's blank
679: 'new-pass' is unacceptable, maybe because it's being re-used
680: Password is too long
681: Password is unacceptable
682: Password is unacceptable, probably because it's blank
683: Password is unacceptable, maybe because it's being re-used
684: Error getting server information
685: Error enumerating shares
686: Invalid format

687: Error enumerating connections
688: Invalid share or computer name
689: Error getting time from server
690: Invalid timeout
691: Invalid 'force' flag
692: Invalid 'reboot' flag
693: Unable to access specified computer
694: Error shutting down specified computer
695: Error performing synchronization
696: Invalid 'value'
697: Error setting account policy information
698: Unknown interface (server is probably not a DC)
699: Service already exists

700: Invalid 'values' string
701: Error setting group information
702: Invalid class
703: Error retrieving policy information
704: Error changing policy information
705: Invalid SID string
706: Invalid floating point value
707: Error opening access token
708: Error getting token information

901: WWWNT Extender: Unknown function
903: WWWNT Extender: Need newer version of WIL DLL

AddExtender(filename)

Installs a WIL extender Dll.

Syntax:
AddExtender(filename)

Parameters:

(s) filename	WIL extender Dll filename.

Returns:

(i)	**@TRUE** if function succeeded. **@FALSE** if function failed.

WIL extender Dlls are special Dlls designed to extend the built-in function set of the WIL processor. These Dlls typically add functions not provided in the basic WIL set, such as network commands for particular networks (Novell, Windows for WorkGroups, LAN Manager and others), MAPI, TAPI, and other important Application Program Interface functions as may be defined by the various players in the computer industry from time to time. These Dlls may also include custom built function libraries either by the original authors, or by independent third party developers. (An Extender SDK is available). Custom extender Dlls may add nearly any sort of function to the WIL language, from the mundane network, math or database extensions, to items that can control fancy peripherals, including laboratory or manufacturing equipment.

Use this function to install extender Dlls as required. Up to 10 extender Dlls may be added. The total number of added items may not exceed 200 functions and constants. The **AddExtender** function must be executed before attempting to use any functions in the extender library. The **AddExtender** function should be only executed once in each WIL script that requires it.

The documentation for the functions added are supplied either in a separate manual or disk file that accompanies the extender Dll.

Example:

```
; Add vehicle radar processing dll controlling billboard visible to
; motorists, and link to enforcement computers.
; The WIL Extender SPEED.DLL adds functions to read a radar speed
; detector(GetRadarSpeed) , put a message on a billboard visible to
; the motorist (BillBoard), take a video of the vehicle (Camera), and
; send a message to alert enforcement personnel (Alert) that a
; motorist in  violation along with a picture id number to help
; identify the offending vehicle and the speed which it was going.
;
AddExtender("SPEED.DLL")
BillBoard("Drive Safely")
While @TRUE
    ; Wait for next vehicle
    while GetRadarSpeed()<5 ; if low, then just radar noise
            Yield                  ; wait a bit, then look again
    endwhile
```

```
    speed=GetRadarSpeed()          ; Something is moving out there
    if speed < 58
            BillBoard("Drive Safely")     ; Not too fast.
    else

            if speed < 63
                    BillBoard("Watch your Speed")   ; Hmmm a hot one
            else
                    if speed < 66
                            BillBoard("Slow Down")  ; Tooooo fast
                    else
                            BillBoard("Violation  Pull Over")
                            pictnum = Camera()      ; Take Video Snapshot
                            Alert(pictnum, speed); Pull this one over
                    endif
            endif
    endif
endwhile
```

See Also:

DllCall *(found in main WIL documentation)*

LastError( )

Returns the most-recent error encountered during the current WIL program.

Syntax:
LastError( )

Parameters:
None

Returns:
(i) most-recent WIL error code encountered.

In addition to the normal behavior of the LastError function documented in the WIL Reference Guide, if the most recent error occurred in a WIL Extender, then a number assigned by the Extender will be returned. The numbers are documented in the appendix of this Extender document.

It may be possible to obtain error numbers not documented. The "Notes" section of the WIL manual has been provided to allow you to keep records of undocumented error codes.

Example:

```
ErrorMode(@OFF)
FileCopy("data.dat", "c:\backups\*.*", @FALSE)
ErrorMode(@CANCEL)
If LastError() == 1006
        Message("Error", "Please call Tech Support at 555-9999.")
endif
Message("LastError"," Done.")
```

See Also:
Debug, ErrorMode *(both found in main WIL documentation)*

Net101

All network functionality for WIL is performed via "WIL Extenders", add-on Dlls for WIL, which contain Network commands for assorted networks.

NetInfo is the only WIL network function. It returns the types of the networks currently active on the local machine, and can be used to help determine which network extenders should be loaded in multi-network environments.

Documentation for the various network extenders are found either in a manual for a particular extender or in an associated disk file.

See Also:
NetInfo, AddExtender, DllCall *(found in main WIL documentation)*

NetInfo(requestcode)

Determines network(s) installed.

Syntax:
NetInfo(requestcode)

Parameters:

(i) requestcode	0 for primary network name. 1 for secondary subnet list.

Returns:

(s)	Primary network name for request code 0, or Secondary network list for request code 1.

Use this function to determine the network type(s) running on a workstation. When running in a mixed network environment, it may be important to be able to determine the types of networks running on a workstation so as to be able to load the appropriate network extender Dlls and issue the corresponding commands.

NetInfo(0) will return the name of the primary network, or will return "**MULTINET**" , which indicates the Windows multinet driver is active and the secondary subnet list should be queried. **NetInfo**(0) will return one of the following strings:

NetInfo(0) return values:

NONE	No network installed
MULTINET	Multinet driver installed, see subnet codes.
WINNT	Win32
MSNET	Microsoft Network
LANMAN	LAN Manager
NETWARE	Novell NetWare
VINES	Banyan Vines
10NET	10 Net
LOCUS	Locus
SUNPCNFS	SUN PC NFS
LANSTEP	LAN Step
9TILES	9 Tiles
LANTASTIC	Lantastic
AS400	IBM AS/400
FTPNFS	FTP NFS
PATHWORK	DEC PathWorks
OTHER1	Other (code 1)
OTHER2	Other(code 2)
UNKNOWN	Other (unknown)

If **NetInfo**(0) returned "**MULTINET**" then **NetInfo**(1) will return one or more of the following in a space delimited list:

NetInfo(1) return values:

NONE	No networks active
MSNET	Microsoft Network
LANMAN	LAN Manager
WINNET	Windows Network (Windows for WorkGroups, etc.)
NETWARE	Novell NetWare
VINES	Banyan Vines
OTHER2	Other (code 0x20)
OTHER4	Other (code 0x40)
OTHER8	Other (code 0x80)

32 Bit Windows

NetInfo(0) will always return the string "WINNT" for 32 bit Windows platforms, regardless of whether the platform is Windows 95/98/Me or Windows NT.

Under Windows 95/98/Me, **NetInfo(1)** will return a list of installed network client ID's, delimited with the standard file delimiter (by default, a tab).

Possible client ID's, with their corresponding descriptions, are:

Client ID	Description
3OPEN	3Com 3+Open (all versions)
3SHARE	3Com 3+Share (all versions)
DLR	IBM OS/2 LAN Server (versions below 1.2)
DLR12	IBM OS/2 LAN Server (version 1.2)
DLR13	IBM OS/2 LAN Server (versions 1.2, 1.3, and 1.2 without /API)
DLR13CSD	IBM OS/2 LAN Server (version 1.3 CSD 5015/5050)
DLR20	IBM OS/2 LAN Server (version 2.0)
FTPNFS	FTP Software NFS Client (InterDrive 95)
LANMAN	Microsoft Real Mode LAN Manager
LANT5	Artisoft LANtastic (version 5.X and above)
MSNET	Real mode MS-Net Compatible
NETWARE3	Novell NetWare (Workstation Shell 3.X [NETX])
NETWARE4	Novell NetWare (Workstation Shell 4.0 and above [VLM])
NOVELL32	Novell NetWare Client 32
NWREDIR	Client for NetWare Networks
PATHWKS	DEC PATHWORKS (versions below 4.0)
PATHWKS40	DEC PATHWORKS (version 4.x)
PCLP	IBM PC LAN Program (all versions)
PCNFS50	SunSoft PC-NFS (version 5.0)
VINES552	Banyan DOS/Windows 3.1 client
VREDIR	Client for Microsoft Networks

Under Windows NT, NetInfo(1) will return a list of installed network provider ID's, delimited with the standard file delimiter (by default, a tab).

Possible providers, with their corresponding descriptions, are:

Provider ID	Description
LanmanWorkstation	Microsoft Windows Network (NT client)
NetWareWorkstation	NetWare Services (Novell Netware client)
NWCWorkstation	NetWare or Compatible Network (Microsoft Netware client)

Example:

```
a=NetInfo(0)
if a=="MULTINET"
        b=NetInfo(1)
        count=ItemCount(b," ")
        Message("Multinet supporting  %count% networks", b)
else
        Message("Installed Network", a)
endif
```

See Also:
Net101, AddExtender, DllCall *(found in main WIL documentation)*

netAddDrive(user-id, pswd, net-resource, local drive, persist)

Maps a drive.

Syntax:
netAddDrive(user-id, pswd, net-resource, local-drive, persist)

Parameters:

(s) user-id	user-id or **@DEFAULT** for current user
(s) pswd	password or **@DEFAULT** for current password or **@NONE** for no password
(s) net-resource	UNC netname of net resource
(s) local drive	local drive id e.g. ("K:") or **@NONE** for connect only
(s) persist	**@TRUE** - Specifies persistent connection, one that will automatically reconnect when you reboot windows. **@FALSE** - Specifies a temporary connection.

Returns:

(i)	always 1

This function allows a connection to be made to a net resource, and, optionally, a drive to be mapped to the net resource.

Windows 95 and 98 only allow the currently 'logged-in' user credentials, therefore you must specify **@DEFAULT** for user-id and password.

Windows NT allows you to specify a user id and password, other than the currently logged-in user.

Example:

```
AddExtender("WWNET34i.DLL")
netAddDrive(@default,@default,"\\SERVER\PUB","E:",@false)
RunWait("E:\EXCEL\EXCEL.EXE","\E")
netCancelCon("E:",@True,@True)
```

See Also:
netDirDialog, netCancelCon

netAddPrinter(user-id, pswd, net-resource, local device, persist)

Maps a printer resource to a local port.

Syntax:
netAddPrinter(user-id, pswd, net-resource, local device, persist)

Parameters:

(s) user-id	user-id or **@DEFAULT** for current user
(s) pswd	password or **@DEFAULT** for current password or **@NONE** for no password
(s) net-resource	UNC netname of net resource
(s) local device	local printer port e.g. ("lpt1") or **@NONE** for connect only
(s) persist	**@TRUE** - Specifies persistent connection, one that will automatically reconnect when you reboot windows. **@FALSE** - Specifies a temporary connection.

Returns:

(i)	**@TRUE** if the port was mapped; **@FALSE** the port was not mapped.

This function allows a connection to be made to a net resource, and, optionally, a local device to be mapped to the net resource.

Example:

```
AddExtender("WWNET34i.DLL")
netAddPrinter(@default,@default,"\\SERVER\LJ4","lpt2",@false)
```

See Also:
netDirDialog, netCancelCon

netCancelCon(local drive, persist, forceflag)

Breaks a network connection.

Syntax:
netCancelCon(local drive, persist, forceflag)

Parameters:

(s) local drive	the mapped device.
(s) persist	**@TRUE** - update persistent connection table ; **@FALSE** - do not update persistent connection table to remove this device
(i) forceflag	**@TRUE** to break the connection regardless of open files. **@FALSE** - if files are open, connection will not be broken.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

When a mapped local drive is specified, only that connection will be closed.

If persist is set to **@TRUE**, then the persistent connection will be updated to remove this drive mapping from the list of persistent connections.

If forceflag is set to **0**, **netCancelCon** will not break the connection if any files on that connection are still open. If forceflag is set to **1**, the connection will be broken regardless.

Example:

```
AddExtender("WWNET34i.DLL")
netAddDrive(@default,@default,"\\SERVER\PUB\EXCEL","E:",@false)
RunWait("E:\EXCEL.EXE","\E")
netCancelCon("E:",@false,@false)
```

See Also:
netAddDrive, netDirDialog

netDirDialog(flag)

Brings up a network drive connect/disconnect dialog box

Syntax:
netDirDialog(flag)

Parameters:

(i) flag	**@FALSE**=disconnect dialog **@TRUE**=connect dialog

Returns:

(i)	always 1

This function prompts the user with either a standard Connect or Disconnect dialog box. The user may make or break network drive mappings via the dialog box.

Example:

```
AddExtender("WWNET34i.DLL")
err=netDirDialog(@TRUE)
runwait("excel.exe", "/e")
netDirDialog(@FALSE)
```

See Also:
netAddDrive

netGetCon(local drive)

Returns the name of a connected network resource.

Syntax:

netGetCon(local name)

Parameters:

(s) local name	local drive name.

Returns:

(i)	name of a network resource.

netGetCon returns the name of the network resource currently connected to a 'local name'. If the resource is not mapped a null string will be returned.

Example:

```
AddExtender("WWNET34i.DLL")
netrsrc=netGetCon("K:")
if netrsrc=="" then Message("Drive K: is","not mapped")
else Message("Drive K: is mapped to",netrsrc)
```

See Also:

netAddDrive, netDirDialog

netGetUser(netname)

Returns the name of the user currently logged into the network.

Syntax:
netGetUser(netname)

Parameters:

(s) netname	name of network or **@DEFAULT** for default network.

Returns:

(s)	the user name.

This function will interrogate the network and return the current user name. **@default** will return the user id of the local user.

"netname" can be the name of a local device, or the name of a remote network resource that the user has made a connection to, or the name of a network provider (i.e, "Microsoft Windows Network" or "NetWare Services")

Example:

```
AddExtender("WWNET34i.DLL")
username=netGetUser(@default)
Message("Current User is",username)
```

See Also:
netGetCon

netResources(net-resource, scope, type, usage)

Itemizes network resources.

Syntax:
netResources(net-resource, scope, type, usage)

Parameters:

(s) net-resource	a UNC (e.g., "\\FredsPC"), a domain (e.g., "SALES"), or ("") for the root of the network.
(i) scope	see below.
(i) type	see below.
(i) usage	see below.

Returns:

(s)	a tab-delimited list of network resources.

This function returns a tab-delimited list of network resources which are located immediately below the specified 'net-resource'.

'Scope' can be one of the following:

Req #	Meaning
1	All currently connected resources
2	All resources on the network
3	All remembered (persistent) connections

'Type' can be one of the following:

Req #	Meaning
0	All resources
1	All disk resources
2	All print resources
3	All disk and print resources

'Usage' can be one of the following:

Req #	Meaning
0	All resources
1	All connectable resources
2	All container resources
3	All connectable and container resources

Note: 'usage' is ignored unless 'scope' == 2.

Example:

```
AddExtender("WWNET34i.DLL")
servers = netResources("OFFICE", 2, 1, 1)
server = AskItemList("Available servers", servers, @TAB,@SORTED,@SINGLE)
shares = netResources(server, 2, 1, 1)
share = AskItemList("Shared directories on %server%", shares, @TAB,@SORTED,@SINGLE)
```

netVersion( )

Returns the version of this Extender DLL.

Syntax:
netVersion( )

Parameters:
none

Returns:
(i) the version number of this extender Dll.

This function is used to check the version number of this Dll in cases where older DLL's exist and alternate processing is desirable. Version numbers of newer versions will be larger than that of older versions.

Example:

```
AddExtender("WWNET34i.DLL")
a=netVersion( )
Message("Dll Version",a)
```

wntAccessAdd(server-name, resource/share-name, user/group name, object-type, access-string)

Adds or updates access (permission) records for a resource.

Syntax:

wntAccessAdd(server-name, resource/share-name, user/group-name,
object-type, access-string)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(s) user/group-name	name of a user or of a group to whom access is being granted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object.See below.
(i) access-string	the type of access that is being granted. Either a predefined access type, or a delimited list. See below.

Returns:

(s)	1.

WntAccessAdd can be used for looking up user/group names and share names (if 'resource/share-name' specifies a share).

Server-Name

All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Object-Type

'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

100	share (e.g., a directory accessed through a share)
200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be

	passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name].
	Whenever the permissions or audit settings on a printer object are changed, the permissions on any print shares associated with that printer object will also be modified. Printer object security and printer share security are linked at a very low level in WinNT/Win2K. Adding various types of permissions to a printer object will automatically add certain corresponding types of permissions to all printer shares associated with the printer object. These "linked" permissions are not identical, in that, if you grant one type of permissions on the printer object for a user/group, then a more generic set of permissions gets set on any associated printer shares. Security for printer objects can be managed. Security on printer shares should never be directly modified since the modifications will most likely be undone, if the share's printer object security is ever directly modified.
300	file or directory in an NTFS partition

*** Note:** the object types **301, 302 303**and **401** cannot be used to manipulate ACLs on resources that are located on a Win2K system. This restriction was implemented due to the fact that Win2K does have inheritable ACLs that do get automatically propagated to all child objects under a folder or registry key. Allowing a user to explicitly add an ACE [which is also inheritable] to an entire hierarchy of folders & files or registry keys could have a negative effect by creating unexpected types of access or auditing ACLs.

301	directory in an NTFS partition, and all its subdirectories *
302	directory in an NTFS partition, and all files in the directory *
303	directory in an NTFS partition, and all its subdirectories, and all files in the directory and all its subdirectories

400	registry key
401	registry key, and all its subkeys *

If **'object-type'** = 100 (share), then 'resource/share-name' should specify the name of the share.

If **'object-type'** = 300 (file or directory in an NTFS partition), then 'resource/share-name' should be UNC (ie, "\\Server\Share\Dir").

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

Access-String

'Access-string' specifies the type of access that is being granted. It can be either (A) a predefined access type, or (B) a delimited list of one or more specific 'access-records'. Both are described below:

(A) Predefined access types:

These get translated into specific access records, as shown. It is possible that the appropriate values may vary depending on your system configuration, or among different versions of Windows NT:

Access-string	Meaning	Specific equivalent
"DirNT:List"	List	"0:2:1179817"
"DirNT:Read"	Read	"0:9:-1610612736|0:2:1179817"
"DirNT:Add"	Add	"0:2:1180086"
"DirNT:AddRead"	Add & Read	"0:9:-1610612736|0:2:1180095"
"DirNT:Change"	Change	"0:9:-536805376|0:2:1245631"
"DirNT:Full"	Full Control	"0:9:268435456|0:2:2032127"
"DirNT:None"	No Access	"1:9:268435456|1:2:2032127"
"Dir2K:Full"	Full Control	"0:3:2032127"
"Dir2K:Modify"	Modify	"0:3:1245631"
"Dir2K:ReadExecute"	ReadExecute	"0:3:1179817"
"Dir2K:List"	List	"0:2:1179817"
"Dir2K:Read"	Read	"0:3:1179785"
"Dir2K:Write"	Write	"0:3:1048854"
"Dir2K:None"	No acesss	"1:3:983551"
"DirShare:Read"	Read	"0:0:1179817"
"DirShare:Change"	Change	"0:0:1245631"
"DirShare:Full"	Full Control	"0:0:2032127"
"DirShare:None"	No Access	"1:0:2032127"

Note: "DirShare" is a directory accessed through a share ("object-type" = 100). "DirNT' is a

directory in an NTFS partition, accessed through NT security ("object-type" = 300).

"File:Read"	Read	"0:0:1179817"
"File:Change"	Change	"0:0:1245631"
"File:Full"	Full Control	"0:0:2032127"
"File:None"	No Access	"1:0:2032127"
"PrintNT:Print"	Print	"0:2:131080"
"PrintNT:Manage"	Manage Documents	"0:9:268435456|0:2:131072"
"PrintNT:Full"	Full Control	"0:9:268435456|0:2:983052"
"PrintNT:None"	No Access	"1:9:268435456|1:2:983052"
"Print2K:Print	Print	"0:0:131080"
"Print2K:Manage"	Manage Documents	"0:10:131072|0:9:983056"
"Print2K:Full"	Full Control	"0:9:983056|0:0:983052"
"Print2K:None"	No Access	"1:0:983052|1:9:983056"
"Reg:Read"	Read	"0:2:131097"
"Reg:Full"	Full Control	"0:2:983103"

Despite the fact that print share permissions should not generally be directly managed, Here are some predefined permissions for print shares. Print share permissions differ between WinNT and Win2K even if "identical" permissions were set on the corresponding printer objects via the Windows explorer.

"PrintShareNT:Print"	"0:2:2032127"
"PrintShareNT:Manage"	"0:2:0"
"PrintShareNT:Full"	"0:2:2032127"
"PrintShareNT:None"	"1:2:2032127"
"PrintShare2K:Print"	"0:0:2032127"
"PrintShare2K:Manage"	"" [empty string, ignored, not applicable on Win2K]
"PrintShare2K:Full"	"0:0:2032127"
"PrintShare2K:None"	"1:0:2032127"

Also, if the pre-defined values of "PrintShare:Print", "PrintShare:Manage", "PrintShare:Full" or "PrintShare:None" are specified, then the appropriate "PrintShareNT:*" or "PrintShare2K:*" pre-defined permissions will be substituted.

(B) Specific 'access-records':

This can be a single record, or a list of records (maximum of 10) delimited with vertical bars (**|**). Each record is in the format:

```
record-type:access-flags:access-rights
```

where 'record-type', 'access-flags', and 'access-rights' are each a decimal number, separated with colons (**:**).

It is not expected that most users will want to manually create or edit 'access-records' strings.

Instead, you can use the **wntAccessGet** function to return an 'access-records' string in the proper format for use with this function. This is useful for transferring access rights from one user to another.

A brief description of the fields in the 'access-records' string follows. Please note that any detailed explanation is beyond the scope of this document, but might be obtained from the WIN32 SDK programmers' documentation available from Microsoft and other publishers.

'record-type' (aka., ACE Type) (one of the following):

0	Access Allowed ACE type
1	Access Denied ACE type
2	Audit ACE type
3	Alarm ACE Type
4	Access Allowed Compound ACE type
5	Access Allowed Object ACE type
6	Access Denied Object ACE type
7	Audit Object ACE Type
8	Alarm Object ACE Type

The ACE types 0 - 2 are commonly used. The NT extender can report on ACE types 0 - 8, but it can only manage ACE types 0 - 2 w/respect to **wnt[Access|Audit][Add|Del]()**.

'access-flags' (0, or, one or more of the following):
You can specify 0, or, one or more of the following flags, by combining them with the bitwise OR ('|') operator. These are the inherit flags.

1	OBJECT_INHERIT_ACE	Noncontainer child objects inherit the ACE as an effective ACE. For child objects that are containers, the ACE is inherited as an inherit-only ACE unless the NO_PROPAGATE_INHERIT_ACE bit flag is also set.
2	CONTAINER_INHERIT_ACE	Child objects that are containers, such as directories, inherit the ACE as an effective ACE. The inherited ACE is inheritable unless the NO_PROPAGATE_INHERIT_ACE bit flag is also set.
4	NO_PROPAGATE_INHERIT_ACE	If the ACE is inherited by a child object, the system clears the OBJECT_INHERIT_ACE and CONTAINER_INHERIT_ACE flags in the inherited ACE. This prevents the ACE from being inherited by subsequent generations of objects.
8	INHERIT_ONLY_ACE	Windows 2000: Indicates that the ACE was inherited. The system sets this bit when it propagates an inherited ACE to a child object.

16	INHERITED_ACE	Windows 2000: This flag bit **cannot** be set in an ACE via the **wntAccessAdd** and **wntAuditAdd** functions. ACEs with this flag bit enabled cannot be removed with the **wntAccessDel** or the **wntAuditDel** functions. However, ACEs retrieved with the **wntAccessGet** and **wntAuditGet** functions can have this bit enabled. To manipulate an ACE that is inherited by an object from its parent requires manipulating the ACE directly on the parent object to which the ACE is explicitly applied.
64	SUCESSFUL_ACCESS_ACE_TYPE	[Only valid with audit ACEs] Used with system-audit ACEs in a SACL to generate audit messages for successful access attempts.
128	FAILED_ACCESS_ACE_TYPE	[Only valid with audit ACEs] Used with system-audit ACEs in a SACL to generate audit messages for failed access attempts.

'access-rights' (one or more of the following, usually several, depending on the object type.)

Note: For all practical purposes, this is a complete list:

1	FILE_LIST_DIRECTORY (Dir)
1	FILE_READ_DATA (File)
2	FILE_ADD_FILE (Dir)
2	FILE_WRITE_DATA (File)
4	FILE_ADD_SUBDIRECTORY (Dir)
4	FILE_APPEND_DATA (File)
8	FILE_READ_EA
16	FILE_WRITE_EA
32	FILE_TRAVERSE (Dir)
32	FILE_EXECUTE (File)
64	FILE_DELETE_CHILD
128	FILE_READ_ATTRIBUTES
256	FILE_WRITE_ATTRIBUTES
65536	DELETE
131072	READ_CONTROL
262144	WRITE_DAC
524288	WRITE_OWNER
1048576	SYNCHRONIZE
268435456	GENERIC_ALL
536870912	GENERIC_EXECUTE

1073741824	GENERIC_WRITE
2147483648	GENERIC_READ

If any access records already exist for 'resource/share-name' for the specified 'user/group name', they will be removed before adding the records specified by 'access-string'.

Note: If you have not previously added any permissions to a share, it may implicitly have some default permissions. For example, when you create a share for a directory, it defaults to giving "Full Control" access to "Everyone". When **wntAccessAdd** is used to create an access record for such a share, it will supersede those default permissions (i.e., the default permissions will be removed). If you wish to keep the default permissions, use **wntAccessAdd** to set them explicitly.

To use wntAccessAdd to change the permissions on a file on a remote server (ie, not the local machine), you need to specify the file name as a UNC, eg:

```
wntAccessAdd("server2", "\\server2\C$\test", "Everyone", 300, "DirNT:Read")
```

("C$" and "D$" are standard admin shares to "C:\" and "D:\", etc.)

Example:

```
;This example sets the share called "Public" on the current machine so
;that any member of the group "Everybody" has full access to the contents
;of the directory associated with the "Public" share.  This function does
;not affect any permissions that may have been set with a NTFS file system
;with respect to the directory associated with the share.
;
AddExtender("WWWNT34i.DLL")
wntAccessAdd("","Public","Everybody",100,"DirShare:Full")
```

See Also:

wntAccessDel

wntAccessDel(server-name, resource/share-name, user-name, object-type)

Removes an access (permission) records from a resource.

Syntax:

wntAccessDel(server-name, resource/share-name, user-name, object-type)

Parameters:

(s) server-name		name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name		identifies the object to be modified.
(s) user-name		name of a user or of a group to whom access is being deleted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type		identifies the object type of the 'resource/share-name'.
	100	share (e.g., a directory accessed through a share)
	200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name]. Whenever the permissions or audit settings on a printer object are changed, the permissions on any print shares associated with that printer object will also be modified. Printer object security and printer share security are linked at a very low level in WinNT/Win2K. Adding various types of permissions to a printer object will automatically add certain corresponding types of permissions to all printer shares associated with the printer object. These "linked" permissions are not identical, in that, if you grant one type of permissions on the printer object for a user/group, then a more generic set of permissions gets set on any associated printer shares. Security for printer objects can be managed. Security on printer shares should never be directly modified since the modifications will most likely be undone, if the share's printer object security is ever directly modified.
	300	file or directory in an NTFS partition

*** Note:** the object types **301, 302, 303** and **401** cannot be used to manipulate ACLs on resources that are located on a Win2K system. This restriction was implemented due to the fact that Win2K does have inheritable ACLs that do get automatically propagated to all child objects under a folder or registry key. Allowing a user to explicitly add an ACE [which is also inheritable] to an entire hierarchy of folders & files or registry keys could have a negative effect by creating unexpected types of access or

auditing ACLs.

301	directory in an NTFS partition, and all its subdirectories
302	directory in an NTFS partition, and all files in the directory
303	directory in an NTFS partition, and all its subdirectories, and all files in the directory and all its subdirectories
400	registry key
401	registry key, and all its subkeys

Returns:

(i) **@TRUE** if records were deleted, **@FALSE** if no records were found.

Server-Name

All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Example:

```
; This example shows the removal of all specific references to the
; group "Supervisors" in relation to a file "D:\NTFS\blackbook.txt"
; on a NTFS file system for the current system.  Note that this could
; either remove specific permission to access the file, or it could
; remove specific access denial to the file.

AddExtender("WWWNT34i.DLL")
rslt=wntAccessDel("","D:\NTFS\blackbook.txt","Supervisors",300)
if rslt
   Message("wntAccessDel","Access records deleted")
else
   Message("wntAccessDel","No access records found")
endif
```

See Also:

wntAccessAdd

wntAccessGet(server-name, resource/share-name, user-name, object-type)

Returns access (permission) records for a resource.

Syntax:
wntAccessGet(server-name, resource/share-name, user-name, object-type)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object.
(s) user-name	name of a user or of a group for whom access is being determined. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object.

Returns:

(i)	a delimited list of access records or ("") if no records were found.

WntAccessGet returns a list of access records for 'resource/share-name' for the specified 'user/group name', delimited with vertical bars (|). If there are no appropriate records, it returns a blank string ("").

Server-Name
All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

See **wntAccessAdd** for information on the format of the access records.

Note: If no permissions were previously added to a share, it may implicitly have some default permissions. For example, when a share is created for a directory, it defaults to giving "Full Control" access to "Everyone", although there may not actually be any access records for the share. Therefore, **wntAccessGet** may return a blank string (""). However, implicit permissions will become actual permissions when from the File Manager (or Explorer) the "Permissions" dialog for the share is brought up and "OK" is selected. **wntAccessGet** can then retrieve the actual permissions.

Example:

```
records=wntAccessGet("","Public","jdoe",100)
if records==""
   Message("wntAccessGet","No records found for share 'Public'")
else
Message("wntAccessGet",strcat("'Public' Share Records are",@crlf, records))
endif
```

See Also:
wntAccessAdd, wntAccessDel

wntAccessList(server-name, resource/share-name, object-type, flags)

Returns list of users who have access (permission) records for a resource.

Syntax:

wntAccessList(server-name, resource/share-name, object-type, access string)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object .
(i) object-type	indicates the type of object identified by 'resource/share-name'. See below.
(i) flags	specifies the format of the returned names.

Returns:

(s)	a tab-delimited list of users and groups who have access records for "resource/share-name"; ie, users and groups for whom permissions have explicitly been set. Returns a blank string ("") if there are no appropriate records.

Server-Name

All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Object-Type

'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

Req#	object type
100	share (e.g., a directory accessed through a share)
200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name].
300	file or directory in an NTFS partition
400	registry key

If **'object-type'** = 100 (share), then 'resource/share-name' should specify the name of the share.

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

Flags

'Flag' specifies the format of the returned names, and can be one of the following:

0 account (eg, "johndoe")
1 domain\account (eg, "OFFICE\johndoe")

Note: For built-in accounts which are predefined by the system (eg, "Administrators"), only the account name is returned, regardless of the "flag" setting.

Example:

```
;This example sets the share called "Public" on the current machine so
;that any member of the group "Everybody" has full access to the contents
;of the directory associated with the "Public" share.  This function does
;not affect any permissions that may have been set with a NTFS file system
;with respect to the directory associated with the share.
;
AddExtender("WWWNT34i.DLL")
wntAccessAdd("","Public","Everybody",100,"DirShare:Full")
users = wntAccessList("", "Public", 100, 0)
```

See Also:

wntAccessDel, wntAccessAdd

wntAccessMod(server-name, resource, object-type, request, flag)

Manage ACL inheritance settings in the security descriptors of folders and files on NTFS partitions.

Syntax:

wntAccessMod(server-name, resource, object-type, request, flag)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	file or folder specified with either a drive letter or a UNC specification.
(i) object-type	indicates the type of object identified by 'resource'. See below.
(i) request	See below.
(i) flag	a bit masked integer value. The effects caused by having various bits set or cleared varies with the particular request # that has been specified. Refer to the documentation of the various request #'s in order to determine which flags values are valid in any particular situation.

Returns:

(i)	always 1.

Object-Type:

300 file (either a drive letter or UNC specification)

Request:

1	For WinNT, remove all ACEs from all child folders & files and then re-propagate all inheritable ACEs from the specified folder to all child folders and files. In effect, this resets the security on a tree of folders and files and makes it consistent with the DACL or SACL settings of the parent folder at the root of that branch of the folder hierarchy on a volume. For Win2K, re-enable either DACL or SACL inheritance on all child folders & files, remove all explicitly assigned ACEs on all child folders & files, and then re-propagate all inheritable ACEs to all child folders and files.
2	For Win2K only, disable either DACL or SACL inheritance on the specified folder. If Bit 0 in the flags mask is clear then all currently inherited ACEs will be converted to explicit ACEs. If Bit 0 in the flags mask is set then all currently inherited ACEs will be removed. All child folders & files will have their DACLs or SACLs updated as necessary to reflect the possible changes in their parent folder's DACL or SACL w/respect to inheritable ACE changes.
3	For Win2K only, re-enable either DACL or SACL inheritance on the specified folder. If Bit 0 in the flags mask is clear then all current explicit ACEs will be kept. If Bit 0 in the flags mask is set then all current explicit ACEs will be removed. All child folders and files will have their DACLs or

SACLs updated as necessary to reflect the possible changes in their parent folder's DACL or SACL w/respect to inheritable ACE changes.

Please note that Request 3 cannot be used with the root folder of a network drive letter that is connected to a share that does not share out the root of a physical drive letter. The reason for this is that it is impossible to get the DACL or SACL from the specified folder's parent in order to determine if the specified folder should inherit any inheritable ACEs from the parent folder's DACL or SACL. This same problem will occur if a UNC specification is passed in and the UNC spec only contains the server and share information [e.g. no subfolder information] and the share does not share the root of a physical disk drive on the server. Attempting to re-enable the ACL inheritance for either the DACL or SACL of a folder whose parent folder cannot be accessed will result in an error being generated.

For example, we have a workstation computer named WS1 and server named SRV1. Server SRV1 has a local drive named D: and there is a user share named JOHN_DOE that shares out the directory D:\HOME\JOHN_DOE. If the user "johndoe" logs in on WS1 and connects the drive letter H: to the share \\SRV1\JOHN_DOE and then tries to use wntAccessMod() with request 3 and passes in "H:\" as the value for "resource" then the function will return an error. The reason for this is that it is impossible for wntAccessMod() [being executed in a script on WS1] to get the DACL of the folder "D:\HOME" on SRV1. If the DACL of "D:\HOME" on SRV1 is not obtainable then wntAccessMod() cannot determine if there any ACEs in the DACL of "D:\HOME" on SRV1 that are inheritable by "D:\HOME\JOHN_DOE" and thus need to be propagated to the DACL of "D:\HOME\JOHN_DOE". This same problem would occur if the value of "resource" had been specified as "\\SRV1\JOHN_DOE" [as opposed to something like "H:\FOLDER1" or "\\SRV1\JOHN_DOE\FOLDER1", both of which would be OK because the parent of "FOLDER1" could be accessed to find inheritable ACEs to add to the DACL or SACL of "FOLDER1"].

Example:

See Also:

wntAuditMod

wntAcctInfo(server-name, account-name, request)

Returns information about a user account.

Syntax:
wntAcctInfo(server-name, account-name, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) account-name	the name of a user who has an account on "server-name".
(i) request	specifies the information to be returned (see below)

Returns:

(s)	information about a user account.

"**request**" specifies the information to be returned, and can be one of the following:

Value	Returns
0	Domain where the specified account name is found
1	SID (security identifier), in standard text form

Example:

```
AddExtender("WWWNT34i.DLL")
info=wntAcctInfo( "\\Server\L4", "jdoe", 0)
Message("information about a jdoe's account.",info)
exit
```

See Also:

wntUserInfo

wntAcctPolGet(server-name, request)

Gets account policy information for a server.

Syntax:
wntAcctPolGet(server-name, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(i) request	specifies the information to be returned (see below).

Returns:

(i)	an integer value.

"request" specifies the information to be returned, and can be one of the following:

Value	Returns
1	Minimum allowable password length.
2	Maximum allowable password age, in days, or -1 if passwords never expire.
3	Minimum elapsed time, in days, between when the password was last changed and when it can be changed again. A value of 0 indicates that no delay is required between password updates.
4	Amount of time, in seconds, between the end of the valid logon time and the time when the user is forced to log off the network, or -1 if users are never forced to log off. A value of 0 indicates that users will be forced to log off immediately when the valid logon time expires.
5	Length of password history maintained (ie, the number of previous passwords stored in the password history buffer). A new password cannot match any of the passwords in the buffer.
6	Lockout duration, in minutes (ie, the amount of time that a locked account will remain locked before being automatically unlocked).
7	Maximum time, in minutes, between any two failed logon attempts for lockout to occur.
8	Number of invalid password authentications for which an account is to be marked as locked out.

Note: This function can only be performed by members of the Administrators or Account Operators local group.

Example:

```
AddExtender("WWWNT34i.DLL")
passlen=wntAcctPolGet("", 1)
Message("Minimum allowable password length=", passlen)
passage=wntAcctPolGet("", 2)
Message("Maximum allowable password age for pass to expire","%passage% days (-1 if
```

```
passwords never expire)")
passelapsed=wntAcctPolGet("", 3)
Message("Minimum elapsed time between when pass last changed","%passelapsed% in
days, or 0 if no delay is required between changes")
forcedlogofftime=wntAcctPolGet("", 4)
Message("Amount of time before forced log off","%forcedlogofftime% seconds (-1 if
users are never forced to log off)%@crlf%(0=users will be forced to log off
immediately when valid logon time expires.")
numprevpasses=wntAcctPolGet("", 5)
Message("Number of previous passwords stored","%numprevpasses% passwords stored")
lockoutduration=wntAcctPolGet("", 6)
Message("Lockout duration","%lockoutduration% minutes")
maxtimefailedpass=wntAcctPolGet("", 7)
Message("Maximum time between any two failed logon attempts","%maxtimefailedpass%
minutes")
numinvalidpass=wntAcctPolGet("", 8)
Message("Number of invalid password authentications for which an account is to be
marked as locked out.","%numinvalidpass% password authentications allowed")
exit
```

See Also:

wntAcctPolSet, wntAcctInfo

wntAcctPolSet(server-name, request, value)

Sets account policy information for a server.

Syntax:
wntAcctPolSet(server-name, request, value)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(i) request	specifies the information to be set (see below).
(i) value	the new value to be set for "request".

Returns:

(i)	1, always.

"request" specifies the information to be set, and can be one of the following:

Value	Returns
1	Minimum allowable password length.
2	Maximum allowable password age, in days, or -1 if passwords never expire.
3	Minimum elapsed time, in days, between when the password was last changed and when it can be changed again. A value of 0 indicates that no delay is required between password updates.
4	Amount of time, in seconds, between the end of the valid logon time and the time when the user is forced to log off the network, or -1 if users are never forced to log off. A value of 0 indicates that users will be forced to log off immediately when the valid logon time expires.
5	Length of password history maintained (ie, the number of previous passwords stored in the password history buffer). A new password cannot match any of the passwords in the buffer.
6	Lockout duration, in minutes (ie, the amount of time that a locked account will remain locked before being automatically unlocked).
7	Maximum time, in minutes, between any two failed logon attempts for lockout to occur.
8	Number of invalid password authentications for which an account is to be marked as locked out.

Note: This function can only be performed by members of the Administrators or Account Operators local group.

Example:

```
AddExtender("WWWNT34i.DLL")
info=wntAcctPolSet("\\Server\L4", 8, 3)
```

```
Message("Set Number of invalid password authentications allowed","Number of
password authentications is set to 3)
exit
```

See Also:

wntAcctPolGet, wntAcctInfo

wntAddDrive(user-id, pswd, net-resource, local drive, persist)

Maps a drive.

Syntax:
wntAddDrive(user-id, pswd, net-resource, local-drive, persist)

Parameters:

(s) user-id	user-id or **@DEFAULT** for current user
(s) pswd	password or **@DEFAULT** for current password or **@NONE** for no password
(s) net-resource	UNC netname of net resource
(s) local drive	local drive id e.g. ("K:") or **@NONE** for connect only
(s) persist	**@TRUE** - Specifies persistent connection, one that will automatically reconnect when you reboot windows.**@FALSE** - Specifies a temporary connection.

Returns:

(i)	always 1

This function allows a connection to be made to a net resource, and, optionally, a drive to be mapped to the net resource.

Example:

```
AddExtender("WWWNT34i.DLL")
wntAddDrive(@default,@default,"\\SERVER\PUB","E:",@false)
RunWait("E:\EXCEL\EXCEL.EXE","\E")
wntCancelCon("E:",@TRUE,@TRUE)
```

See Also:
wntDirDialog, wntCancelCon

wntAddPrinter(user-id, pswd, net-resource, local device, persist)

Maps a printer resource to a local port.

Syntax:
wntAddPrinter(user-id, pswd, net-resource, local device, persist)

Parameters:

(s) user-id	user-id or **@DEFAULT** for current user
(s) pswd	password or **@DEFAULT** for current password or **@NONE** for no password
(s) net-resource	UNC netname of net resource
(s) local device	local printer port e.g. ("lpt1") or **@NONE** for connect only
(s) persist	**@TRUE** - Specifies persistent connection, one that will automatically reconnect when you reboot windows. **@FALSE** - Specifies a temporary connection.

Returns:

(i)	**@TRUE** if the port was mapped; **@FALSE** the port was not mapped.

This function allows a connection to be made to a net resource, and, optionally, a local device to be mapped to the net resource.

Example:

```
AddExtender("WWWNT34i.DLL")
wntAddPrinter(@default,@default,"\\SERVER\LJ4","lpt2",@false)
```

See Also:
wntDirDialog, wntCancelCon

wntAuditAdd(server-name, resource/share-name, user/group name, object-type, access-string)

Adds audit records for a resource.

Syntax:
wntAuditAdd(server-name, resource/share-name, user/group name, object-type, access-string)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(s) user/group-name	name of a user or of a group to whom access is being granted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object (see below).
(i) access-string	the type of access that is being granted. Either a predefined access type, or a delimited list. See below.

Returns:

(s)	1.

Server-Name
All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Object-Type
'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name]. Whenever the permissions or

	audit settings on a printer object are changed, the permissions on any print shares associated with that printer object will also be modified. Printer object security and printer share security are linked at a very low level in WinNT/Win2K. Adding various types of permissions to a printer object will automatically add certain corresponding types of permissions to all printer shares associated with the printer object. These "linked" permissions are not identical, in that, if you grant one type of permissions on the printer object for a user/group, then a more generic set of permissions gets set on any associated printer shares. Security for printer objects can be managed. Security on printer shares should never be directly modified since the modifications will most likely be undone, if the share's printer object security is ever directly modified.
300	file or directory in an NTFS partition

*** Note:** the object types **301, 302, 303** and **401** cannot be used to manipulate ACLs on resources that are located on a Win2K system. This restriction was implemented due to the fact that Win2K does have inheritable ACLs that do get automatically propagated to all child objects under a folder or registry key. Allowing a user to explicitly add an ACE [which is also inheritable] to an entire hierarchy of folders & files or registry keys could have a negative effect by creating unexpected types of access or auditing ACLs.

301	directory in an NTFS partition, and all its subdirectories *
302	directory in an NTFS partition, and all files in the directory *
303	directory in an NTFS partition, and all its subdirectories, and all files in the directory and all its subdirectories
400	registry key
401	registry key, and all its subkeys *

If **'object-type'** = 300 (file or directory in an NTFS partition), then 'resource/share-name' should be UNC (ie, "\\Server\Share\Dir").

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

wntAuditAdd does not remove any existing audit records for the specified user (or group). If you create multiple audit records for a user, it will have a cumulative effect, and should not cause any problems. If you wish to start with a "clean slate", use **wntAuditDel** first.

Access-String

'**Access-string**' specifies the type of access that is being granted. It can be either (A) a delimited list of one or more specific 'access-records', or (B) a predefined access type. Both are described below:

(A) Specific 'access-records':
This can be a single record, or a list of records (maximum of 10) delimited with vertical bars (|). Each record is in the format:

```
record-type:access-flags:access-rights
```

where 'record-type', 'access-flags', and 'access-rights' are each a decimal number, separated with colons (:).

It is not expected that most users will want to manually create or edit 'access-records' strings. Instead, you can use the **wntAuditGet** function to return an 'access-records' string in the proper format for use with this function. This is useful for transferring access rights from one user to another.

A brief description of the fields in the 'access-records' string follows. Please note that any detailed explanation is beyond the scope of this document, but might be obtained from the WIN32 SDK programmers' documentation available from Microsoft and other publishers.

'record-type' (aka., ACE Type) (one of the following):

0	Access Allowed ACE type
1	Access Denied ACE type
2	Audit ACE type
3	Alarm ACE Type
4	Access Allowed Compound ACE type
5	Access Allowed Object ACE type
6	Access Denied Object ACE type
7	Audit Object ACE Type
8	Alarm Object ACE Type

The ACE types 0 - 2 are commonly used. The NT extender can report on ACE types 0 - 8, but it can only manage ACE types 0 - 2 w/respect to **wnt[Access|Audit][Add|Del]()**.

'access-flags' (0, or, one or more of the following):
You can specify 0, or, one or more of the following flags, by combining them with the bitwise OR ('|') operator. These are the inherit flags.

1	OBJECT_INHERIT_ACE	Noncontainer child objects inherit the ACE as an effective ACE. For child objects that are containers, the ACE is inherited as an inherit-only ACE unless the NO_PROPAGATE_INHERIT_ACE bit flag is also set.
2	CONTAINER_INHERIT_ACE	Child objects that are containers, such as directories, inherit the ACE as an effective ACE. The inherited ACE is inheritable unless the NO_PROPAGATE_INHERIT_ACE bit flag is also set.
4	NO_PROPAGATE_INHERIT_ACE	If the ACE is inherited by a child object, the system clears the OBJECT_INHERIT_ACE and CONTAINER_INHERIT_ACE flags in the inherited ACE. This prevents the ACE from being inherited by subsequent generations of objects.
8	INHERIT_ONLY_ACE	Windows 2000: Indicates that the ACE was inherited. The system sets this bit when it propagates an inherited ACE to a child object.
16	INHERITED_ACE	Windows 2000: This flag bit **cannot** be set in an ACE via the **wntAccessAdd** and **wntAuditAdd** functions. ACEs with this flag bit enabled cannot be removed with the **wntAccessDel** or the **wntAuditDel** functions. However, ACEs retrieved with the **wntAccessGet** and **wntAuditGet** functions can have this bit enabled. To manipulate an ACE that is inherited by an object from its parent requires manipulating the ACE directly on the parent object to which the ACE is explicitly applied.
64	SUCESSFUL_ACCESS_ACE_TYPE	[Only valid with audit ACEs] Used with system-audit ACEs in a SACL to generate audit messages for successful access attempts.
128	FAILED_ACCESS_ACE_TYPE	[Only valid with audit ACEs] Used with system-audit ACEs in a SACL to generate audit messages for failed access attempts.

'**access-rights**' (one or more of the following, usually several, depending on the object type.)
Note: For all practical purposes, this is a complete list:

1	FILE_LIST_DIRECTORY (Dir)
1	FILE_READ_DATA (File)
2	FILE_ADD_FILE (Dir)
2	FILE_WRITE_DATA (File)
4	FILE_ADD_SUBDIRECTORY (Dir)
4	FILE_APPEND_DATA (File)
8	FILE_READ_EA
16	FILE_WRITE_EA
32	FILE_TRAVERSE (Dir)
32	FILE_EXECUTE (File)
64	FILE_DELETE_CHILD
128	FILE_READ_ATTRIBUTES
256	FILE_WRITE_ATTRIBUTES
65536	DELETE
131072	READ_CONTROL
262144	WRITE_DAC
524288	WRITE_OWNER
1048576	SYNCHRONIZE
16777216	ACCESS_SYSTEM_SECURITY
268435456	GENERIC_ALL
536870912	GENERIC_EXECUTE
1073741824	GENERIC_WRITE
-2147483648	GENERIC_READ

(B) Predefined access types:
These get translated into specific access records, as shown. It is possible that the appropriate values may vary depending on your system configuration, or among different versions of Windows NT:

Directories/Files

17957001	Read
17957142	Write
1179808	Execute
65536	Delete
262144	Change Permissions
524288	Take Ownership

Printers

8	Print
16777220	Full Control
65536	Delete
262144	Change Permissions
524288	Take Ownership

Registry keys

1	Query Value
2	Set Value
4	Create Subkey
8	Enumerate Subkeys
16	Notify
32	Create Link
65536	Delete
262144	Write DAC
131072	Read Control

You can combine multiple values using the bitwise OR ('|') operator.
For example, to audit failed reads and writes for a file:

```
rights = 17957001 | 17957142 ; READ/WRITE
wntAuditAdd("", "f:\public\readme.txt", "Guests", 300, "2:128:%rights%")
```

Instead of manually creating "access-records" strings, you can use the **wntAuditGet** function to return an "access-records" string in the proper format to be used with **wntAuditAdd**. This can be useful for duplicating audit records from one object or user to another.

Example:

```
AddExtender("WWWNT34i.DLL")
rights = 17957001 | 17957142
wntAuditAdd("", "f:\public\readme.txt", "Guests", 300, "2:128:%rights%")
```

See Also:

wntAuditDel, wntAuditGet, wntAuditList

wntAuditDel(server-name, resource/share-name, user/group name, object-type)

Removes audit records from a resource.

Syntax:
wntAuditDel(server-name, resource/share-name, user/group name, object-type)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(s) user-name	name of a user or of a group to whom access is being deleted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the object type of the 'resource/share-name'.

Server-Name
All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Object-Type identifies the object type of the 'resource/share-name' and can be one of the following

200 printer object
the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name].

Whenever the permissions or audit settings on a printer object are changed, the permissions on any print shares associated with that printer object will also be modified. Printer object security and printer share security are linked at a very low level in WinNT/Win2K. Adding various types of permissions to a printer object will automatically add certain corresponding types of permissions to all printer shares associated with the printer object. These "linked" permissions are not identical, in that, if you grant one type of permissions on the printer object for a user/group, then a more generic set of permissions gets set on any associated printer shares. Security for printer objects can be managed. Security on printer shares should never be directly modified since the modifications will most likely be undone, if the share's printer object security is ever directly modified.

300 file or directory in an NTFS partition

*** Note:** the object types **301, 302, 303** and **401** cannot be used to manipulate ACLs on resources that are located on a Win2K system. This restriction was implemented due to the fact that Win2K does have inheritable ACLs that do get automatically propagated to all child objects under a folder or registry key.

Allowing a user to explicitly add an ACE [which is also inheritable] to an entire hierarchy of folders & files or registry keys could have a negative effect by creating unexpected types of access or auditing ACLs.

301	directory in an NTFS partition, and all its subdirectories
302	directory in an NTFS partition, and all files in the directory
303	directory in an NTFS partition, and all its subdirectories, and all files in the directory and all its subdirectories
400	registry key
401	registry key, and all its subkeys

Returns:

(i) **@TRUE** if records were deleted, **@FALSE** if no records were found.

Example:

```
AddExtender("WWWNT34i.DLL")
rights = 17957001 | 17957142
wntAuditDel("", "f:\public\readme.txt", "Guests", 300)
```

See Also:

wntAuditAdd, wntAuditGet, wntAuditList

wntAuditGet(server-name, resource/share-name, user/group name, object-type)

Returns audit records for a resource.

Syntax:

wntAuditGet(server-name, resource/share-name, user/group name, object-type)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object.
(s) user-name	name of a user or of a group for whom access is being determined. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object.

Returns:

(i)	a delimited list of access records or ("") if no records were found.

WntAuditGet returns a list of access records for 'resource/share-name' for the specified 'user/group name', delimited with vertical bars (|). If there are no appropriate records, it returns a blank string ("").

See **wntAuditAdd** for information on the format of the access records.

Server-Name

All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Example:

```
AddExtender("WWWNT34i.DLL")
list=wntAuditGet("", "f:\public\readme.txt", "Guests", 300)
AskItemList("List of access records",list,@TAB,@UNSORTED,@SINGLE)
```

See Also:

wntAuditAdd, wntAuditDel, wntAuditList

wntAuditList(server-name, resource/share-name, object-type, flag)

Returns list of users who have audit records for a resource.

Syntax:
wntAuditList(server-name, resource/share-name, object-type, flag)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object .
(i) object-type	indicates the type of object identified by 'resource/share-name'. See below.
(i) flags	specifies the format of the returned names.

Returns:

(s)	a tab-delimited list of users and groups who have audit records for "resource/share-name"; ie, users and groups for whom permissions have explicitly been set. Returns a blank string ("") if there are no appropriate records.

Server-Name
All attempts to work with permissions or auditing settings on remote computers will require that those remote computers have the "Server" service running.

Object-Type
'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

Req#	object type
200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name].
300	file or directory in an NTFS partition
400	registry key

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

Flags
'Flag' specifies the format of the returned names, and can be one of the following:

0 account (eg, "johndoe")
1 domain\account (eg, "OFFICE\johndoe")

Note: For built-in accounts which are predefined by the system (eg, "Administrators"), only the account name is returned, regardless of the "flag" setting.

Example:

```
AddExtender("WWWNT34i.DLL")
file="f:\public\readme.txt"
list=wntAuditList( "", file, 300,1)
AskItemList("List of users who have audit records for a
file",list,@TAB,@UNSORTED,@SINGLE)
```

See Also:
wntAuditAdd, wntAuditDel, wntAuditGet

wntAuditMod(server-name, resource, object-type, request, flag)

Manage ACL inheritance settings in the security descriptors of folders and files on NTFS partitions.

Syntax:

wntAuditMod(server-name, resource, object-type, request, flag)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	file or folder specified with either a drive letter or a UNC specification.
(i) object-type	indicates the type of object identified by 'resource'. See below.
(i) request	See below.
(i) flag	a bit masked integer value. The effects caused by having various bits set or cleared varies with the particular request # that has been specified. Refer to the documentation of the various request #'s in order to determine which flags values are valid in any particular situation.

Returns:

(i)	always 1.

Object-Type:

300 file (either a drive letter or UNC specification)

Request:

1	For WinNT, remove all ACEs from all child folders & files and then re-propagate all inheritable ACEs from the specified folder to all child folders and files. In effect, this resets the security on a tree of folders and files and makes it consistent with the DACL or SACL settings of the parent folder at the root of that branch of the folder hierarchy on a volume. For Win2K, re-enable either DACL or SACL inheritance on all child folders & files, remove all explicitly assigned ACEs on all child folders & files, and then re-propagate all inheritable ACEs to all child folders and files.
2	For Win2K only, disable either DACL or SACL inheritance on the specified folder. If Bit 0 in the flags mask is clear then all currently inherited ACEs will be converted to explicit ACEs. If Bit 0 in the flags mask is set then all currently inherited ACEs will be removed. All child folders & files will have their DACLs or SACLs updated as necessary to reflect the possible changes in their parent folder's DACL or SACL w/respect to inheritable ACE changes.
3	For Win2K only, re-enable either DACL or SACL inheritance on the specified folder. If Bit 0 in the flags mask is clear then all current explicit ACEs will be kept. If Bit 0 in the flags mask is set then all current explicit ACEs will be removed. All child folders and files will have their DACLs or SACLs updated as necessary to reflect the possible changes in their

parent folder's DACL or SACL w/respect to inheritable ACE changes.

Please note that Request 3 cannot be used with the root folder of a network drive letter that is connected to a share that does not share out the root of a physical drive letter. The reason for this is that it is impossible to get the DACL or SACL from the specified folder's parent in order to determine if the specified folder should inherit any inheritable ACEs from the parent folder's DACL or SACL. This same problem will occur if a UNC specification is passed in and the UNC spec only contains the server and share information [e.g. no subfolder information] and the share does not share the root of a physical disk drive on the server. Attempting to re-enable the ACL inheritance for either the DACL or SACL of a folder whose parent folder cannot be accessed will result in an error being generated.

For example, we have a workstation computer named WS1 and server named SRV1. Server SRV1 has a local drive named D: and there is a user share named JOHN_DOE that shares out the directory D:\HOME\JOHN_DOE. If the user "johndoe" logs in on WS1 and connects the drive letter H: to the share \\SRV1\JOHN_DOE and then tries to use wntAccessMod() with request 3 and passes in "H:\" as the value for "resource" then the function will return an error. The reason for this is that it is impossible for wntAccessMod() [being executed in a script on WS1] to get the DACL of the folder "D:\HOME" on SRV1. If the DACL of "D:\HOME" on SRV1 is not obtainable then wntAccessMod() cannot determine if there any ACEs in the DACL of "D:\HOME" on SRV1 that are inheritable by "D:\HOME\JOHN_DOE" and thus need to be propagated to the DACL of "D:\HOME\JOHN_DOE". This same problem would occur if the value of "resource" had been specified as "\\SRV1\JOHN_DOE" [as opposed to something like "H:\FOLDER1" or "\\SRV1\JOHN_DOE\FOLDER1", both of which would be OK because the parent of "FOLDER1" could be accessed to find inheritable ACEs to add to the DACL or SACL of "FOLDER1"].

Example:

See Also:

wntAccessMod

wntCancelCon(local drive, persist, forceflag)

Breaks a network connection.

Syntax:
wntCancelCon(local drive, persist, forceflag)

Parameters:

(s) local drive	the mapped device. Drive letter or UNC path.
(s) persist	**@TRUE** - update persistent connection table ; **@FALSE** - do not update persistent connection table to remove this device.
(i) forceflag	**@TRUE** to break the connection regardless of open files. **@FALSE** - if files are open, connection will not be broken.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

You can use EITHER the drive letter OR UNC path name in the netcancelcon, wntcancelcon, w95cancelcon functions,in the 32-bit network extenders. Note that when you specify the UNC path name in these 3 functions, that any drive letters that are mapped to that UNC path will be broken when you cancel the connection.

When a mapped local drive is specified, only that connection will be closed.

If persist is set to **@TRUE**, then the persistent connection will be updated to remove this drive mapping from the list of persistent connections.

NOTE: It is important to specify **@TRUE** for the persist parameter in wntCancelCon, if you plan to map this drive letter again using the function **wntAddDrive**.

If forceflag is set to **@FALSE**, **wntCancelCon** will not break the connection if any files on that connection are still open. If forceflag is set to **@TRUE**, the connection will be broken regardless.

Example:

```
AddExtender("WWWNT34i.DLL")
wntAddDrive(@default,@default,"\\SERVER\PUB\EXCEL","E:",@false)
RunWait("E:\EXCEL.EXE","\E")
wntCancelCon("E:",@TRUE,@TRUE)
```

See Also:
wntAddDrive, wntDirDialog

wntChgPswd(server/domain, user-name, old-password, new-password)

Changes a user's password.

Syntax:

wntChgPswd(server/domain, user-name, old-password, new-password)

Parameters:

(s) server/domain	either a server name (e.g., "\\MYSERVER") or domain name (e.g., "SALES") on which the password will be changed, or a blank string ("") for the current user's login domain.
(s) user-name	the name of the user whose password will be changed, or a blank string ("") for the current user's login name.
(s) old-password	the user's password on "server/domain" or "*UNKNOWN*" (see below).
(s) new-password	the new password to be set or a blank string ("") to indicate no password is desired.

Returns:

(i)	1

old-password

wntChgPswd can be used to specify a new password without knowing the old password, if you are a member of the Administrators or Account Operators local group. To do this, specify "*UNKNOWN*" as the old password. In this case, the "user" parameter must specify an actual user name (ie, it cannot be a blank string).

Example:

```
AddExtender("wwwnt34i.dll")
old=AskPassword("Change Password","Enter old password")
while @TRUE
    new1=AskPassword("Change Password","Enter new password")
    new2=AskPassword("Change Password","Enter new password again")
    if new1==new2 then break
    Message("Try Again","The new passwords you entered do not match")
endwhile

rslt=wntChgPswd("", "", old, new1)

if rslt
    Message("Password","Successfully changed")
else
    Message("Password","Not changed")
endif
```

See Also:

wntCurrUsers(server-name, flags)

Lists users currently logged into a server.

Syntax:

wntCurrUsers(server-name, flags)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) flags	specifies the format of the returned names (see below).

Returns:

(s)	a tab-delimited list of users currently logged into a server.

"**flags**" specifies the format of the returned names, and can be one of the following:

0	account	(eg, "johndoe")
1	domain\account	(eg, "OFFICE\johndoe")

Example:

```
AddExtender("wwwnt34i.dll")
server="\\SERVER\LJ4"
userlist=wntCurrUsers(server, 0)
AskItemList("List of Users on %server%",userlist,@tab, @sorted, @single)
exit
```

See Also:

wntUserExist, wntUserAdd, wntUserDel

wntDirDialog(flag)

Brings up a network drive connect/disconnect dialog box

Syntax:
wntDirDialog(flag)

Parameters:

(i) flag	**@FALSE**=disconnect dialog
	@TRUE=connect dialog

Returns:

(i)	**1**

This function prompts the user with either a standard Connect or Disconnect dialog box. The user may make or break network drive mappings via the dialog box.

Example:

```
AddExtender("WWWNT34i.DLL")
err=wntDirDialog(@TRUE)
runwait("excel.exe", "/e")
wntDirDialog(@FALSE)
```

See Also:
wntAddDrive

wntDomainSync(server-name, request)

Synchronizes domain controllers.

Syntax:

wntDomainSync(server-name, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(i) request	specifies the operation to perform. See below.

Returns:

(i)	1, always.

"request" specifies the operation to perform, and can be one of the following values:

Value	Meaning
2	Force replicate on BDC (standard replication). "server-name" should be a BDC.
3	Force synchronize on BDC (full replication). "server-name" should be a BDC.
4	Force PDC to broadcast change. "server-name" should be a PDC.
5	Force to re-discover trusted domain DCs.
7	Notify netlogon that a new transport has come online.

Example:

```
AddExtender("WWWNT34i.DLL")
;This example will cause all BDC's to be replicated.
PDC=wntGetDc( "", "WorkGrp", 1)
message("Primary Domain controller for Workgrps",PDC)
wntDomainSync( PDC, 2)
exit
```

See Also:

wntGetDc, wntServerList

wntEventLog

Writes a "WBMSG" entry to the NT application event log.

Syntax:
wntEventLog(server-name, text)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) text	a string that will be stored with the event, or "" for none.

Returns:

(i)	always 1.

This function will write an event with the message "WBMSG: xxx" to the NT application event log on the specified server, with "xxx" being replaced by the specified "text" string, and with an EventID of 1.

This function will attempt to set the following values in the registry on the local machine, to specify the location of the message file:

REGEDIT4
```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\Application\WBMSG]

"EventMessageFile"="c:\\program files\\winbatch\\system\\wwwnt34i.dll"
"TypesSupported"=dword:00000004
```

If that key is not writable, or needs to be set on a remote machine, you will need to set those values manually before using this function.

Note: that the value "EventMessageFile" must be set to the full path of the Windows NT extender DLL.

The message file is part of the extender DLL itself. Since the Event Viewer locks and caches message files being used, you may need to close the Event Viewer before replacing or moving the extender DLL.

Example:
```
AddExtender("wwnt34i.dll")
;write to the application event log
wntEventLog("","Successful Exit")
exit
```

See Also:

wntEventWrite

wntEventWrite(server-name, source-name, type/category, event-id, description)

Writes an entry to an NT event log.

Syntax:
wntEventWrite(server-name, source-name, type/category, event-id, description)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) source-name	a subkey of a logfile entry under the "EventLog" key in the registry (see below).
(i) type/category	consists of one value for "type" and one value for "category", combined with the bitwise OR ('|') operator (see below)
(i) event-id	the event identifier. The event identifier specifies the message that goes with this event as an entry in the message file associated with the event source.
(s) description	an optional string containing additional information that will be stored with the event, or "".

Returns:

(i)	**1**

"**source-name**" is a subkey of a logfile entry under the "EventLog" key in the registry. For example, the source name "WinApp" would be valid if the registry had the following form:

```
HKEY_LOCAL_MACHINE
  System
    CurrentControlSet
      Services
        EventLog
          Application
            WinApp
          Security
          System
```

If the source name cannot be found, the event logging service uses the Application logfile with no message files for the event identifier or category.

"**type/category**" consists of one value for "type" and one value for "category", combined with the bitwise OR ('|') operator:

type: specifies the type of event being logged, and can be one of the following values:

Value	Meaning
65536	EVENTLOG_ERROR_TYPE
131072	EVENTLOG_WARNING_TYPE
262144	EVENTLOG_INFORMATION_TYPE
524288	EVENTLOG_AUDIT_SUCCESS
1048576	EVENTLOG_AUDIT_FAILURE

category: specifies the event category. This is source-specific information; the category can have any value (0 - 65535).

Example:

```
;In order to run this script, you will need to have created and compiled a ;message
file defining a message #15, otherwise the event will say "message ;not found" and
it will also include the specified string (ie, "Unexpected ;response").
AddExtender("WWWNT34i.DLL")
wntEventWrite(server, "My Monitor", 262144, 15, "Unexpected response")
Exit
```

See Also:

wntEventLog, Complete list of NT functions

wntFileClose(server-name, file-pathname)

Close all network connections to a file.

Syntax:
wntFileClose(server-name, file-pathname)

Parameters:

(s) server-name	"server-name" is the name of a remote server on which the function will execute, or a blank string ("") to indicate the local computer.
(s) file-pathname	"file-pathname" is a fully-qualified file name (eg,"C:\DOC\MYFILE.TXT"). NOTE: The file name MUST be fully-qualified.

Returns:

(i)	Returns the number of connections which existed (and were closed) for the specified file.

Note: this function will not work to do a wntFileClose on a local file that was opened by yourself on the local machine, even if it's done over a mapped drive with a UNC.

Example:

```
;the "C:\DOC\MYFILE.TXT" on the local machine needs to be opened by
someone else over the network.

AddExtender("WWWNT34i.DLL")
wntFileClose("", "C:\DOC\MYFILE.TXT")
```

See Also:
wntAddDrive

wntFilesOpen(server-name)

Lists open files on a server.

Syntax:
wntFilesOpen(server-name)

Parameters:

(s) server-name	"server-name" is the name of a remote server on which the function will execute, or a blank string ("") to indicate the local computer.

Returns:

(s)	Returns a tab-delimited list of file names.

Note: This function can only be performed by members of the Administrators or Account Operators local group.

Example:

```
AddExtender("WWWNT34i.DLL")
files=wntFilesOpen("")
AskItemList("List of open files on a server.",files,@TAB,@SORTED,@SINGLE)
exit
```

See Also:

wntFileClose

wntFileUsers(server-name, file-pathname)

Lists network users who have a file open.

Syntax:
wntFileUsers(server-name, file-pathname)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER") or a blank string ("") to indicate the current machine.
(s) file-pathname	a fully-qualified file name (eg, "C:\DOC\MYFILE.TXT") **Note**: The file name MUST be fully-qualified.

Returns:

(s)	a tab-delimited list of user names.

Note: The full path to the file in the WntFileUsers command is the path "as seen if you are at the server itself" and not "path as mapped from the workstation"

Example:

```
AddExtender("WWWNT34i.DLL")
users=wntFileUsers( "\\Server","C:\DOC\MYFILE.TXT")
AskItemList("List of network users who have a file open", users, @TAB, @SORTED,
@SINGLE)
exit
```

See Also:

wntCurrUsers

wntGetCon(local drive)

Returns the name of a connected network resource.

Syntax:
wntGetCon(local name)

Parameters:

(s) local name	local drive name or printer resource (ie.,LPT1).

Returns:

(s)	name of a network resource.

wntGetCon returns the name of the network resource currently connected to a 'local name'. If the resource is not mapped a null string will be returned.

Example:

```
AddExtender("WWWNT34i.DLL")
netrsrc=wntGetCon("K:")
if netrsrc==="" then Message("Drive K: is","not mapped")
else Message("Drive K: is mapped to",netrsrc)
```

See Also:
wntAddDrive, wntDirDialog

wntGetDc(server-name, domain-name, flag)

Returns the domain controller for a domain.

Syntax:
wntGetDc(server-name, domain-name, flag)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) domain-name	is the name of a domain, or a blank string ("") to indicate the primary domain.
(i) flag	(see below)

Returns:

(s)	Returns a domain controller name in UNC format (eg, "\\MYSERVER").

"**flag**" can be one of the following:

Flag	Meaning
0	Return any domain controller for the specified domain This will get the name of any domain controller for "domain-name" that is directly trusted by "server-name". "server-name" must be part of a domain (it cannot be a standalone computer). If "server-name" is a Windows NT Workstation that is a member of a domain or a Windows NT Server member, "domain-name" must be in the same domain as "server-name". If "server-name" is a Windows NT Server domain controller, "domain-name" 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.
1	Return the primary domain controller for the specified domain

Example:

```
AddExtender("WWWNT34i.DLL")
dc=wntGetDc( "", "WorkGrp", 1)
message("Primary Domain controller for Workgrps",dc)
```

See Also:
wntServerList

wntGetDrive(net-resource)

Lists local drives mapped to a UNC.

Syntax:
wntGetDrive(net-resource)

Parameters:

(s) net-resource	specifies a UNC, in the form "\\SERVER\SHARE". It is not case-sensitive, but must otherwise EXACTLY match the UNC name to which the drive(s) are mapped (eg, must not have a trailing backslash).

Returns:

(s)	Returns a tab-delimited list of drives (eg, "H: W:").

Example:

```
AddExtender("WWWNT34i.DLL")
drvltr=wntGetDrive("\\Server\Share")
Askitemlist("Server is mapped to",drvltr,@tab,@unsorted,@single)
```

See Also:

wntGetCon

wntGetUser(netname)

Returns the name of the user currently logged into the network.

Syntax:
wntGetUser(netname)

Parameters:

(s) netname -	name of network or **@DEFAULT** for default network.

Returns:

(s)	the user name.

This function will interrogate the network and return the current user name. **@default** will return the user id of the local user.

"netname" can be the name of a local device, or the name of a remote network resource that the user has made a connection to, or the name of a network provider (i.e, "Microsoft Windows Network" or "NetWare Services")

Example:

```
AddExtender("WWWNT34i.DLL")
username=wntGetUser(@default)
Message("Current User is",username)
```

See Also:
wntGetCon

wntGroupAdd(server-name, group-name, group-type, comment)

Creates a user group.

Syntax:
wntGroupAdd(server-name, group-name, group-type, comment)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) group-name	the name of the group to be created.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.
(s) comment	is an optional description of the group, or "" for none.

Returns:

(s)	always 1.

Example:

```
AddExtender("WWWNT34i.DLL")
wntGroupAdd("\\Server\L4", "Everybody", @GLOBALGROUP, "")
Message("Everybody","Group added")
exit
```

See Also:

wntListGroups, wntGroupInfo

wntGroupDel(server-name, group-name, group-type)

Deletes a user group.

Syntax:
wntGroupDel(server-name, group-name, group-type)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) group-name	is the name of a group to be deleted.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.

Returns:

(s)	always 1.

Example:

```
AddExtender("WWWNT34i.DLL")
wntGroupDel("\\Server\L4", "DaGroup", @GLOBALGROUP)
Message("DaGroup","Group Deleted")
exit
```

See Also:

wntGroupAdd

wntGroupEdit(server-name, group, group-type, request, value)

Modifies information about a group.

Syntax:
wntGroupEdit(server-name, group, group-type, request, value)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) group-name	is the name of a group to be deleted.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.
(i) request	the element to be modified. See below.
(i/s) value	specifies the new value to be set for the element.

Returns:

(s)	always 1.

"request" specifies the element to be modified, and can be the following:

Request	Element
1	group comment

Note: This function can only be performed by members of the Administrators or Account Operators local group.

Example:

```
AddExtender("WWWNT34i.DLL")
wntGroupEdit("\\Server\L4", "DaGroup", @GLOBALGROUP,1,"Comment goes here")
Message("DaGroup","Comment was Edited")
exit
```

See Also:
wntGroupAdd

wntGroupInfo(server-name, group, group-type, request)

Returns information about a group.

Syntax:
wntGroupInfo(server-name, group, group-type, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(s) group	name of a group.
(s) group-type	can be **@LOCALGROUP** or **@GLOBALGROUP**.
(s) request	specifies the information to be returned, (see below).

Returns:

(s)	Returns a string.

"request" specifies the information to be returned, and can be one of the following:

request	type	description
-1	(i)	**@TRUE** if group exists, **@FALSE** if it doesn't.
0	(s)	group name
1	(s)	group comment
2	(i)	group's RID (relative identifier) This request is valid only with global groups.

Example:

```
AddExtender("WWWNT34i.DLL")
comment = wntGroupInfo("", "Administrators", @LOCALGROUP, 1)
Message("Comment for 'Administrators'", comment)
```

See Also:
wntListGroups

wntGroupRen(server-name,group, group-type, new-name)

Renames a group.

Syntax:
wntGroupRen(server-name, group, group-type, new-name)

Parameters:

(s) server	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(s) group	group name.
(s) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.
(s) new-name	specifies the new name for the group.

Returns:
(i) always 1.

Note: This function can only be performed by members of the Administrators or Account Operators local group.

Example:

```
AddExtender("WWWNT34i.DLL")
wntGroupRen("", "Office", @LOCALGROUP, "Office01")
Message("Group rename", "Successful")
```

See Also:

wntListGroups

wntListGroups(server-name, group-type)

Returns tab-delimited list of all user groups on a specified server.

Syntax:

wntListGroups(server-name, group-type)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.

Returns:

(i)	a tab-delimited list of user groups.

Example:

```
AddExtender("WWWNT34i.DLL")
localgroups=wntListGroups("",@LOCALGROUP)
AskItemList("Local Groups",localgroups,@tab,@sorted,@single)
globalgroups=wntListGroups("",@GLOBALGROUP)
AskItemList("Global Groups",globalgroups,@tab,@sorted,@single)
```

See Also:

wntMemberList

wntLsaPolGet

Gets LSA (Local Security Authority) policy information.

Syntax:
wntLsaPolGet(server-name, class, element)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) class	one of the following: "AuditEvents" "PrimaryDomain" "AccountDomain" "LsaServerRole" "DnsDomain" (Windows 2000 only)
(i) element	For each class, the possible elements are shown below.

Returns:

(s/i)	Returns a string or integer value.

For each **class**, the possible **elements** are:

"AuditEvents"

-1 Auditing mode (1 if auditing is enabled, 0 if not)

0 Number of Audit Event types

Audit Category:

1 Restart, Shutdown & System
2 Logon and Logoff
3 File and Object Access
4 Use of User Rights
5 Process Tracking
6 Security Policy Changes
7 User and Group Management
8 DirectoryServiceAccess (Windows 2000)
9 AccountLogon (Windows 2000)

Each 'audit category' returns an integer indicating the auditing options for that category:

0 None
1 Success
2 Failure
3 Success and Failure

"PrimaryDomain"

1 Name

2 Sid

"AccountDomain"

1 Domain Name
2 Domain Sid

"LsaServerRole"

1 Server Role (2=backup, 3=primary)

"DnsDomain" (Windows 2000 only)

1 Name
2 Dns Domain Name
3 Dns Forest Name
4 Domain Guid
5 Sid

Example:

```
AddExtender("wwwnt34i.dll")
flag=wntLsaPolGet("","AuditEvents",1)
if flag==0
   Message("Restart, Shutdown and System","Not being audited")
endif
if flag==1
   Message("Restart, Shutdown and System","Successes Only")
endif
if flag==2
   Message("Restart, Shutdown and System","Faliures Only")
endif
if flag==3
   Message("Restart, Shutdown and System","Both Successes and Failures")
endif
```

See Also:

wntLsaPolSet

wntLsaPolSet

Sets LSA (Local Security Authority) policy information.

Syntax:
wntLsaPolSet(server-name, class, element, value)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) class	is one of the following: "AuditEvents" "PrimaryDomain" "AccountDomain" "LsaServerRole" "DnsDomain" (Windows 2000 only)
(i) element	For each class, the possible elements are shown below.
(i) value	depends on which element is to be set.

Returns:

(i)	Returns 1

For each **class**, the possible **elements** are:

"AuditEvents"

-1 Auditing mode (1 - enable event auditing, 0 - Do not audit)

Audit Category:

1 Restart, Shutdown & System
2 Logon and Logoff
3 File and Object Access
4 Use of User Rights
5 Process Tracking
6 Security Policy Changes
7 User and Group Management
8 DirectoryServiceAccess (Windows 2000)
9 AccountLogon (Windows 2000)

Each 'audit category' requires an integer indicating the auditing options for that category:

0 None
1 Success
2 Failure
3 Success and Failure

"PrimaryDomain"

1 Name
2 Sid

"AccountDomain"

1 Domain Name
2 Domain Sid

"LsaServerRole"

1 Server Role (2=backup, 3=primary)

"DnsDomain" (Windows 2000 only)

1 Name
2 Dns Domain Name
3 Dns Forest Name
(4 Domain Guid) *** not yet implemented
5 Sid

Example:

```
AddExtender("wwwnt34i.dll")
flag=wntLsaPolGet("","AuditEvents",1)
Message("Before",flag)
wntLsaPolSet("","AuditEvents",1,1)
flag=wntLsaPolGet("","AuditEvents",1)
Message("After",flag)
```

See Also:

wntLsaPolGet

wntMemberDel(server-name, group-name, user-name, group-type)

Deletes the specified user from the specified group on the specified server.

Syntax:
wntMemberDel(server-name, group-name, user-name, group-type)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(s) group-name	name of the group.
(s) user-name	name of the current user.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if the user is not a member of the group.

Assuming that the person running this script has sufficient authority to delete users from the specified group, this function will delete the specified user from the group.

Example:

```
AddExtender("WWWNT34i.DLL")
rslt=wntMemberDel("","testgroup","jdoe",@LOCALGROUP)
if rslt
     Message("jdoe","removed from testgroup")
else
     Message("jdoe","is not a member of testgroup")
endif
```

See Also:
wntMemberGet, wntMemberSet

wntMemberGet(server-name, group-name, user-name, group-type)

Determines if the specified user is a member of the specified group on the specified server.

Syntax:
wntMemberGet(server-name, group-name, user-name, group-type)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(s) group-name	name of the group.
(s) user-name	name of the current user.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

Assuming that the person running this script has sufficient authority to query members of the specified group, this function will allow the person to determine if the user is a member of the specified group or not.

Example:

```
AddExtender("WWWNT34i.DLL")
rslt=wntMemberGet("","testgroup","jdoe",@LOCALGROUP)
if rslt
     Message("jdoe","is a member of testgroup")
else
     Message("jdoe","is not a member of testgroup")
endif
```

See Also:
wntMemberSet, wntMemberDel

wntMemberGrps(server-name, user-name, group-type, flags)

Returns a list of groups to which the specified user belongs.

Syntax:

wntMemberGrps(server-name, user-name, group-type, flags)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(s) user-name	name of the current user.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**
(i) flags	1 - with group-type of **@LOCALGROUP** to include groups in which user is an indirect member. Otherwise, set to 0.

Returns:

(i)	a tab delimited list of groups.

Example:

```
AddExtender("WWWNT34i.DLL")
groups=wntMemberGrps("","jdoe",@LOCALGROUP,1)
AskItemList("jdoe is associated with",groups,@tab,@sorted,@single)
```

See Also:

wntMemberSet, wntMemberDel

wntMemberList(server-name, group-name, group-type[,get-domain-flag])

Returns a list of the members of a user group.

Syntax:

wntMemberList(server-name, group-name, group-type [,get-domain-flag])

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(s) group-name	name of the group.
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP**
(i) get-domain-flag	optional parameter. If this parameter is omitted or is specified with an integer value of zero (0) then the function will behave in its original manner and only return the basic names of users/groups that are members of a local group. However, if this parameter is specified with an integer value of one (1), then the function will return fully qualified names of the form "domain\user" or "domain\group" when dealing with local groups that have global groups or domain users as their members]. This parameter has no effect when retrieving the list of members of a local group.

Returns:

(i)	a tab delimited list of users.

Example:

```
AddExtender("WWWNT34i.DLL")
people=wntMemberList("","Everybody",@LOCALGROUP,0)
AskItemList("Member of group Everybody",people,@tab,@sorted,@single)
```

See Also:

wntMemberSet, wntMemberDel

wntMemberSet(server-name, group-name, user-name, group-type)

Sets the specified user/ or group as a member of the specified group on the specified server.

Syntax:
wntMemberSet(server-name, group-name, user-name or group-name, group-type)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(s) group-name	name of the group.
(s) user-name/group-name	name of the current user or group
(i) group-type	**@LOCALGROUP** or **@GLOBALGROUP.**

Returns:

(i)	**@TRUE** if successful; **@FALSE** if the user is already a member of the group.

The person running this script must have sufficient authority to add users/or groups to the specified group.

Example:

```
AddExtender("WWWNT34i.DLL")
rslt=wntMemberSet("","testgroup","jdoe",@LOCALGROUP)
if rslt
     Message("jdoe","is a NEW member of testgroup")
else
     Message("jdoe","is ALREADY a member of testgroup")
endif

;To add a domain globalgroup to a local machine's group
AddExtender("WWWNT34i.DLL")
rslt=wntMemberSet("\\server", "localgroup", "domain\globalgroup", @LOCALGROUP)
if rslt
     Message("domain\globalgroup","is a NEW member of localgroup")
else
     Message("domain\globalgroup","is ALREADY a member of localgroup")
endif
```

See Also:
wntMemberDel, wntMemberGet

wntOwnerGet(server-name, reg-key, resource-name, object-type, flag)

Returns the owner of an object.

Syntax:

wntOwnerGet(server-name, reg-key, resource-name, object-type, flag)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(i) reg-key	handle of an open registry key or 0. See below.
(s) resource-name	identifies the object to be accessed.
(i) object-type	indicates the type of object identified by 'Resource-name'.
(i) flag	specifies the format of the returned string.

Returns:

(s)	the name of the account that owns the object, or a blank string ("") if the object has no owner.

'Server-name' can be the name of the server on which the function will execute, or a blank string ("") to indicate the current machine. This is used for looking up user/group names.

'Reg-key' is used only if the object is a registry key. If 'Object-type' = 400 (registry key), then 'Reg-key' should be the handle of an open registry key (opened with the "RegOpenKey" function), or a predefined registry handle (listed in the WIL Reference under "Registration Database Operations"). Otherwise, 'Reg-key' is ignored, and should be specified as 0.

'Resource-name' identifies the object to be accessed. if 'Object-type' = 400 (registry key), then 'Resource-name' can be a subkey string relative to 'Reg-key', or a blank string ("") if 'Reg-key' represents the actual object to be accessed. Otherwise, 'Resource-name' should specify the name of the object (eg, "C:\UTIL\MYFILE.TXT").

'Object-type' indicates the type of object identified by 'Resource-name', and can be one of the following:

Req #	Meaning
300	file or directory in an NTFS partition
400	registry key

'Flag' specifies the format of the returned string, and can be one of the following:

Req #	Meaning
0	account (eg, "johndoe")
1	domain\account (eg, "OFFICE\johndoe")

Note: For built-in accounts which are predefined by the system (eg, "Administrators"), only the

account name is returned, regardless of the 'Flag' setting.

Example:

```
;For a file:
AddExtender("WWWNT34i.DLL")
ErrorMode(@OFF)
owner = wntOwnerGet("", 0, "f:\test\myfile.txt", 300, 0)
ErrorMode(@ON)
Message("Owner is", owner)


;For a registry key:
AddExtender("WWWNT34i.DLL")
ErrorMode(@OFF)
owner = wntOwnerGet("", @REGMACHINE, "Software\Test", 400, 0)
ErrorMode(@ON)
Message("Owner is", owner)
```

See Also:

wntOwnerSet, wntMemberGet

wntOwnerSet(server-name, reg-key, resource-name, object-type, user/group name)

Takes ownership of an object.

Syntax:

wntOwnerSet(server-name, reg-key, resource-name, object-type, user/group name)

Parameters:

(s) server-name	name of a network file server ("\\MYSERVER") or ("") for a local PC.
(i) reg-key	handle of an open registry key or 0.
(s) resource-name	identifies the object to be accessed.
(i) object-type	indicates the type of object identified by 'Resource-name'.
(s) user/group name	name of user/group who is taking ownership of the object.

Returns:

(i)	1

'User/group name' is the name of the user or group who is taking ownership of the object. If necessary, it can be fully qualified as "server\user". It can be either the current user, or "Administrators". You can specify a blank string ("") to indicate "Administrators".

'Object-type' indicates the type of object identified by 'Resource-name', and can be one of the following:

Req #	Meaning
300	file or directory in an NTFS partition
400	registry key

In addition to taking ownership of an object, you may also wish to reset the object's access (permission) records to the system default (which grants full control to everyone). To do this, add 50 to the values for **'Object-type'** listed above, ie:

Req #	Meaning
350	file or directory in an NTFS partition -- reset access records
450	registry key -- reset access records

After resetting the object's access records, you can use the wntAccessAdd function to change the object's permissions.

See wntOwnerGet for additional parameter information.

Example:

```
;For a file:
AddExtender("WWWNT34i.DLL")
```

```
wntOwnerSet("", 0, "f:\test\myfile.txt", 350, "Administrators")

;For a registry key:
AddExtender("WWWNT34i.DLL")
wntOwnerSet("", @REGMACHINE, "Software\Test", 450, "Administrators")
```

See Also:
wntOwnerGet, wntMemberGet

wntPrivAdd(server-name, user/group name, privilege)

Grants a privilege to a user or group..

Syntax:

wntPrivAdd(server-name, user/group name, privilege)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) user/group name	the name of a user or group. If necessary, it can be fully qualified as "server\user" (or "server\group").
(s) privilege	a user right which may be granted. See below.

Returns:

(i)	1

This function requires Administrator privileges on the local workstation.

The following is a list of possible "privilege" values (although there may be others):
Note: These privileges will not take affect until next login.

Privilege	Description	User Right
SeAssignPrimaryTokenPrivilege	Required to assign the primary token of a process.	Replace a process level token.
SeAuditPrivilege	Required to generate audit-log entries. Give this privileg to secure servers.	Generate security audits.
SeBackupPrivilege	Required to perform backup operations.	Back up files and directories.
SeBatchLogonRight	Required for an account to log on using the batch logon type	Log on using the batch logon
SeChangeNotifyPrivilege	Required to receive notifications of changes to files or directories. This privilege also causes the system to skip all traversal access checks. It is enabled by default for all users.	Bypass traverse checking.
SeCreatePagefilePrivilege	Required to create a paging file.	Create a pagefile.
SeCreatePermanentPrivilege	Required to create a permanent object.	Create permanent shared objects.
SeCreateTokenPrivilege	Required to create a primary token.	Create a token object.
SeDebugPrivilege	Required to debug a process.	Debug programs.
SeDenyBatchLogonRight	Explicitly denies an account the right to log on using the batch logon type	Denies an account the right to log on using the batch logon type

SeDenyInteractiveLogonRight	Explicitly denies an account the right to log on using the interactive logon type.	Denies an account the right to log on using the interactive logon type.
SeDenyNetworkLogonRight	Explicitly denies an account the right to log on using the network logon type.	Denies an account the right to log on using the network logon type.
SeDenyServiceLogonRight	Explicitly denies an account the right to log on using the service logon type.	Denies an account the right to log on using the service logon type.
SeIncreaseBasePriorityPrivilege	Required to increase the base priority of a process.	Increase scheduling priority.
SeIncreaseQuotaPrivilege	Required to increase the quota assigned to a process.	Increase quotas.
SeInteractiveLogonRight	Required for an account to log on using the interactive logon type.	Log on using the interactive logon
SeLoadDriverPrivilege	Required to load or unload a device driver.	Load and unload device drivers.
SeLockMemoryPrivilege	Required to lock physical pages in memory.	Lock pages in memory
SeMachineAccountPrivilege	Required to create a machine account.	Add workstations to domain.
SeNetworkLogonRight	Required for an account to log on using the network logon type.	Log on using the network logon type.
SeProfileSingleProcessPrivilege	Required to gather profiling information for a single process.	Profile single process.
SeRemoteShutdownPrivilege	Required to shut down a system using a network request.	Force shutdown from a remote system.
SeRestorePrivilege	Required to perform restore operations. This privilege enables you to set any valid user or group SID as the owner of an object.	Restore files and directories.
SeSecurityPrivilege	Required to perform a number of security-related functions, such as controlling and viewing audit messages. This privilege identifies its holder as a security operator.	Manage auditing and security log.
SeServiceLogonRight	Required for an account to log on using the service logon type.	Log on using the service logon type.
SeShutdownPrivilege	Required to shut down a local system.	Shut down the system.
SeSystemEnvironmentPrivilege	Required to modify the nonvolatile RAM of systems that use this type of memory to store configuration information.	Modify firmware environment values.
SeSystemProfilePrivilege	Required to gather profiling information for the entire system.	Profile system performance.
SeSystemtimePrivilege	Required to modify the system time.	Change the system time.
SeTakeOwnershipPrivilege	Required to take ownership of an	Take ownership of files or

	object without being granted discretionary access. This privilege allows the owner value to be set only to those values that the holder may legitimately assign as the owner of an object.	other objects.
SeTcbPrivilege	This privilege identifies its holder as part of the trusted computer base. Some trusted protected subsystems are granted this privilege. This privilege is required to call the LogonUser function.	Act as part of the operating system.
SeUnsolicitedInputPrivilege	Required to read unsolicited input from a terminal device.	Read unsolicited input from a terminal device.

Example:

```
AddExtender("WWWNT34i.DLL")
wntPrivAdd("\\server","server\user","SeShutdownPrivilege")
Message("User", "Now has shut down privilege")
exit
```

See Also:

wntPrivDel, wntPrivGet, wntPrivList

wntPrivDel(server-name, user/group name, privilege)

Removes a privilege from a user or group.

Syntax:

wntPrivDel(server-name, user/group name, privilege)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) user/group name	the name of a user or group. If necessary, it can be fully qualified as "server\user" (or "server\group").
(s) privilege	a user right which may be removed. See below.

Returns:

(i)	1

This function requires Administrator privileges on the local workstation.

The following is a list of possible "privilege" values (although there may be others):

Privilege	Description	User Right
SeAssignPrimaryTokenPrivilege	Required to assign the primary token of a process.	Replace a process level token.
SeAuditPrivilege	Required to generate audit-log entries. Give this privileg to secure servers.	Generate security audits.
SeBackupPrivilege	Required to perform backup operations.	Back up files and directories.
SeBatchLogonRight	Required for an account to log on using the batch logon type	Log on using the batch logon
SeChangeNotifyPrivilege	Required to receive notifications of changes to files or directories. This privilege also causes the system to skip all traversal access checks. It is enabled by default for all users.	Bypass traverse checking.
SeCreatePagefilePrivilege	Required to create a paging file.	Create a pagefile.
SeCreatePermanentPrivilege	Required to create a permanent object.	Create permanent shared objects.
SeCreateTokenPrivilege	Required to create a primary token.	Create a token object.
SeDebugPrivilege	Required to debug a process.	Debug programs.
SeDenyBatchLogonRight	Explicitly denies an account the right to log on using the batch logon type	Denies an account the right to log on using the batch logon type

SeDenyInteractiveLogonRight	Explicitly denies an account the right to log on using the interactive logon type.	Denies an account the right to log on using the interactive logon type.
SeDenyNetworkLogonRight	Explicitly denies an account the right to log on using the network logon type.	Denies an account the right to log on using the network logon type.
SeDenyServiceLogonRight	Explicitly denies an account the right to log on using the service logon type.	Denies an account the right to log on using the service logon type.
SeIncreaseBasePriorityPrivilege	Required to increase the base priority of a process.	Increase scheduling priority.
SeIncreaseQuotaPrivilege	Required to increase the quota assigned to a process.	Increase quotas.
SeInteractiveLogonRight	Required for an account to log on using the interactive logon type.	Log on using the interactive logon
SeLoadDriverPrivilege	Required to load or unload a device driver.	Load and unload device drivers.
SeLockMemoryPrivilege	Required to lock physical pages in memory.	Lock pages in memory
SeMachineAccountPrivilege	Required to create a machine account.	Add workstations to domain.
SeNetworkLogonRight	Required for an account to log on using the network logon type.	Log on using the network logon type.
SeProfileSingleProcessPrivilege	Required to gather profiling information for a single process.	Profile single process.
SeRemoteShutdownPrivilege	Required to shut down a system using a network request.	Force shutdown from a remote system.
SeRestorePrivilege	Required to perform restore operations. This privilege enables you to set any valid user or group SID as the owner of an object.	Restore files and directories.
SeSecurityPrivilege	Required to perform a number of security-related functions, such as controlling and viewing audit messages. This privilege identifies its holder as a security operator.	Manage auditing and security log.
SeServiceLogonRight	Required for an account to log on using the service logon type.	Log on using the service logon type.
SeShutdownPrivilege	Required to shut down a local system.	Shut down the system.
SeSystemEnvironmentPrivilege	Required to modify the nonvolatile RAM of systems that use this type of memory to store configuration information.	Modify firmware environment values.
SeSystemProfilePrivilege	Required to gather profiling information for the entire system.	Profile system performance.
SeSystemtimePrivilege	Required to modify the system time.	Change the system time.
SeTakeOwnershipPrivilege	Required to take ownership of an	Take ownership of files or

	object without being granted discretionary access. This privilege allows the owner value to be set only to those values that the holder may legitimately assign as the owner of an object.	other objects.
SeTcbPrivilege	This privilege identifies its holder as part of the trusted computer base. Some trusted protected subsystems are granted this privilege. This privilege is required to call the LogonUser function.	Act as part of the operating system.
SeUnsolicitedInputPrivilege	Required to read unsolicited input from a terminal device.	Read unsolicited input from a terminal device.

Example:

```
AddExtender("WWWNT34i.DLL")
wntPrivDel("\\server","server\user","SeShutdownPrivilege")
Message("User", "Nolonger has shut down privilege")
exit
```

See Also:

wntPrivAdd, wntPrivGet, wntPrivList

wntPrivGet(server-name, user/group name, privilege)

Determines whether a user or group holds a privilege.

Syntax:

wntPrivGet(server-name, user/group name, privilege)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) user/group name	the name of a user or group. If necessary, it can be fully qualified as "server\user" (or "server\group").
(s) privilege	a user right to query. See below.

Returns:

(i)	**@TRUE** if the user holds the specified privilege **@FALSE** if not.

The following is a list of possible "privilege" values (although there may be others):

Privilege	Description	User Right
SeAssignPrimaryTokenPrivilege	Required to assign the primary token of a process.	Replace a process level token.
SeAuditPrivilege	Required to generate audit-log entries. Give this privileg to secure servers.	Generate security audits.
SeBackupPrivilege	Required to perform backup operations.	Back up files and directories.
SeBatchLogonRight	Required for an account to log on using the batch logon type	Log on using the batch logon
SeChangeNotifyPrivilege	Required to receive notifications of changes to files or directories. This privilege also causes the system to skip all traversal access checks. It is enabled by default for all users.	Bypass traverse checking.
SeCreatePagefilePrivilege	Required to create a paging file.	Create a pagefile.
SeCreatePermanentPrivilege	Required to create a permanent object.	Create permanent shared objects.
SeCreateTokenPrivilege	Required to create a primary token.	Create a token object.
SeDebugPrivilege	Required to debug a process.	Debug programs.
SeDenyBatchLogonRight	Explicitly denies an account the right to log on using the batch logon type	Denies an account the right to log on using the batch logon type

SeDenyInteractiveLogonRight	Explicitly denies an account the right to log on using the interactive logon type.	Denies an account the right to log on using the interactive logon type.
SeDenyNetworkLogonRight	Explicitly denies an account the right to log on using the network logon type.	Denies an account the right to log on using the network logon type.
SeDenyServiceLogonRight	Explicitly denies an account the right to log on using the service logon type.	Denies an account the right to log on using the service logon type.
SeIncreaseBasePriorityPrivilege	Required to increase the base priority of a process.	Increase scheduling priority.
SeIncreaseQuotaPrivilege	Required to increase the quota assigned to a process.	Increase quotas.
SeInteractiveLogonRight	Required for an account to log on using the interactive logon type.	Log on using the interactive logon
SeLoadDriverPrivilege	Required to load or unload a device driver.	Load and unload device drivers.
SeLockMemoryPrivilege	Required to lock physical pages in memory.	Lock pages in memory
SeMachineAccountPrivilege	Required to create a machine account.	Add workstations to domain.
SeNetworkLogonRight	Required for an account to log on using the network logon type.	Log on using the network logon type.
SeProfileSingleProcessPrivilege	Required to gather profiling information for a single process.	Profile single process.
SeRemoteShutdownPrivilege	Required to shut down a system using a network request.	Force shutdown from a remote system.
SeRestorePrivilege	Required to perform restore operations. This privilege enables you to set any valid user or group SID as the owner of an object.	Restore files and directories.
SeSecurityPrivilege	Required to perform a number of security-related functions, such as controlling and viewing audit messages. This privilege identifies its holder as a security operator.	Manage auditing and security log.
SeServiceLogonRight	Required for an account to log on using the service logon type.	Log on using the service logon type.
SeShutdownPrivilege	Required to shut down a local system.	Shut down the system.
SeSystemEnvironmentPrivilege	Required to modify the nonvolatile RAM of systems that use this type of memory to store configuration information.	Modify firmware environment values.
SeSystemProfilePrivilege	Required to gather profiling information for the entire system.	Profile system performance.
SeSystemtimePrivilege	Required to modify the system time.	Change the system time.
SeTakeOwnershipPrivilege	Required to take ownership of an	Take ownership of files or

	object without being granted discretionary access. This privilege allows the owner value to be set only to those values that the holder may legitimately assign as the owner of an object.	other objects.
SeTcbPrivilege	This privilege identifies its holder as part of the trusted computer base. Some trusted protected subsystems are granted this privilege. This privilege is required to call the LogonUser function.	Act as part of the operating system.
SeUnsolicitedInputPrivilege	Required to read unsolicited input from a terminal device.	Read unsolicited input from a terminal device.

Example:

```
AddExtender("WWWNT34i.DLL")
ret=wntPrivGet("\\server","server\user","SeShutdownPrivilege")
if ret==0
 Message("User", "Does NOT have shut down privilege")
else
 Message("User", "Does have shut down privilege")
endif
exit
```

See Also:

wntPrivAdd, wntPrivDel, wntPrivList

wntPrivList(server-name, user/group name)

Returns a list of priviliges granted to a user or group.

Syntax:

wntPrivList(server-name, user/group name)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) user/group name	the name of a user or group. If necessary, it can be fully qualified as "server\user" (or "server\group").

Returns:

(s) list	a tab-delimited list of privileges.

Example:

```
AddExtender("WWWNT34i.DLL")
list=wntPrivList("\\server","server\user")
AskItemList("List of privileges for user", list,@Tab,@Sorted,@Single)
exit
```

See Also:

wntPrivAdd, wntPrivDel, wntPrivGet

wntRasUserGet (server-name, user-name, request)

Gets RAS information for a user.

Syntax:
wntRasUserGet(server-name, user-name, request)

Parameters:

(s) server-name	is the UNC name of the primary or backup domain controller that has the user account database (eg, "\\MYSERVER"). If the machine on which the user account resides is a standalone NT workstation or server, you can specify the UNC name of that machine, or "" for the local computer.
(s) user-name	is the name of a user.
(i) request	specifies the information to be returned, and can be one of the following: 1 - Dial-in privilege or 2 - Callback phone number

Returns:

(i/s) Value — Returns one of the following, depending on the request.

1	(i) Dial-in privilege
2	(s) Callback phone number

"Privilege"
Dial-in privilege specifies both the dial-in permission and the callback mode, and will be one of the following values:

Value	Meaning
1	No dial-in permission / No callback privilege
2	No dial-in permission / Callback number preset by administratorr
4	No dial-in permission / Callback number specified by caller
9	Dial-in permission / No callback privilege
10	Dial-in permission / Callback number preset by administrator
12	Dial-in permission / Callback number specified

Note: **wntRasUserGet** and **wntRasUserSet** require RASSAPI.DLL.

Example:

```
AddExtender("WWWNT34i.DLL")
flags=wntRasUserGet("\\SERV","user1",1)
Message("RASFlags on User1", flags)
wntRasUserSet("\\SERV","user1","9","")
flags=wntRasUserGet("\\SERV","user1",1)
```

```
Message("RASFlags on User1", flags)
```

See Also:

wntRasUserSet

wntRasUserSet(server-name, user-name, privilege, phone-number)

Sets RAS information for a user.

Syntax:

wntRasUserSet (server-name,:user-name, privilege, phone-number)

Parameters:

(s) server-name	is the UNC name of the primary or backup domain controller that has the user account database (eg, "\\MYSERVER"). If the machine on which the user account resides is a standalone NT workstation or server, you can specify the UNC name of that machine, or "" for the local computer.
(s) user-name	is the name of a user.
(i) request	specifies the information to be returned, and can be one of the following:
(i) privilege	see below.
(s) phone-number	specifies an administrator-preset callback phone number, if appropriate. You can specify a blank string ("") to leave the currently-set number (if any) unchanged.

Returns:

(i)	Returns 1.

"Privilege"

Value	Meaning
1	No dial-in permission / No callback privilege
2	No dial-in permission / Callback number preset by administratorr
4	No dial-in permission / Callback number specified by caller
9	Dial-in permission / No callback privilege
10	Dial-in permission / Callback number preset by administrator
12	Dial-in permission / Callback number specified

Note: **wntRasUserGet** and **wntRasUserSet** require RASSAPI.DLL.

Example:

```
AddExtender("WWWNT34i.DLL")
flags=wntRasUserGet("\\SERV","user1",1)
Message("RASFlags on User1", flags)
wntRasUserSet("\\SERV","user1","9","")
flags=wntRasUserGet("\\SERV","user1",1)
Message("RASFlags on User1", flags)
```

See Also:

wntRasUserGet

wntRemoteTime(server-name, format)

Gets the time of day from a server.

Syntax:
wntRemoteTime(server-name, format)

Parameters:

(s) server-name	the UNC name of a server (eg, "\\SERVER1"), or a blank string ("") to indicate the local machine.
(i) format	specifies the format in which the time will be returned. See below.

Returns:

(s) server time	format of return value depends on which format was specified.

"format" can be one of the following values:

Value	Format
1	UTC/GMT time, in YYYY:MM:DD:hh:mm:ss format
2	local time (on local machine), in YYYY:MM:DD:hh:mm:ss format

Example:

```
AddExtender("WWWNT34i.DLL")
srvtime = wntRemoteTime( "\\server", 1)
message("Current server time", srvtime )
```

See Also:

wntResources(net-resource, scope, type, usage)

Itemizes network resources.

Syntax:
wntResources(net-resource, scope, type, usage)

Parameters:

(s) net-resource	a UNC (e.g., "\\FredsPC"), a domain (e.g., "SALES") or ("") for the root of the network.
(i) scope	see below.
(i) type	see below.
(i) usage	see below.

Returns:

(s)	a tab-delimited list of network resources.

This function returns a tab-delimited list of network resources which are located immediately below the specified 'net-resource'.

'Scope' can be one of the following:

Req #	Meaning
1	All currently connected resources
2	All resources on the network
3	All remembered (persistent) connections

'Type' can be one of the following:

Req #	Meaning
0	All resources
1	All disk resources
2	All print resources
3	All disk and print resources

'Usage' can be one of the following:

Req #	Meaning
0	All resources
1	All connectable resources
2	All container resources
3	All connectable and container resources

Note: 'usage' is ignored unless 'scope' == 2.

Example:

```
AddExtender("WWWNT34i.DLL")
servers = wntResources("OFFICE", 2, 1, 1)
server = AskItemList("Available servers", servers, @TAB,@SORTED,@SINGLE)
shares = wntResources(server, 2, 1, 1)
share = AskItemList("Shared directories on %server%", shares, @TAB, @SORTED,
@SINGLE)
```

See Also:

WntResources2

wntResources2(net-resource, scope, type, usage, provider)

Itemizes network resources.
(This functions supersedes wntResources.)

Syntax:
wntResources2(net-resource, scope, type, usage, provider)

Parameters:

(s) net-resource	a server UNC (e.g., "\\FredsPC"), a domain (e.g., "SALES"), or the name of a container object with a domain (e.g. "Directory Services"). Cannot be the a network provider. Specify an empty string ("") for the root of the network or the root of the network provider.
(i) scope	see below.
(i) type	see below.
(i) usage	see below.
(s) provider	name of the network provider under which 'net resource' is located.

Returns:

(s)	a tab-delimited list of network resources located immediately below the specified 'net-resource'.

Optional Returns:

(s)	If you specify blank strings ("") for both 'net-resource' and 'provider', this function will return a tab delimited list of network providers.
(s)	If you specify a blank string ("") for net-resource' and a non-blank 'provider', this function will return a tab delimited list of network resources located beneath 'provider'.

'**Scope**' can be one of the following:

Req #	Meaning
1	All currently connected resources
2	All resources on the network
3	All remembered (persistent) connections

'**Type**' can be one of the following:

Req #	Meaning
0	All resources
1	All disk resources

2	All print resources
3	All disk and print resources

Note: 'net-resource', 'usage', and 'provider' are ignored unless 'scope' is 2. That is, you can itemize all current or persistent connections network-wide, eg:

```
wntResources2("", 1, 0, 1, "")
```

Or you can itemize all available resources under a specific 'net-resource', eg:

```
wntResources2("\\JELLO", 2, 0, 1, "Microsoft Windows Network")
```

But you can not do both at the same time, ie, you can not itemize current connections under a specific 'net-resource':

Req #	**Meaning**
0	All resources
1	All connectable resources
2	All container resources
3	All connectable and container resources

Note: A "connectable" object is an object that can be connected to, such as a directory or a printer. A "container" object is an object that contains other objects, such as a domain or a server.

'**Provider**' is the name of the network provider under which 'net-resource' is located. Most commonly 'provider' is "Microsoft Windows Network" (for Windows NT) or "NetWare Services" (for NetWare).

Example:

```
AddExtender("WWWNT32I.DLL")
;;examples for the different types of resources.
providers = wntResources2("", 2, 0, 2, "")
domains = wntResources2("", 2, 0, 2, "Microsoft Windows Network")
servers = wntResources2("OFFICE", 2, 0, 2, "Microsoft Windows Network")
shares = wntResources2("\\SERVER", 2, 0, 1, "Microsoft Windows Network")
```

See Also:

WntResources

wntRunAsUser(domain/server, user-name, password, login-type, flags)

Run as a different user. (Requires NT 4.0 or newer)

Syntax:

wntRunAsUser(domain/server, user-name, password, login-type, flags)

Parameters:

(s) domain/server	the name of the domain or server on which the specified user account resides, or "." to indicate that the local account database should be searched for the user, or "" to indicate that the local account database and (then) any trusted domain account databases should be searched for the user.
(s) user-name	the name of the user to run as.
(s) password	the specified user's login password.
(i) login-type	see below.
(i) flags	see below.

Returns:

(i)	always 1.

login-type can be one of the following:

Type	Meaning
2	**Interactive login:** This logon type is intended for users who will be interactively using the machine, such as a user being logged on by a terminal server, remote shell, or similar process. This logon type has the additional expense of caching logon information for disconnected operation, and is therefore inappropriate for some client/server applications, such as a mail server.
3	**Network login :** This is the fastest logon path, but there is one main limitation. Any programs lauched by the WIL script will not be able to access network resources.
4	**Batch login:** This logon type is intended for batch servers, where processes may be executing on behalf of a user without their direct intervention.
5	**Service login:** Indicates a service-type logon. The account provided must have the service privilege enabled.

flag can be one of the following:

Flag	Meaning

0	Default.
1	Allow new child processes to inherit security privileges.
2	Load user profile into HKEY_USERS\<User SID>

This function causes the specified user to be impersonated for the duration of the currently-running instance of the WIL Interpreter.

Note: The specified user must have the appropriate user rights assigned to be able to log in as a batch job or service. In order to use this function, the currently-logged-in user must have the following rights assigned:

"Act as part of the operating system"
"Increase quotas"
"Replace a process level token"

These user rights can be set in one of two ways:
(**Note:** these privileges will not take affect until next login)
1) Using the function wntPrivAdd

Act as part of the operating system
```
wntPrivAdd("\\server","server\user","SeTcbPrivilege")
```

Increase quotas
```
wntPrivAdd("\\server","server\user","SeIncreaseQuotaPrivilege")
```

Replace a process level token
```
wntPrivAdd("\\server","server\user","SeAssignPrimaryTokenPrivilege")
```

2) Or manually set user rights from the "Policy" menu in the NT User Manager (note that the "Show Advanced User Rights" option in the dialog box must be checked).

Note: If you do not want password information displayed to a user in case of an error, make sure to create a variable for the password, as follows:
```
pswd="boom";  notice a variable has been created for the password
ret = wntRunAsUser( "", "Joe", pswd, 3, 0)
```

Do not hard code the password or use variable substitution. If you do the following, the password could be displayed to the user.
```
pswd="boom";  notice a variable has been created for the password
; WRONG this next line uses variable substition
ret = wntRunAsUser( "", "Joe", "%pswd%", 3, 0)
```

Note: To end the impersonation in the script, you can specify a user name of "" .
```
wntRunAsUser( "", "", "", 0, 0)
```

Example:
```
AddExtender("WWWNT34i.DLL")
```

```
curuser=wntGetUser(@DEFAULT)
Message("Current user:",curuser)

;run as new user
user="Administrator"
pswd="goat"
ret = wntRunAsUser( "", user, pswd, 2, 0)

newuser=wntGetUser(@DEFAULT)
Message("Running as new user:",newuser)
exit
```

See Also:

wntUserInfo, wntUserAdd

wntServerInfo(server-name, request)

Returns information about a server.

Syntax:

wntServerInfo(server-name, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(i) request	specifies the information about "server-name" to be returned. See below.

Returns:

(i)	depends on request.

Security Requirements

Only the Administrators or Accounts Operators local group, or those with Communication, Print, or Server operator group membership can successfully execute the **wntServerInfo** function.

Req#	Description
1	(i) major version
2	(i) minor version
3	(i) type (see wntServiceAt for a list of server types)
4	(s) comment
5	(i) maximum users who can log on
6	(i) auto-disconnect time, in minutes, or -1 if no auto-disconnect
7	(i) hidden to other computers in domain? (0 = visible, 1 = hidden)
8	(i) network announce rate, in seconds
9	(i) delta value by which announce rate can vary, in milliseconds
10	(s) path to user directories

Example:

```
AddExtender("WWWNT34i.DLL")
info=wntServerInfo( "", 10)
Message("The path to user directories",info)
exit
```

See Also:

wntServerList

wntServerList(server-name, domain-name, server-type)

Lists servers in a domain.

Syntax:

wntServerList(server-name, domain-name, server-type)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) domain-name	the name of the domain which will be used (e.g., "SALES"), or ("") for the primary domain.
(i) server-type	identifies the type of servers which will be examined. See below.

Returns:

(i)	a tab-delimited list of server UNC names (e.g., "\\MYSERVER").

Note: An NT workstation can be considered to be a "server".

Server-type

Specify **-1** for all servers. Or, specify one or more of the following flags, combined using the binary OR ("|") operator.

Req#	Server Type
1	All workstations
2	All servers
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT/Windows 2000 (workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT/Windows 2000 Non-domain controller
65536	Server that can run the browser service
131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser
1048576	OSF Server

2097152	VMS Server
4194304	Windows 95 or later
8388608	Root of DFS tree
16777216	Server clusters available in the domain
33554432	Terminal Server
268435456	IBM DSS (Directory & Security Services) or equivalent
536870912	return list for alternate transport
1073741824	Servers maintained by the browser
-2147483648	Primary Domain

Note: This function can take a while to run, depending on how many servers are in the domain. Also, it will only return the names of servers which it is able to access, which requires that the user have browse access to their service control managers.

Example:

```
AddExtender("WWWNT34i.DLL")
;return a list of all Windows NT (either workstation or server)
servers = wntServerList("","",4096)
AskItemList("list of Windows NT workstations or
servers",servers,@tab,@sorted,@single)
exit
```

See Also:

wntServiceAt, wntServiceInf

wntServerType(server-name)

Returns a server's platform.

Syntax:
wntServerType(server-name)

Parameters:

(s) server-name	the UNC name of a server (eg, "\\SERVER1"), or a blank string ("") to indicate the local machine.

Returns:

(i)	See below.

Returns one of the following values:

Value	Meaning
---	---
0	Invalid server name
1	Other
2	Windows for Workgroups
3	Windows 95 or later
4	Windows NT

Example:

```
; This example returns a servers platform

AddExtender("WWWNT34i.DLL")
type=wntServerType("\\Server1")
switch type
 case 0
   message("Error","invalid server name")
   break
 case 1
   message("Server Type is:","Other")
   break
 case 2
   message("Server Type is:","Windows for Workgroups")
   break
 case 3
   message("Server Type is:","Windows 95 or later")
   break
 case 4
   message("Server Type is:","Windows NT")
   break
EndSwitch
```

See Also:

w95ServerType

wntServiceAt(server, domain, server-type, service-name, flags)

Lists all servers in a domain which contain a specified service.

Syntax:
wntServiceAt(server, domain, server-type, service-name, flags)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) domain	the name of the domain which will be used (e.g., "SALES"), or ("") for the primary domain.
(i) server-type	identifies the type of servers which will be examined. See below.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies information on the service being looked for. See below.

Returns:

(i)	a tab-delimited list of server UNC names (e.g., "\\MYSERVER").

Note: An NT workstation can be considered to be a "server".

Server-type
Specify **-1** for all servers. Or, specify one or more of the following flags, combined using the binary OR ("|") operator.

Req#	Server Type
1	All LAN Manager workstation
2	All LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT (either workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT Non-DC server
65536	Server that can run the browser service

131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser
4194304	Windows 95 or newer
-2147483648	Domain announcement

Service-Name

'Service name' is the name of a service (e.g., "Spooler") or driver (e.g., "Atdisk"). The name can be specified either as the "display name" which is listed in Control Panel (name-type = 0) or the "service name" which is the actual registry key for the service (name-type = 1000). The SDK documentation describes them as:

DisplayName = a string that is to be used by user interface programs to identify the service.
ServiceName = a string that names a service in a service control manager database.

So, the following two commands will yield identical results:

```
servers = wntServiceAt("", "", -1, "Browser", 101)  ; display name
servers = wntServiceAt("","", -1,"Computer Browser", 1001) ;service name
```

Flags

'Flags' specifies information on the service being looked for. It consists of one entry from each of the following three groups, added together:

Req#	service type
1	services
2	drivers
3	both

Req#	service state
100	active services
200	inactive services
300	both

Name type indicates what the **'service-name'** parameter represents.

Req#	name type
0	display name (the name shown in Control Panel)
1000	service name (the actual registry key name)

Note: This function can take a while to run, depending on how many servers are in the domain. Also, it will only return the names of servers which it is able to access, which requires that the user have browse access to their service control managers.

Example:

```
;return a list of all servers running the "Spooler" service
```

```
servers = wntServiceAt("", "", -1, "Spooler", 101)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)

;return a list of all NT machines with an "Atdisk" driver installed
servers = wntServiceAt("", "", 4096, "Atdisk", 302)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)
```

wntServiceInf

Returns information about a server's type.

Syntax:

wntServiceInf(server-name)

Parameters:

(s) server-name	the UNC name of a server (eg, "\\SERVER1"), or a blank string ("") to indicate the local machine.

Returns:

(i)	a bitmask indicating the type of server, or 0 on error. The individual flag bits in the bitmask can be extracted using the binary AND ("&") operator. See below.

Return Values	Server Type
1	All LAN Manager workstation
2	All LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT (either workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT Non-DC server
65536	Server that can run the browser service
131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser
1048576	Unknown service
2097152	Unknown service
4194304	Windows 95 or newer
-2147483648	Domain announcement

Example:

```
AddExtender("WWWNT34i.DLL")
servertype=wntServiceInf("\\SERVER1")
title=strcat("Server Type : ",servertype)
if servertype == 0
  Message("Error","There was an Error with the function wntServiceInf")
Else
```

```
   if (servertype & 4194304)
      message(title, "Windows 95 or newer")
   Else
      if (servertype & 128)
         message(title, "Novell server")
      Else
         If ( servertype & (8|16))
             message(title,"Primary or Backup Domain Controller")
         Else
             Message(title, "Other")
         EndIf
      Endif
   Endif
Endif
```

See Also:

wntServerType

wntShareAdd(server-name, resource, share-name,object-type, max-users)

Shares a resource.

Syntax:

wntShareAdd(server-name, resource, share-name,share-type, max-users)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	identifies the object to be shared. Resource can be a directory name (e.g., "c:\util"), a printer object (e.g., "HP LaserJet III"), or the name of a sharable device.
(s) share-name	name by which other users will access the resource.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(i) max-users	the maximum number of users allowed, or -1 for the highest possible number.

Returns:

(s)	1.

If the 'share-type' is 1 for printer both the 'server-name' and 'max-users' are ignored. Set 'server-name' to a blank string ("") and 'max-users' to -1.

Example:

```
; This example adds a share called "Public" to the C:\TEMP
; directory.  It sets max-users to -1 to provide unlimited
; concurrent access to the directory.
AddExtender("WWWNT34i.DLL")
wntShareAdd("","C:\TEMP","Public",0,-1)
```

See Also:

wntShareDel, wntShareSet

wntShareDel(server-name, resource/share-name, share-type)

UN-shares a resource.

Syntax:
wntShareDel(server-name, resource/share-name, share-type)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.

Returns:

(s)	**@TRUE** if the share was deleted; **@FALSE** if there was no share with the specified name.

If 'share-type' = 1 (printer), then 'resource/share-name' should specify the object's resource name (e.g., "HP LaserJet III"). Otherwise, it should specify the object's share name.

Example:

```
; This example removes the "Public" share from the system.
AddExtender("WWWNT34i.DLL")
rslt=wntShareDel("","Public",0)
if rslt
     Message("wntShareDel","Share PUBLIC deleted")
else
     Message("wntShareDel","Share PUBLIC not found")
endif
```

See Also:
wntShareAdd, wntShareSet

wntShareInfo(server-name, resource/share-name, share-type, request)

Returns information about a shared resource.

Syntax:
wntShareInfo(server-name, resource/share-name, share-type, request)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(i) request	specifies the information to be returned. See below.

Returns:

(s)/(i)	a string or integer, depending on "request".

"request" specifies the information to be returned, and can be one of the following:

-1	(i) **@TRUE** if share exists, **@FALSE** if it doesn't.
0	(s) share name
1	(s) resource
2	(s) comment
3	(s) location
6	(i) share type
8	(i) max users
9	(i) current users

Note: This function can only be performed by members of the Administratorsor Account Operators local group, or those with Communication, Print, or Server operator group membership.

Example:

```
AddExtender("WWWNT34i.DLL")
comment = wntShareInfo("", "Share", 0, 2)
Message("Comment for 'Share'", comment)
```

See Also:
wntShareAdd, wntShareSet

wntShareList(server-name, share-type, flags)

Returns a list of shares on a server.

Syntax:

wntShareList(server-name, share-type, flags)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(i) share-type	a bitmask specifying the type(s) of shares that should be returned. See below.
(i) flags	reserved for future use and should be set to 0.

Returns:

(s)	a tab-delimited list of shared resources on a server, including hidden shares.

"share-type" can be set to 0 to return all shares, or it can be one or more of the following values combined with the bitwise OR ('|') operator:

Value	Share type
1	Disk drive
2	Print queue
4	Communication device
8	Interprocess communication (IPC)
16	Administrative or special share (eg: C$, D$, ADMIN$, IPC$)

Example:

```
AddExtender("WWWNT34i.DLL")
list=wntShareList("\\SERVER", 1 | 16, 0)
AskItemList("List of shares",list, @TAB, @UNSORTED, @SINGLE)
exit
```

See Also:

wntShareAdd, wntShareDel, wntShareInfo,

wntShareSet(server-name, resource/share-name, share-type, comment, description)

Sets additional share information for a resource.

Syntax:

wntShareSet(server-name, resource/share-name, share-type, comment, location)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(s) comment	text string used to describe the share or a blank string ("") if no comment is desired.
(s) location	a text string that can be used to describe the location of the printer, 'share-type' 1, or a blank string ("") if no location description is desired. A blank string ("") must be specified for any other 'share-type'.

Returns:

(s) 1.

Example:

```
; This example adds a comment to the Public share.  Other network users can
; see the comment when they browser the network.
AddExtender("WWWNT34i.DLL")
wntShareSet("","Public",0,"Public Access Directory on my machine","")
```

See Also:

wntShareAdd, wntShareDel

wntShareUsers(server-name, share/computer-name, share-type, format, flags)

Returns a list of users connected to a share or server..

Syntax:

wntShareUsers(server-name, share/computer-name, share-type, format, flags)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) share/computer-name	specifies either a share name (eg, "PUBLIC") on "server-name", or a computer name (eg, "\\FRED"). See explanation below.
(i) share-type	a bitmask specifying the type(s) of shares that should be returned. See below.
(i) format	specifies the information that will be returned for each connection. See Below.
(i) flags	reserved for future use and should be set to 0.

Returns:

(s) .

"share-type" is a bitmask specifying the type(s) of shares that should be returned if "share/computer-name" specifies a computer name. It can be set to 0 to return all shares, or it can be one or more of the following values combined with the bitwise OR ('|') operator:

Value	Share type
1	Disk drive
2	Print queue
4	Communication device
8	Interprocess communication (IPC)
16	Administrative or special share (eg: C$, D$, ADMIN$, IPC$)

"format" specifies the information that will be returned for each connection. It can be one of the following values:

Value	Return format
1	username
2	netname
3	username|netname

This function can return either a list of users connected to a share, or a list of computers connected to a server. If "share/computer-name" specifies a share name, then the function will return a list of users connected to that share. If "share/computer-name" specifies a computer name, then the function will return a list of shares on "server-name" (filtered by "share-type") to which that computer

is connected.

This function returns a tab-delimited list of user names and/or net names, depending on "format". The values returned are as follows:

username:
If "server-name" is running with user-level security, then "username" will be the name of the user who made the connection. If "server-name" is running with share-level security, then "username" will be the computer that made the connection.

netname:
"netname" is the opposite of "share/computer-name". In other words, if "share/computer-name" specifies a share name, then "netname" will be the name of the computer connected to that share, and if "share/computer-name" specifies a computer name, then "netname" will be the name of the share to which that computer is connected.

Example:

```
AddExtender("WWWNT34i.DLL")
list=wntShareUsers("", "PUBLIC", 1, 1, 0)
AskItemList("list of users connected to PUBLIC",list, @TAB, @UNSORTED, @SINGLE)
exit
```

See Also:

wntShareAdd, wntShareDel, wntShareInfo, wntShareList, wntShareSet

wntShutdown(computer-name, message, timeout, force, reboot)

Shuts down (and optionally restarts) a computer..

Syntax:

wntShutdown(computer-name, message, timeout, force, reboot)

Parameters:

(s) computer-name	the name of an NT machine (eg, "FRED" or "\\FRED"), or a blank string ("") to indicate the local machine.
(s) message	a message to be displayed in the shutdown dialog box, or a blank string ("") if no message is desired.
(i) timeout	the amount of time, in seconds, that a shutdown dialog box should be displayed on the specified computer before the computer is shut down. While the dialog is displayed, a countdown timer shows the number of seconds remaining, and when it reaches 0 the computer is shut down. "timeout" can be set to 0, in which case no dialog is displayed and the computer is shut down immediately.
(i) force	If "force" is **@TRUE**, then applications with unsaved data will be forcibly closed. If "force" is **@FALSE**, then the user will be prompted to close such applications, and if the user cancels (or doesn't respond) then the computer will not be shut down.
(i) reboot	If "reboot" is **@TRUE**, the computer will be automatically restarted after shutting down. If "reboot" is **@FALSE**, the system flushes all caches to disk, clears the screen, and displays a message indicating that it is safe to power down.

Returns:

(i)	Always returns 1 on success, or an error.

Note: In order to shut down a remote computer, the caller must have the privilege "Force shutdown from a remote system" (SeRemoteShutdownPrivilege).

Example:

```
AddExtender("WWWNT34i.DLL")
;Reboot the local machine
wntShutdown( "", "", 0, @FALSE, @TRUE)
exit
```

wntSvcCfgGet(server, service-name, flags, request)

Gets a configuration parameter for a service.

Syntax:
wntSvcCfgGet(server, service-name, flags, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)
(i) request	specifies the parameter to be returned (see below).

Returns:

(s/i)	Returns a string or integer value.

"**request**" specifies the parameter to be returned, and can be one of the following:

Request	Return Value
0	(i):ServiceType
1	(i):StartType
2	(i):ErrorControl
3	(s):BinaryPathName
4	(s):LoadOrderGroup
5	(i):TagId
6	(s):Dependencies
7	(s):ServiceStartName
8	(unused)
9	(s):DisplayName

Further information on these values follows:

ServiceType:
One of the following service type flags to indicate the type of service. In addition, for a SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag might be set, indicating that the service process can interact with the desktop:

Value	Name	Meaning

1	SERVICE_KERNEL_DRIVER	Windows NT device driver.
2	SERVICE_FILE_SYSTEM_DRIVER	Windows NT file system driver.
16	SERVICE_WIN32_OWN_PROCESS	Win32 service that runs in its own process.
32	SERVICE_WIN32_SHARE_PROCESS	Win32 service that shares a process with other services.
256	SERVICE_INTERACTIVE_PROCESS	Win32 service process that can interact with the desktop.

StartType:
Specifies when to start the service. One of the following values is specified:

Value	Name	Meaning
0	SERVICE_BOOT_START	Device driver started by the operating system loader. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
1	SERVICE_SYSTEM_START	Device driver started by the IoInitSystem function. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
2	SERVICE_AUTO_START	Device driver or Win32 service started by the service control manager automatically during system startup.
3	SERVICE_DEMAND_START	Device driver or Win32 service started by the service control manager when a process calls the StartService function.
4	SERVICE_DISABLED	Device driver or Win32 service that can no longer be started.

ErrorControl:
Specifies the severity of the error if this service fails to start during startup, and determines the action taken by the startup program if failure occurs. One of the following values can be specified:

Value	Name	Meaning
0	SERVICE_ERROR_IGNORE	The startup (boot) program logs the error but continues the startup operation.
1	SERVICE_ERROR_NORMAL	The startup program logs the error and displays a message box pop-up but continues the startup operation.
2	SERVICE_ERROR_SEVERE	The startup program logs the error. If the last-known good configuration is being started, the startup operation

		continues. Otherwise, the system is restarted with the last-known-good configuration.
3	SERVICE_ERROR_CRITICAL	The startup program logs the error, if possible. If the last-known good configuration is being started, the startup operation fails. Otherwise, the system is restarted with the last-known good configuration.

BinaryPathName:
The fully qualified path to the service binary file.

LoadOrderGroup:
The load ordering group of which this service is a member. If a blank string, the service does not belong to a group. The registry has a list of load ordering groups located at:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder
```

The startup program uses this list to load groups of services in a specified order with respect to the other groups in the list. You can place a service in a group so that another service can depend on the group. The order in which a service starts is determined by the following criteria:

1. The order of groups in the registrys load-ordering group list. Services in groups in the load-ordering group list are started first, followed by services in groups not in the load-ordering group list and then services that do not belong to a group.

2. The services dependencies listed in the "Dependencies" parameter and the dependencies of other services dependent on the service.

TagId:
Specifies a unique tag value for this service in the group specified by the "LoadOrderGroup" parameter. A value of zero indicates that the service has not been assigned a tag. You can use a tag for ordering service startup within a load order group by specifying a tag order vector in the registry located at:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\GroupOrderList
```

Tags are only evaluated for SERVICE_KERNEL_DRIVER and SERVICE_FILE_SYSTEM_DRIVER type services that have SERVICE_BOOT_START or SERVICE_SYSTEM_START start types.

Dependencies:
A tab-delimited list of names of services or load ordering groups that must start before this service. If a blank string, the service has no dependencies. If a group name is specified, it must be prefixed by a '+' character to differentiate it from a service name, because services and service groups share the same name space. Dependency on a service means that this service can only run if the service it depends on is running. Dependency on a group means that this service can run if at least one member of the group is running after an attempt to start all members of the group.

ServiceStartName:
If the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS, this name is the account name in the form of "DomainName\Username", which the service process

will be logged on as when it runs. If the account belongs to the built-in domain, ".\Username" can be specified. If a blank string, the service will be logged on as the LocalSystem account.

If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this name is the Windows NT driver object name (that is, \FileSystem\Rdr or \Driver\Xns) which the input and output (I/O) system uses to load the device driver. If a blank string, the driver is run with a default object name created by the I/O system based on the service name.

DisplayName:
String that is to be used by user interface programs to identify the service. This string has a maximum length of 256 characters. The name is case-preserved in the service control manager. Display name comparisons are always case-insensitive.

Example:

```
AddExtender("WWWNT34i.DLL")
ret=wntSvcCfgGet("", "Spooler", 0, 1)
Message("Start type for Spooler is:",ret)
exit
```

See Also:
wntSvcCfgSet, wntSvcStart

wntSvcCfgSet(server, service-name, flags, request, value)

Changes a configuration parameter for a service.

Syntax:
wntSvcCfgSet(server, service-name, flags, request, value)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)
(i) request	specifies the parameter to be changed (see below).
(s/i) value	specifies the new value for the parameter.

Returns:

(i)	always 1.

"**request**" specifies the parameter to be modified, and can be one of the following:

Request	Return Value
0	(i):ServiceType
1	(i):StartType
2	(i):ErrorControl
3	(s):BinaryPathName
4	(s):LoadOrderGroup
5	(i):TagId
6	(s):Dependencies
7	(s):ServiceStartName or (s):ServiceStartName|Password
9	(s):DisplayName

Further information on these values follows:

ServiceType:
Service type are the flags to indicate the type of service. In addition, for a SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag might be set, indicating that the service process can interact with the desktop:

Value	Name	Meaning
1	SERVICE_KERNEL_DRIVER	Windows NT device driver.
2	SERVICE_FILE_SYSTEM_DRIVER	Windows NT file system driver.
16	SERVICE_WIN32_OWN_PROCESS	Win32 service that runs in its own process.
32	SERVICE_WIN32_SHARE_PROCESS	Win32 service that shares a process with other services.

In addition to the SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag may be set, indicating that the service process can interact with the desktop:

256	SERVICE_INTERACTIVE_PROCESS	(Optional) Win32 service process that can interact with the desktop. **Note:** you must also specify a SERVICE_WIN32 service type.

StartType:
Specifies when to start the service. One of the following values is specified:

Value	Name	Meaning
0	SERVICE_BOOT_START	Device driver started by the operating system loader. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
1	SERVICE_SYSTEM_START	Device driver started by the IoInitSystem function. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
2	SERVICE_AUTO_START	Device driver or Win32 service started by the service control manager automatically during system startup.
3	SERVICE_DEMAND_START	Device driver or Win32 service started by the service control manager when a process calls the StartService function.
4	SERVICE_DISABLED	Device driver or Win32 service that can no longer be started.

ErrorControl:
Specifies the severity of the error if this service fails to start during startup, and determines the action taken by the startup program if failure occurs. One of the following values can be specified:

Value	Name	Meaning

0	SERVICE_ERROR_IGNORE	The startup (boot) program logs the error but continues the startup operation.
1	SERVICE_ERROR_NORMAL	The startup program logs the error and displays a message box pop-up but continues the startup operation.
2	SERVICE_ERROR_SEVERE	The startup program logs the error. If the last-known good configuration is being started, the startup operation continues. Otherwise, the system is restarted with the last-known-good configuration.
3	SERVICE_ERROR_CRITICAL	The startup program logs the error, if possible. If the last-known good configuration is being started, the startup operation fails. Otherwise, the system is restarted with the last-known good configuration.

BinaryPathName:
The fully qualified path to the service binary file.

LoadOrderGroup:
The load ordering group of which this service is a member. If a blank string, the service does not belong to a group. The registry has a list of load ordering groups located at:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder
```

The startup program uses this list to load groups of services in a specified order with respect to the other groups in the list. You can place a service in a group so that another service can depend on the group. The order in which a service starts is determined by the following criteria:

1. The order of groups in the registrys load-ordering group list. Services in groups in the load-ordering group list are started first, followed by services in groups not in the load-ordering group list and then services that do not belong to a group.

2. The services dependencies listed in the "Dependencies" parameter and the dependencies of other services dependent on the service.

TagId:
Specifies a unique tag value for this service in the group specified by the "LoadOrderGroup" parameter. A value of zero indicates that the service has not been assigned a tag. You can use a tag for ordering service startup within a load order group by specifying a tag order vector in the registry located at:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\GroupOrderList
```

Tags are only evaluated for SERVICE_KERNEL_DRIVER and SERVICE_FILE_SYSTEM_DRIVER type services that have SERVICE_BOOT_START or SERVICE_SYSTEM_START start types.

Dependencies:

A tab-delimited list of names of services or load ordering groups that must start before this service. If a blank string, the service has no dependencies. If a group name is specified, it must be prefixed by a '+' character to differentiate it from a service name, because services and service groups share the same name space. Dependency on a service means that this service can only run if the service it depends on is running. Dependency on a group means that this service can run if at least one member of the group is running after an attempt to start all members of the group.

ServiceStartName:
If the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS, this name is the account name in the form of "DomainName\Username", which the service process will be logged on as when it runs. If the account belongs to the built-in domain, ".\Username" can be specified. To log on as the LocalSystem account, specify "LocalSystem".

If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this name is the Windows NT driver object name (that is, \FileSystem\Rdr or \Driver\Xns) which the input and output (I/O) system uses to load the device driver.

You can specify both a ServiceStartName and a Password together, in the form "ServiceStartName|Password". If ServiceStartName does not contain a '\', then '.\' will automatically be prepended, unless ServiceStartName == "LocalSystem".

Password:
Password to the account name specified by the "ServiceStartName" parameter if the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS. If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this parameter is ignored.

DisplayName:
String that is to be used by user interface programs to identify the service. This string has a maximum length of 256 characters. The name is case-preserved in the service control manager. Display name comparisons are always case-insensitive.

Example:

```
AddExtender("WWWNT34i.DLL")
wntSvcCfgSet("", "Spooler", 0, 1, 2)
Message("Done","Changed configuration parameter for Spooler 'Start-
Type'[SERVICE_AUTO_START]")
exit
```

See Also:
wntSvcCfgGet

wntSvcControl(server-name, service-name, flags, control-code)

Stops or controls a service.

Syntax:
wntSvcControl(server, service-name, flags, control-code)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)
(i) control code	specifies the operation to be performed. See below.

Returns:

(i)	returns 1.

Control codes:

Value	Name	Meaning
1	SERVICE_CONTROL_STOP	Requests the service to stop.
2	SERVICE_CONTROL_PAUSE	Requests the service to pause.
3	SERVICE_CONTROL_CONTINUE	Requests the paused service to resume.
4	SERVICE_CONTROL_INTERROGATE	Requests the service to update immediately its current status information to the service control manager.

Or it can be a user-defined control code in the range of 128 to 255, inclusive.

Example:

```
wntSvcControl("", "Spooler", 0, 1)  ; stop the Spooler service
```

See Also:

wntSvcStart

wntSvcCreate(server-name, string-values, numeric-values, dependencies, reserved)

Adds a service to the service control manager database.

Syntax:
wntSvcCreate(server-name, string-values, numeric-values, dependencies, reserved)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) string-values	a tab-delimited or vertical-bar ('|') delimited list of string properties for the service being created. See below.
(s) numeric-values	a tab-delimited or vertical-bar ('|') delimited list of numeric properties for the service being created. See below.
(s) dependencies	a tab-delimited or vertical-bar ('|') delimited list of names of services or load ordering groups that must start before this service. **Note: There should be no spaces around the delimiters.**
(s) reserved	reserved for future use, and should be set to "".

Returns:

(i)	always 1.

"string-values" is a is a tab-delimited or vertical-bar ('|') delimited list of string properties for the service being created, in the following format:

```
"ServiceName | DisplayName | BinaryPathName | LoadOrderGroup | ServiceStartName
| Password"
```

(Spaces around the delimiters are permitted, but not necessary.)

ServiceName:
String that specifies the name of the service to install. The maximum string length is 256 characters. The service control manager database preserves the case of the characters, but service name comparisons are always case insensitive. Forward-slash (/) and back-slash (\) are invalid service name characters.

DisplayName:
String that is to be used by user interface programs to identify the service. This string has a maximum length of 256 characters. The name is case-preserved in the service control manager. Display name comparisons are always case-insensitive.

BinaryPathName:
The fully qualified path to the service binary file.

LoadOrderGroup:
The load ordering group of which this service is a member. If a blank string, the service does not

belong to a group. The registry has a list of load ordering groups located at:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder
```

The startup program uses this list to load groups of services in a specified order with respect to the other groups in the list. You can place a service in a group so that another service can depend on the group. The order in which a service starts is determined by the following criteria:

1. The order of groups in the registrys load-ordering group list. Services in groups in the load-ordering group list are started first, followed by services in groups not in the load-ordering group list and then services that do not belong to a group.

2. The services dependencies listed in the "Dependencies" parameter and the dependencies of other services dependent on the service.

ServiceStartName:
If the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS, this name is the account name in the form of "DomainName\Username", which the service process will be logged on as when it runs. If the account belongs to the built-in domain, ".\Username" can be specified. To log on as the LocalSystem account, specify "LocalSystem".

If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this name is the Windows NT driver object name (that is, \FileSystem\Rdr or \Driver\Xns) which the input and output (I/O) system uses to load the device driver.

You can specify both a ServiceStartName and a Password together, in the form "ServiceStartName|Password". If ServiceStartName does not contain a '\', then '.\' will automatically be prepended, unless ServiceStartName == "LocalSystem".

Password:
Password to the account name specified by the "ServiceStartName" parameter if the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS. If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this parameter is ignored.

"numeric-values" is a tab-delimited or vertical-bar ('|') delimited list of numeric properties for the service being created, in the following format:

```
"ServiceType | StartType | ErrorControl"
```

(Spaces around the delimiters are permitted, but not necessary.)

ServiceType:
Service type are the flags to indicate the type of service. In addition, for a SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag might be set, indicating that the service process can interact with the desktop:

Value	Name	Meaning
1	SERVICE_KERNEL_DRIVER	Windows NT device driver.
2	SERVICE_FILE_SYSTEM_DRIVER	Windows NT file system

		driver.
16	SERVICE_WIN32_OWN_PROCESS	Win32 service that runs in its own process.
32	SERVICE_WIN32_SHARE_PROCESS	Win32 service that shares a process with other services.

In addition to the SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag may be set, indicating that the service process can interact with the desktop:

256	SERVICE_INTERACTIVE_PROCESS	(Optional) Win32 service process that can interact with the desktop. **Note:** you must also specify a SERVICE_WIN32 service type.

StartType:

Specifies when to start the service. One of the following values is specified:

Value	Name	Meaning
0	SERVICE_BOOT_START	Device driver started by the operating system loader. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
1	SERVICE_SYSTEM_START	Device driver started by the IoInitSystem function. This value is valid only if the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER.
2	SERVICE_AUTO_START	Device driver or Win32 service started by the service control manager automatically during system startup.
3	SERVICE_DEMAND_START	Device driver or Win32 service started by the service control manager when a process calls the StartService function.
4	SERVICE_DISABLED	Device driver or Win32 service that can no longer be started.

ErrorControl:

Specifies the severity of the error if this service fails to start during startup, and determines the action taken by the startup program if failure occurs. One of the following values can be specified:

Value	Name	Meaning
0	SERVICE_ERROR_IGNORE	The startup (boot) program logs the error but continues the startup operation.

1	SERVICE_ERROR_NORMAL	The startup program logs the error and displays a message box pop-up but continues the startup operation.
2	SERVICE_ERROR_SEVERE	The startup program logs the error. If the last-known good configuration is being started, the startup operation continues. Otherwise, the system is restarted with the last-known-good configuration.
3	SERVICE_ERROR_CRITICAL	The startup program logs the error, if possible. If the last-known good configuration is being started, the startup operation fails. Otherwise, the system is restarted with the last-known good configuration.

Notes:

wntSvcCreate prepends '.\' to the specified ServiceStartName, if it doesn't already contain a contain a '\' (unless it's "LocalSystem").

A service that is running under the LocalSystem account cannot access network resources.

To have a service access the desktop, that service must log on as the LocalSystem account, and be allowed to interact with the desktop.

Example:

```
AddExtender("WWWNT34i.DLL")
ServiceName="Computer Browser"
DisplayName="Browser"
BinaryPathName="d:\my share\browserservice.exe"
LoadOrderGroup=""
ServiceStartName="LocalSystem"
Password=""
createstring=Strcat(ServiceName,"|",DisplayName,"|",BinaryPathName,"|",LoadOrderGro
up,"|",ServiceStartName,"|",Password)
ServiceType=256|16
StartType=3
ErrorControl=0
createflags=strcat(ServiceType,"|",StartType,"|",ErrorControl)
wntSvcCreate("", createstring, createflags, "","")
Message("Done","Added a service to the service control manager database.")
exit
```

See Also:

wntSvcCfgSet

wntSvcDelete(server, service-name, flags)

Marks a service for deletion from the service control manager database.

Syntax:

wntSvcDelete(server, service-name, flags)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(s)service-name	the name of a service.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)

Returns:

(i)	**@TRUE** on success, or **@FALSE** if the specified service has already been marked for deletion.

Example:

```
AddExtender("WWWNT34i.DLL")
DisplayName="Browser"
wntSvcDelete( "", DisplayName, 0)
Message("Done","Marked Broswer service for deletion from the service control
manager database.")
exit
```

See Also:

wntSvcCreate

wntSvcStart(server, service-name, flags, params, delimiter)

Starts a service.

Syntax:

wntSvcStart(server, service-name, flags, params, delimiter)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)
(s) params	specifies a delimited list of parameters to be passed to the service, or a blank string ("") for no parameters. You can select any character you wish to be the delimiter, and must set the "delimiter" parameter accordingly. Note: driver services do not receive parameters.
(s) delimiter	specifies a single character, indicating the delimiter used for "params". For example, if "params" specifies a space-delimited list, then "delimiter" should be set to " ". If "params" is a blank string, then "delimiters" is ignored and can be set to a blank string.

Returns:

(i)	returns 1.

Example:

```
AddExtender("WWWNT34i.DLL")
wntSvcStart("", "Spooler", 0, "", "")  ; start the Spooler service
```

See Also:

wntSvcControl, wntSvcStatus

wntSvcStatus(server, service-name, flags, request)

Returns status information on a service.

Syntax:
wntSvcStatus(server, service-name, flags, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies the type of name that "service-name" represents: 0 display name (the name shown in Control Panel) 1000 service name (the actual registry key name)
(i) request	specifies the information to be returned. See below.

Returns:

(i)	an integer.

"request" specifies the information to be returned, and can be one of the following:

0 ServiceExist

1 ServiceType

2 CurrentState

3 ControlsAccepted

4 Win32ExitCode

5 ServiceSpecificExitCode

6 CheckPoint

7 WaitHint

Further information on these values follows:

ServiceExist:

Returns **@FALSE** if a service doesn't exist. (other request #'s will return an error if the service doesn't exist). If the service does exist, it will return a non-zero value

ServiceType:

The value returned includes one of the following service type flags to indicate the type of service. In addition, for a SERVICE_WIN32 service, the SERVICE_INTERACTIVE_PROCESS flag might be set, indicating that the service process can interact with the desktop.

Value	Name	Meaning
1	SERVICE_KERNEL_DRIVER	A service type flag that indicates a Windows NT device driver.
2	SERVICE_FILE_SYSTEM_DRIVER	A service type flag that indicates a Windows NT file system driver.
16	SERVICE_WIN32_OWN_PROCESS	A service type flag that indicates a Win32 service that runs in its own process.
32	SERVICE_WIN32_SHARE_PROCESS	A service type flag that indicates a Win32 service that shares a process with other services.
256	SERVICE_INTERACTIVE_PROCESS	A flag that indicates a Win32 service process that can interact with the desktop.

CurrentState:

Indicates the current state of the service. One of the following values is specified:

Value	Name	Meaning
1	SERVICE_STOPPED	The service is not running.
2	SERVICE_START_PENDING	The service is starting.
3	SERVICE_STOP_PENDING	The service is stopping.
4	SERVICE_RUNNING	The service is running.
5	SERVICE_CONTINUE_PENDING	The service continue is pending.
6	SERVICE_PAUSE_PENDING	The service pause is pending.
7	SERVICE_PAUSED	The service is paused.

ControlsAccepted:

Specifies the control codes that the service will accept and process. A user interface process can control a service by specifying a control command in the ControlService function. By default, all services accept the SERVICE_CONTROL_INTERROGATE value. Any or all of the following flags can be specified to enable the other control codes.

Value	Name	Meaning
1	SERVICE_ACCEPT_STOP	The service can be stopped. This enables the SERVICE_CONTROL_STOP value.
2	SERVICE_ACCEPT_PAUSE_CONTINUE	The service can be paused and continued. This enables the SERVICE_CONTROL_PAUSE and SERVICE_CONTROL_CONTINUE values.
4	SERVICE_ACCEPT_SHUTDOWN	The service is notified when system shutdown occurs. This enables the

system to send a SERVICE_CONTROL_SHUTDOWN value to the service. The ControlService function cannot send this control code.

Win32ExitCode:

Specifies a Win32 error code that the service uses to report an error that occurs when it is starting or stopping. To return an error code specific to the service, the service must set this value to ERROR_SERVICE_SPECIFIC_ERROR to indicate that the dwServiceSpecificExitCode member contains the error code. The service should set this value to NO_ERROR when it is running and on normal termination.

ServiceSpecificExitCode:

Specifies a service specific error code that the service returns when an error occurs while the service is starting or stopping. This value is ignored unless the dwWin32ExitCode member is set to ERROR_SERVICE_SPECIFIC_ERROR.

CheckPoint:

Specifies a value that the service increments periodically to report its progress during a lengthy start, stop, or continue operation. For example, the service should increment this value as it completes each step of its initialization when it is starting up. The user interface program that invoked the operation on the service uses this value to track the progress of the service during a lengthy operation. This value is not valid and should be zero when the service does not have a start, stop, or continue operation pending.

WaitHint:

Specifies an estimate of the amount of time, in milliseconds, that the service expects a pending start, stop, or continue operation to take before the service makes its next call to the SetServiceStatus function with either an incremented dwCheckPoint value or a change in dwCurrentState. If the amount of time specified by dwWaitHint passes, and dwCheckPoint has not been incremented, or dwCurrentState has not changed, the service control manager or service control program can assume that an error has occurred.

Example:

```
AddExtender("WWWNT34i.DLL")
state = wntSvcStatus("", "Spooler", 0, 2)  ; get the current state of the Spooler
service

Switch state
  case 1  ; if it's stopped, start it
    wntSvcStart("", "Spooler", 0, "", "")
    break
  case 7  ; if it's paused, resume it
    wntSvcControl("", "Spooler", 0, 3)
    break
EndSwitch
```

See Also:

wntSvcStart, wntSvcControl

wntUserAdd(server-name)

Adds a user account.

Syntax:

wntUserAdd(server-name)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.

Returns:

(i)	always 1.

Before calling this function, you must use **wntUserAddDat** to set parameters for the new user account. At a minimum, you must set the "name" element. The other elements will receive default values.

Calling this function does not reset elements in the user parameter structure. So, you can set various elements, add a user, then change just the "name" element and add another user. All other elements will retain their previous values. To clear all elements, call **wntUserAddDat** specifying blank strings for "variable" and "value".

Example:

```
AddExtender("WWWNT34i.DLL")
wntUserAddDat("name", "jdoe")
wntUserAddDat("full_name", "John Doe")
wntUserAddDat("flags", 1)
wntUserAddDat("acct_expires", "0000:00:00:00:00:00")
wntUserAdd("")
exit

;*OR*

;example to add a computer to the domain
AddExtender("WWWNT34i.DLL")
server="\\server"
computername="work001"
NametoAdd=strUpper(strcat(computername,"$"))
wntUserAddDat("name", NametoAdd)
wntUserAddDat("flags", 4096+1);
wntUserAddDat("acct_expires", "0000:00:00:00:00:00")
wntUserAdd(server)
exit
```

See Also:

wntUserAddDat

wntUserAddDat(element, value)

Sets parameter information for **wntUserAdd**. (This function sets values for elements in a user parameter structure, which is used by the wntUserAdd function.)

Syntax:

Parameters:

(s) element	see below.
(s / i) value	see below.

Returns:

(i)	**@TRUE** on success, **@FALSE** if there was a problem.

Element

Can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):

Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"password" (s):

The password for the user specified in the "name" element. The length cannot exceed 256 bytes. By convention, Windows NT limits the length of passwords to 14 characters. This convention allows LAN Manager, Windows 3.x, Windows for Workgroups 3.x, and Windows 95 clients to access a Windows NT server using the account.

"home_dir" (s):

Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):

Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):

Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
1	Normal Account	This flag is **REQUIRED** for new accounts.
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored in Windows NT.
16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.
64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.
65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE _ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST _ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST _ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST _ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

"script_path" (s):
Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):
Points to a 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.

"usr_comment" (s):
Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

> **Note:** "value" must contain a 4-digit year, and must appear in the precise format "YYYY:MM:DD:hh:mm:ss" (ie, exactly 19 characters long, with colons in exactly the right positions). To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

If "variable" and "value" are both set to blank strings (""), all values will be cleared from the user parameter structure.

You can specify a value of "*NULL*" to set a string element to a NULL pointer, which is not the same as a NULL string (").

wntUserAdd can also add a workstation to a domain.

The server-name parameter requires a valid servername.
The user-name parameter, should be a computer (workstation) name (in upper case) with a trailing '$'.
You should specify password, as the workstation name without the '$'.

Here is an example:

```
WktoAdd="WORK001$" ; make sure uppercase with trailing $
wntUserAddDat("name", NametoAdd)
wntUserAddDat("password", strsub(WktoAdd,0,strlen(NametoAdd)-1))
; 4096 - This is a computer account for a Windows NT
; Workstation or Windows NT Server that is a member of
; this domain.
wntUserAddDat("flags", 4096+1)
wntUserAdd("\\server")
```

Example:

```
AddExtender("WWWNT34i.DLL")
wntUserAddDat("name", "jdoe")
wntUserAddDat("full_name", "John Doe")
wntUserAddDat("flags", 1)
wntUserAddDat("acct_expires", "0000:00:00:00:00:00")
wntUserAdd("")
exit

;*OR*

;example to add a computer to the domain
AddExtender("WWWNT34i.DLL")
server="\\server"
computername="work001"
NametoAdd=strUpper(strcat(computername,"$"))
wntUserAddDat("name", NametoAdd)
wntUserAddDat("flags", 4096+1);
wntUserAddDat("acct_expires", "0000:00:00:00:00:00")
wntUserAdd(server)
exit
```

See Also:

wntUserAdd

wntUserDel(server-name, user-name)

Deletes a user account.

Syntax:
wntUserDel(server-name, user-name)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(s) user-name	is the name of the user to be deleted.

Returns:

(i)	always 1.

wntUserDel can also remove a workstation from a domain. The server-name parameter requires a valid servername. The user-name parameter, should be a computer (workstation) name (in upper case) with a trailing '$'. Here is an example:

```
wntUserDel("\\server", "COMPUTERNAME$")
```

Example:

```
AddExtender("WWWNT34i.DLL")
wntUserDel("\\Server1", "jdoe")
exit

;*OR*

;Remove a workstation from a domain
AddExtender("wwwnt34i.dll")

PC = "Server"
Wks = "Wks"
Wks = StrUpper(Wks)

AccountDel = AskYesNo("Attention", StrCat("Are you sure to remove ", Wks, " from ",
PC, "?"))

If AccountDel == @Yes
    GoSub DelAccount
Else
    Display(3, "Note", "User not sure, Program aborted")
Endif
Exit

:DelAccount
WksAccount = wntUserExist(StrCat("\\", PC), StrCat(Wks, "$"))
If WksAccount == @True
  Del = wntUserDel(StrCat("\\", PC), StrCat(Wks, "$"))
Else
```

```
  Message("ERROR", StrCat(Wks, " doesn't exist on ", PC))
Endif
Return
```

See Also:
wntUserAdd, wntUserAddDat

wntUserExist(server-name, user-name)

Determines whether a user exists.

Syntax:

wntUserExist(server-name, user-name)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or a blank string ("") to indicate the current machine.
(s) user-name	is the name of a user who may have an account on "server-name".

Returns:

(i)	**@TRUE** if the specified user exists, **@FALSE** otherwise.

Example:

```
AddExtender("WWWNT34i.DLL")
f=wntUserExist("\\Server1", "jdoe")
if f == @True
   Message("User Exist?","Yes, user exists")
else
   Message("User Exist?","No, user does NOT exist")
endif
exit
```

See Also:

wntGetUser, wntUserAdd

wntUserGetDat(server-name, user-name, element)

Returns parameter information for a user account.

Syntax:
wntUserGetDat(server-name, user-name, element)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(s) user-name	name of a user who has an account on "server-name".
(s) element	specifies the element to be returned. See below.

Returns:

(i)	Returns a string or integer value, depending on "element"..

<u>Element</u>
Can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):
Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"home_dir" (s):
Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):
Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):
Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
1	Normal Account	This flag is REQUIRED for new accounts
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored in Windows NT.
16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.

64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.
65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE _ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST _ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST _ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST _ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

"script_path" (s):
Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):
Points to a 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.

"usr_comment" (s):
Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

****
Note: The following elements cannot be set using **wntUserAddDat** or **wntUserSetDat**.
****

"user_id" (i):
User's RID (relative identifier).

"primary_group_id" (i):
RID (relative ID) of the user's primary global group. You can determine a group's RID using **wntGroupInfo** with request = 2.

"password_age" (i):
Specifies the number of seconds elapsed since the password was last changed.

"priv" (s):
One of the following values, specifying the level of privilege assigned the user:

Value	Name	Meaning
0	USER_PRIV_GUEST	Guest
1	USER_PRIV_USER	User
2	USER_PRIV_ADMIN	Administrator

"auth_flags" (i):
Specifies the user's operator privileges, based on local group membership. This element can be any of the following values:

Value	Name	Meaning

1	AF_OP_PRINT	The print operator privilege is enabled.
2	AF_OP_COMM	The communications operator privilege is enabled.
4	AF_OP_SERVER	The server operator privilege is enabled.
8	AF_OP_ACCOUNTS	The accounts operator privilege is enabled.

parms" (i):
A string that is set aside for use by applications. This can be a string of any length, or a blank string. Microsoft products use this element to store user configuration information.

"units_per_week" (i):
Specifies the number of equal-length time units into which the week is divided in order to compute the length of the bit string in "logon_hours". This value must be 168 (ie, hours per week) for LAN Manager 2.0. For Windows NT services, this value must be 7, 168, or 10080 (days, hours, or minutes per week, respectively).

"usri3_bad_pw_count" (i):
Specifies 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. This element is maintained separately on each Backup Domain Controller (BDC) in the domain. To get an accurate value, each BDC in the domain must be queried, and the largest value is used.

"num_logons" (i):
Specifies the number of successful times the user tried to log on to this account. A value of -1 indicates that the value is unknown. This element is maintained separately on each Backup Domain Controller (BDC) in the domain. To get an accurate value, each BDC in the domain must be queried, and the largest value is used.

"logon_server" (s):
A string that contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes ("\\"). A server name of an asterisk ("*") indicates that the logon request can be handled by any logon server. A blank string indicates that requests are sent to the domain controller. For Windows NT Servers, this will always return "*".

"last_logon" (s):
Last logon time, in "YYYY:MM:DD:hh:mm:ss" format. These times are maintained separately on each Backup Domain Controller (BDC) in the domain. To get an accurate value, each BDC in the domain must be queried, and the largest value is used.

The return value will always contain a 4-digit year. If no value is available, "0000:00:00:00:00:00" is returned.

"last_logoff" (s):
Last logoff time, in "YYYY:MM:DD:hh:mm:ss" format. These times are maintained separately on each Backup Domain Controller (BDC) in the domain. To get an accurate value, each BDC in the domain must be queried, and the largest value is used.

The return value will always contain a 4-digit year. If no value is available, "0000:00:00:00:00:00" is returned.

Example:

```
user="joe"
theflags=wntUserGetDat("\\SERVER",user,"flags")
if theflags & 2
   Message(user,"Account Disabled")
endif
if theflags & 32
   Message(user,"Password not required")
endif
```

See Also:

wntUserAddDat

wntUserInfo(request)

Returns information about the currently logged-on user.

Syntax:
wntUserInfo(request)

Parameters:

(i) request	specifies the information to be returned. See below.

Returns:

(s)	returns a string.

"request" specifies the information to be returned, and can be one of the following:

Value	Name	Meaning
0	user name	name of the user currently logged on to the workstation
1	logon domain	domain name of the user account of the user currently logged on to the workstation
2	other domains	space-delimited list of other LAN Manager domains browsed by the workstation
3	logon server	name of the computer that authenticated the server

Example:

```
AddExtender("WWWNT34i.DLL")
server = wntUserInfo(3)
Message("Logon server", server)
```

See Also:

wntUserProps

wntUserList(server-name, account-type)

Lists users with accounts on an NT server.

Syntax:
wntUserList(server-name, account-type)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or "" for the local computer.
(i) account-type	specifies the types of user accounts to include. (see below)

Returns:

(s)	Returns a tab-delimited list of user names.

Note: to list users in a domain, specify a domain controller for that domain as "server-name".

"**account-type**" specifies the types of user accounts to include. You can specify 0 to include all account types, or specify one or more of the following flags, combined using the bitwise OR ('|') operator:

Flag	Meaning
0	all account types.
1	Include local user account data on a domain controller.
2	Include global user account data on a computer.
8	Include domain trust account data on a domain controller.
16	Include workstation or member server account data on a domain controller.
32	Include domain controller account data on a domain controller.

Example:

```
AddExtender("WWWNT34i.DLL")
users = wntUserList("",0)
AskItemList("List of users with accounts on an NT server.",
users,@TAB,@SORTED,@SINGLE)
exit
```

See Also:
wntUserInfo

wntUserProps

Returns information about a network user.

Syntax:
wntUserProps(server-name, user-name, request)

Parameters:

(s) server-name	is the name of the server on which the function will execute, or a blank string ("") to indicate the current machine.
(s) user-name	is the name of a user who has an account on "servername".
(i) request	specifies the information to be returned, and can be one of the following:

Returns:

(s)	returns a string. See below.

Note: the user-name parameter can be obtained, using the function **wntGetUser**.

Value	Returns
0	Username
1	Full name
2	Description
3	User profile path
4	Login script name
5	Home directory
6	Home directory logon drive
7	Privilege level ("GUEST", "USER", or "ADMIN")

Example:

```
AddExtender("WWWNT34i.DLL")
server="\\Server"
user=wntGetUser(@default)

response=wntUserProps(server, user, 0)
message("The user name is:",response)

response=wntUserProps(server, user, 1)
message("The full name is:",response)

response=wntUserProps(server, user, 2)
message("The description is:",response)

response=wntUserProps(server, user, 3)
message("The user profile path is:",response)

response=wntUserProps(server, user, 4)
message("The login script name is:",response)
```

```
response=wntUserProps(server, user, 5)
message("The home directory is:",response)

response=wntUserProps(server, user, 6)
message("The home directory logon drive is:",response)

response=wntUserProps(server, user, 7)
message("The Privilege level is:",response)

message("Finished","Retrieving information about a network user")
exit
```

See Also:

wntUserInfo

wntUserRename

Renames a user account.

Syntax:

wntUserRename(server-name, old-username, new-username)

Parameters:

(s) server-name	UNC name of the server on which the function will execute (eg, "\\MYSERVER"), or " " for the local computer.
(s) old-username	An existing account name.
(s) new-username	New name to be given to the account.

Returns:

(i)	**@TRUE** on success, **@FALSE** if there was a problem.

See Also:

wntUserInfo

wntUserSetDat

Modifies parameter information for a user account.

Syntax:

wntUserSetDat(server-name, user-name, element, value)

Parameters:

(s) server-name	is the name of the server on which the function will execute, or a blank string ("") to indicate the current machine.
(s) user-name	is the name of a user who has an account on "servername".
(s) element	See below.
(s) value	See below.

Returns:

(s)	**@TRUE** on success, **@FALSE** if there was a problem.

<u>Element</u>

Can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):

Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"password" (s):

The password for the user specified in the "name" element. The length cannot exceed 256 bytes. By convention, Windows NT limits the length of passwords to 14 characters. This convention allows LAN Manager, Windows 3.x, Windows for Workgroups 3.x, and Windows 95 clients to access a Windows NT server using the account.

"home_dir" (s):

Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):

Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):

Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
1	Normal Account	This flag is REQUIRED for new accounts
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored in Windows NT.

16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.
64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.
65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST_ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

Examples:

```
; ADD additional flags to an account
; and preserve original user settings
AddExtender("WWWNT34i.DLL")
flags=wntUserGetDat("","joed","flags")
flags=flags | 2
wntUserSetDat("","joed","flags",flags)
exit

; REMOVE a single flag from an account
; and preserve all other user settings
AddExtender("WWWNT34i.DLL")
flags=wntUserGetDat("","joed","flags")
flags=flags & (~2); Note: tilde not minus
wntUserSetDat("","joed","flags",flags)
exit
```

"script_path" (s):

Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):

Points to a 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.

"usr_comment" (s):

Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

> **Note:** "value" must contain a 4-digit year, and must appear in the precise format "YYYY:MM:DD:hh:mm:ss" (ie, exactly 19 characters long, with colons in exactly the right positions). To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"primary_group_id" (i):
RID (relative ID) of the user's primary global group. You can determine a group's RID using **wntGroupInfo** with request = 2. **Note:** This element cannot be set using **wntUserAddDat**.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

If "variable" and "value" are both set to blank strings ("""), all values will be cleared from the user

parameter structure.

You can specify a value of "*NULL*" to set a string element to a NULL pointer, which is not the same as a NULL string ("").

Note: This function cannot rename an account. It can only modify parameter information for a user account. To rename a account use the function wntUserRename.

Example:

```
; ADD additional flags to an account
; and preserve original user settings
AddExtender("WWWNT34i.DLL")
flags=wntUserGetDat("","joed","flags")
flags=flags | 2
wntUserSetDat("","joed","flags",flags)
exit

; REMOVE a single flag from an account
; and preserve all other user settings
AddExtender("WWWNT34i.DLL")
flags=wntUserGetDat("","joed","flags")
flags=flags & (~2); Note: tilde not minus
wntUserSetDat("","joed","flags",flags)
exit
```

See Also:

wntUserAddDat, wntUserRename, wntUserInfo

wntUserSidChk

Checks SID's in an access token.

Syntax:
wntUserSidChk(request, SID, reserved-1, reserved-2, reserved-3)

Parameters:
(i) request

Request	Meaning
0	Check if the specified SID is present and enabled in the token.

(s) SID — specifies a security identifier, in string form (eg, "S-1-5-32-544", which is the well-known SID of the local Administrators group).

(i) reserved-1 — 0.

(i) reserved-2 — 0.

(i) reserved-3 — 0.

Returns:
(i) **@TRUE** or **@FALSE**

Example:

```
AddExtender("WWWNT34i.DLL")
SID = "S-1-5-32-544"
wntUserSidChk( 0, SID, 0, 0, 0)
exit
```

wntVersion( )

Returns the version of this Extender DLL.

Syntax:
wntVersion( )

Parameters:
none

Returns:
(i) the version of number of this extender Dll.

This function is used to check the version number of this Dll in cases where older DLL's exist and alternate processing is desirable. Version numbers of newer versions will be larger than that of older versions.

Example:

```
AddExtender("WWWNT34i.DLL")
a=wntVersion()
Message("Dll Version",a)
```

wntWtsUserGet

Gets user information from an NT Terminal Server. This function requires WTSAPI32.DLL to be present.

Syntax:

wntWtsUserGet(server-name, user-name, request)

Parameters:

(s) server-name	the name of a Windows-based Terminal Server or domain controller, or "" to indicate the Terminal Server on which your application is running.
(s) user-name	a user name.
(i) request	specifies the information to get. (see below).

Returns:

(s/i)	Returns a string or integer, depending on "request".

"request" specifies the information to get, and can be one of the following:

Request	Type	Meaning
1	InitialProgram	(s) path of the initial program that Terminal Server runs when the user logs on.
2	WorkingDirectory	(s) path of the user's working directory.
3	InheritInitialProgram	(i) flag for inheriting the initial program.

Value	Meaning
0	The client cannot specify the initial program. The WTSUserConfigInitialProgram string indicates the initial program. If you specify a user's initial program, that's the only program they can run; Terminal Server logs off the user when the user exits that program.
1	The client can specify the initial program.

4	AllowLogonTerminalServer	(i) flag that indicates whether the user account is permitted to log on to a Terminal Server.

Value	Meaning
0	The user cannot logon.
1	The user can logon.

5	TimeoutSettingsConnections	(i) specifies the maximum connection duration, in milliseconds. One minute (60000 milliseconds) before the connection timeout interval expires, the user is notified of the pending disconnection. The user's session is disconnected or terminated depending on the WTSUserConfigBrokenTimeoutSettings value. Every time the user logs on, the timer is reset. A value of zero indicates the connection timer is disabled.
6	TimeoutSettingsDisconnections	(i) specifies the maximum duration, in milliseconds, that a Terminal Server retains a disconnected session before the logon is terminated. A value of zero indicates the disconnection timer is disabled.
7	TimeoutSettingsIdle	(i) specifies the maximum idle time, in milliseconds. If there is no keyboard or mouse activity for the specified interval, the user's session is disconnected or terminated depending on the WTSUserConfigBrokenTimeoutSettings value. A value of zero indicates the idle timer is disabled.
8	DeviceClientDrives	(i) (Citrix ICA clients only) A flag that indicates whether the Terminal Server automatically reestablishes client drive mappings at logon.

Value	Meaning
0	The server does not automatically connect to previously mapped client drives.
1	The server automatically connects to previously mapped client drives at logon.

9	DeviceClientPrinters	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the Terminal Server automatically reestablishes

		client printer mappings at logon.

Value	Meaning
0	The server does not automatically connect to previously mapped client printers.
1	The server automatically connects to previously mapped client printers at logon.

10	DeviceClientDefaultPrinter	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the client printer is the default printer.

Value	Meaning
0	The client printer is not the default printer.
1	The client printer is the default printer.

11	BrokenTimeoutSettings	(i) flag that indicates what happens when the connection or idle timers expire or when a connection is lost due to a connection error.

Value	Meaning
0	The session is disconnected.
1	The session is terminated.

12	ReconnectSettings	(i) flag that indicates how a disconnected session for this user can be reconnected.

Value	Meaning
0	The user can log on to any client computer to reconnect to a disconnected session. Note that sessions started at clients other than the system console cannot be connected to the system console, and sessions started at the system console cannot be disconnected.
1	The user can reconnect to a disconnected session by logging on to the client computer used to establish the disconnected session. If the user logs on from a different client computer, the user gets a new logon session.

13	ModemCallbackSettings	(i) (Citrix ICA clients only): A value that indicates the configuration for dialup connections in which the Terminal Server hangs up and then calls back the client to establish the connection.

Value	Meaning
0	Callback connections are disabled.

1	The server prompts the user to enter a phone number and calls the user back at that phone number. You can use the WTSUserConfigModemCallbackPhoneNumber value to specify a default phone number.
2	The server automatically calls the user back at the phone number specified by the WTSUserConfigModemCallbackPhoneNumber value.

14	ModemCallbackPhoneNumber	(s) (Citrix ICA clients only): A string containing the phone number to use for callback connections.
15	ShadowingSettings	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the user session can be shadowed. Shadowing allows a user to remotely monitor the on-screen operations of another user.

Value	**Meaning**
0	The session cannot be shadowed. (Disable)
1	The session can be shadowed. Enable input, notify
2	Enable input, no notify
3	Enable no input, notify
4	Enable no input, no notify

16	TerminalServerProfilePath	(s) string containing the path of the user's profile for Terminal Server logon.
17	TerminalServerHomeDir	(s) string containing the path of the user's home directory for Terminal Server logon. This string can specify a local path or a UNC path (\\machine\share\path). See WTSUserConfigfTerminalServerRemoteHomeDir.
18	TerminalServerHomeDirDrive	(s) string containing a drive letter to which the UNC path specified in the WTSUserConfigTerminalServerHomeDir string is mapped. See WTSUserConfigfTerminalServerRemoteHomeDir.
19	TerminalServerRemoteHomeDi	(i) flag that indicates whether the user's home

r	directory for Terminal Server logon is a local path or a mapped drive letter.

Value	Meaning
0	The WTSUserConfigTerminalServerHomeDir string contains the local path of the user's Terminal Server logon home directory.
1	The WTSUserConfigTerminalServerHomeDir string contains the UNC path of the user's Terminal Server logon home directory, and the WTSUserConfigTerminalServerHomeDirDrive string contains a drive letter to which the UNC path is mapped.

Notice: The use of milliseconds,rather than minutes, for the following flags: TimeoutSettingsConnections, TimeoutSettingsDisconnections, TimeoutSettingsIdle

[1 second = 1000 milliseconds]
[1 minute = 60000 milliseconds]

To convert from milliseconds to minutes, do the following:
mins= milliseconds / 60000
milliseconds=mins * 60000

Example:

```
AddExtender("WWWNT34i.DLL")
ProfilePath = wntWtsUserGet("\\WTS1","bob", 16)
Message("The path of the user's profile for Terminal Server logon",ProfilePath)
```

See Also:

wntWtsUserSet

wntWtsUserSet

Modifies user information on an NT Terminal Server.

Syntax:
wntWtsUserSet(server-name, user-name, request, value)

Parameters:

(s) server-name	the name of a Windows-based Terminal Server or domain controller, or "" to indicate the Terminal Server on which your application is running.
(s) user-name	a user name.
(i) request	specifies the information to modify. (see below).
(s/i) value	the new value to be set.

Returns:

(i)	always 1.

"request" specifies the information to modify, and can be one of the following:

Request	Type	Meaning
1	InitialProgram	(s) path of the initial program that Terminal Server runs when the user logs on.
2	WorkingDirectory	(s) path of the user's working directory.
3	InheritInitialProgram	(i) flag for inheriting the initial program.

Value	Meaning
0	The client cannot specify the initial program. The WTSUserConfigInitialProgram string indicates the initial program. If you specify a user's initial program, that's the only program they can run; Terminal Server logs off the user when the user exits that program.
1	The client can specify the initial program.

4	AllowLogonTerminalServer	(i) flag that indicates whether the user account is permitted to log on to a Terminal Server.

Value	Meaning
0	The user cannot logon.
1	The user can logon.

5	TimeoutSettingsConnections	(i) specifies the maximum connection duration, in milliseconds. One minute (60000 milliseconds) before the connection timeout interval expires, the user is notified of the pending disconnection. The user's session is disconnected or terminated depending on the WTSUserConfigBrokenTimeoutSettings value. Every time the user logs on, the timer is reset. A value of zero indicates the connection timer is disabled.
6	TimeoutSettingsDisconnections	(i) specifies the maximum duration, in milliseconds, that a Terminal Server retains a disconnected session before the logon is terminated. A value of zero indicates the disconnection timer is disabled.
7	TimeoutSettingsIdle	(i) specifies the maximum idle time, in milliseconds. If there is no keyboard or mouse activity for the specified interval, the user's session is disconnected or terminated depending on the WTSUserConfigBrokenTimeoutSettings value. A value of zero indicates the idle timer is disabled.
8	DeviceClientDrives	(i) (Citrix ICA clients only) A flag that indicates whether the Terminal Server automatically reestablishes client drive mappings at logon.

Value	Meaning
0	The server does not automatically connect to previously mapped client drives.
1	The server automatically connects to previously mapped client drives at logon.

9	DeviceClientPrinters	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the Terminal Server automatically reestablishes client printer mappings at

logon.

Value	Meaning
0	The server does not automatically connect to previously mapped client printers.
1	The server automatically connects to previously mapped client printers at logon.

10	DeviceClientDefaultPrinter	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the client printer is the default printer.

Value	Meaning
0	The client printer is not the default printer.
1	The client printer is the default printer.

11	BrokenTimeoutSettings	(i) flag that indicates what happens when the connection or idle timers expire or when a connection is lost due to a connection error.

Value	Meaning
0	The session is disconnected.
1	The session is terminated.

12	ReconnectSettings	(i) flag that indicates how a disconnected session for this user can be reconnected.

Value	Meaning
0	The user can log on to any client computer to reconnect to a disconnected session. Note that sessions started at clients other than the system console cannot be connected to the system console, and sessions started at the system console cannot be disconnected.
1	The user can reconnect to a disconnected session by logging on to the client computer used to establish the disconnected session. If the user logs on from a different client computer, the user gets a new logon session.

13	ModemCallbackSettings	(i) (Citrix ICA clients only): A value that indicates the configuration for dialup connections in which the Terminal Server hangs up and then calls back the client to establish the connection.

Value	Meaning
0	Callback connections are disabled.
1	The server prompts the user to enter a

		phone number and calls the user back at that phone number. You can use the WTSUserConfigModemCallbackPhoneNumber value to specify a default phone number.
		2 The server automatically calls the user back at the phone number specified by the WTSUserConfigModemCallbackPhoneNumber value.
	14 ModemCallbackPhoneNumber	(s) (Citrix ICA clients only): A string containing the phone number to use for callback connections.
	15 ShadowingSettings	(i) (RDP 5.0 clients and Citrix ICA clients): A flag that indicates whether the user session can be shadowed. Shadowing allows a user to remotely monitor the on-screen operations of another user.

Value	Meaning
0	The session cannot be shadowed. (Disable)
1	The session can be shadowed. Enable input, notify
2	Enable input, no notify
3	Enable no input, notify
4	Enable no input, no notify

16	TerminalServerProfilePath	(s) string containing the path of the user's profile for Terminal Server logon.
17	TerminalServerHomeDir	(s) string containing the path of the user's home directory for Terminal Server logon. This string can specify a local path or a UNC path (\\machine\share\path). See WTSUserConfigfTerminalServerRemoteHomeDir.
18	TerminalServerHomeDirDrive	(s) string containing a drive letter to which the UNC path specified in the WTSUserConfigTerminalServerHomeDir string is mapped. See WTSUserConfigfTerminalServerRemoteHomeDir.

Notice: The use of milliseconds,rather than minutes, for the following flags: TimeoutSettingsConnections, TimeoutSettingsDisconnections, TimeoutSettingsIdle

[1 second = 1000 milliseconds]
[1 minute = 60000 milliseconds]

To convert from milliseconds to minutes, do the following:

mins= milliseconds / 60000
milliseconds=mins * 60000

Example:

```
AddExtender("WWWNT34i.DLL")
wntWtsUserSet("\\WTS1","bob", 1, "notepad.exe")
Message("InitialProgram set","")
```

See Also:

wntWtsUserGet

w9xAccessAdd(server-name, resource/share-name, user/group name, object-type, access-string)

Adds or updates access (permission) records for a resource.

Syntax:

w9xAccessAdd(server-name, resource/share-name, user/group-name, object-type, access-string)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(s) user/group-name	name of a user or of a group to whom access is being granted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object. See below.
(i) access-string	the type of access that is being granted. Either a predefined access type, or a delimited list. See below.

Returns:

(s)	1.

W9xAccessAdd can be used for looking up user/group names and share names (if 'resource/share-name' specifies a share).

Object-Type

'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

Req#	object type
100	share (e.g., a directory accessed through a share)
200	printer object the syntax for the resource name must be "\\server-name\printer-object-name". If the printer is on a remote NT/2K system then the server name must be passed as the first parameter [e.g. "\\server-name"] in addition to the resource name [that also contains the server name].
300	file or directory in an NTFS partition
400	registry key

If **'object-type'** = 100 (share), then 'resource/share-name' should specify the name of the share.

If **'object-type'** = 300 (file or directory in an NTFS partition), then 'resource/share-name' should be UNC (ie, "\\Server\Share\Dir").

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

Access-String

'Access-string' specifies the type of access that is being granted. It can be either (A) a predefined access type, or (B) a delimited list of one or more specific 'access-records'. Both are described below:

(A) Predefined access types:

These get translated into specific access records, as shown. It is possible that the appropriate values may vary depending on your system configuration, or among different versions of Windows NT:

Access-string	Meaning	Specific equivalent
"DirNT:List"	List	"0:2:1179817"
"DirNT:Read"	Read	"0:9:-1610612736|0:2:1179817"
"DirNT:Add"	Add	"0:2:1180086"
"DirNT:AddRead"	Add & Read	"0:9:-1610612736|0:2:1180095"
"DirNT:Change"	Change	"0:9:-536805376|0:2:1245631"
"DirNT:Full"	Full Control	"0:9:268435456|0:2:2032127"
"DirNT:None"	No Access	"1:9:268435456|1:2:2032127"
"Dir2K:Full"	Full Control	"0:3:2032127"
"Dir2K:Modify"	Modify	"0:3:1245631"
"Dir2K:ReadExecute"	ReadExecute	"0:3:1179817"
"Dir2K:List"	List	"0:2:1179817"
"Dir2K:Read"	Read	"0:3:1179785"
"Dir2K:Write"	Write	"0:3:1048854"
"Dir2K:None"	No acesss	"1:3:983551"
"DirShare:Read"	Read	"0:0:1179817"
"DirShare:Change"	Change	"0:0:1245631"
"DirShare:Full"	Full Control	"0:0:2032127"
"DirShare:None"	No Access	"1:0:2032127"

Note: "DirShare" is a directory accessed through a share ("object-type" = 100). "DirNT' is a directory in an NTFS partition, accessed through NT security ("object-type" = 300).

"File:Read"	Read	"0:0:1179817"
"File:Change"	Change	"0:0:1245631"
"File:Full"	Full Control	"0:0:2032127"
"File:None"	No Access	"1:0:2032127"

"PrintNT:Print"	Print	"0:2:131080"
"PrintNT:Manage"	Manage Documents	"0:9:268435456|0:2:131072"
"PrintNT:Full"	Full Control	"0:9:268435456|0:2:983052"
"PrintNT:None"	No Access	"1:9:268435456|1:2:983052"
"Print2K:Print	Print	"0:0:131080"
"Print2K:Manage"	Manage Documents	"0:10:131072|0:9:983056"
"Print2K:Full"	Full Control	"0:9:983056|0:0:983052"
"Print2K:None"	No Access	"1:0:983052|1:9:983056"
"Reg:Read"	Read	"0:2:131097"
"Reg:Full"	Full Control	"0:2:983103"

(B) Specific 'access-records':

This can be a single record, or a list of records (maximum of 10) delimited with vertical bars (**|**). Each record is in the format:

```
record-type:access-flags:access-rights
```

where 'record-type', 'access-flags', and 'access-rights' are each a decimal number, separated with colons (**:**).

It is not expected that most users will want to manually create or edit 'access-records' strings. Instead, you can use the **w9xAccessGet** function to return an 'access-records' string in the proper format for use with this function. This is useful for transferring access rights from one user to another.

A brief description of the fields in the 'access-records' string follows. Please note that any detailed explanation is beyond the scope of this document, but might be obtained from the WIN32 SDK programmers' documentation available from Microsoft and other publishers.

'record-type' (one of the following):

0 ACCESS_ALLOWED_ACE_TYPE
1 ACCESS_DENIED_ACE_TYPE

'access-flags' (0, or, one or more of the following):

1 OBJECT_INHERIT_ACE
2 CONTAINER_INHERIT_ACE
4 NO_PROPAGATE_INHERIT_ACE
8 INHERIT_ONLY_ACE

'access-rights' (one or more of the following, usually several, depending on the object type.)

Note: This is not a complete list):

1 FILE_LIST_DIRECTORY (Dir)

1	FILE_READ_DATA (File)
2	FILE_ADD_FILE (Dir)
2	FILE_WRITE_DATA (File)
4	FILE_ADD_SUBDIRECTORY (Dir)
4	FILE_APPEND_DATA (File)
8	FILE_READ_EA
16	FILE_WRITE_EA
32	FILE_TRAVERSE (Dir)
32	FILE_EXECUTE (File)
64	FILE_DELETE_CHILD
128	FILE_READ_ATTRIBUTES
256	FILE_WRITE_ATTRIBUTES
65536	DELETE
131072	READ_CONTROL
262144	WRITE_DAC
524288	WRITE_OWNER
1048576	SYNCHRONIZE
16777216	ACCESS_SYSTEM_SECURITY
268435456	GENERIC_ALL
536870912	GENERIC_EXECUTE
1073741824	GENERIC_WRITE
-2147483648	GENERIC_READ

If any access records already exist for 'resource/share-name' for the specified 'user/group name', they will be removed before adding the records specified by 'access-string'.

Note: If you have not previously added any permissions to a share, it may implicitly have some default permissions. For example, when you create a share for a directory, it defaults to giving "Full Control" access to "Everyone". When **w9xAccessAdd** is used to create an access record for such a share, it will supersede those default permissions (i.e., the default permissions will be removed). If you wish to keep the default permissions, use **w9xAccessAdd** to set them explicitly.

Example:

```
;This example sets the share called "Public" on the current machine so
;that any member of the group "Everybody" has full access to the contents
;of the directory associated with the "Public" share.  This function does
;not affect any permissions that may have been set with a NTFS file system
;with respect to the directory associated with the share.
;
AddExtender("WWW9X34i.DLL")
w9xAccessAdd("\\MYSERVER","Public","Everybody",100,"DirShare:Full")
```

See Also:

w9xAccessDel

w9xAccessDel(server-name, resource/share-name, user-name, object-type)

Removes an access (permission) records from a resource.

Syntax:

w9xAccessDel(server-name, resource/share-name, user-name, object-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(s) user-name	name of a user or of a group to whom access is being granted. If necessary, it can be fully qualified as 'server\user'.
object-type	identifies the object type of the 'resource/share-name'. 100 share (e.g., a directory accessed through a share) 200 printer (accessed through a share) 300 file or directory in an NTFS partition 400 registry key

Returns:

(i)	**@TRUE** if records were deleted, **@FALSE** if no records were found.

Example:

```
; This example shows the removal of all specific references to the
; group "Supervisors" in relation to a file "D:\NTFS\blackbook.txt"
; on a NTFS file system for the current system.  Note that this could
; either remove specific permission to access the file, or it could
; remove specific access denial to the file.

AddExtender("WWW9X34i.DLL")
rslt=w9xAccessDel("\\MYSERVER","D:\NTFS\blackbook.txt","Supervisors",300)
if rslt
   Message("w9xAccessDel","Access records deleted")
else
   Message("w9xAccessDel","No access records found")
endif
```

See Also:

w9xAccessAdd

w9xAccessGet(server-name, resource/share-name, user-name, object-type)

Returns access (permission) records for a resource.

Syntax:

w9xAccessGet(server-name, resource/share-name, user-name, object-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(s) user-name	name of a user or of a group to whom access is being granted. If necessary, it can be fully qualified as 'server\user'.
(i) object-type	identifies the type of the 'resource/share-name' object.

Returns:

(i)	a delimited list of access records or ("") if no records were found.

W9xAccessGet returns a list of access records for 'resource/share-name' for the specified 'user/group name', delimited with vertical bars (|). If there are no appropriate records, it returns a blank string ("").

See **w9xAccessAdd** for information on the format of the access records.

Note: If no permissions were previously added to a share, it may implicitly have some default permissions. For example, when a share is created for a directory, it defaults to giving "Full Control" access to "Everyone", although there may not actually be any access records for the share. Therefore, **w9xAccessGet** may return a blank string (""). However, implicit permissions will become actual permissions when from the File Manager (or Explorer) the "Permissions" dialog for the share is brought up and "OK" is selected. **w9xAccessGet** can then retrieve the actual permissions.

Example:

```
records=w9xAccessGet("\\MYSERVER","Public","jdoe",100)
if records==""
   Message("w9xAccessGet","No records found for share 'Public'")
else
Message("w9xAccessGet",strcat("'Public' Share Records are",@crlf, records))
endif
```

See Also:

w9xAccessDel, w9xAccessAdd

w9xAccessList(server-name, resource/share-name, object-type, flags)

Returns list of users who have access (permission) records for a resource.

Syntax:
w9xAccessList(server-name, resource/share-name, object-type, access string)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(i) object-type	indicates the type of object identified by 'resource/share-name'. See below.
(i) flags	specifies the format of the returned names.

Returns:

(s)	a tab-delimited list of users and groups who have access records for "resource/share-name"; ie, users and groups for whom permissions have explicitly been set. Returns a blank string ("") if there are no appropriate records.

Object-Type
'Object-type' indicates the type of object identified by 'resource/share-name', and can be one of the following:

Req#	object type
100	share (e.g., a directory accessed through a share)
200	printer (accessed through a share)
300	file or directory in an NTFS partition
400	registry key

If **'object-type'** = 100 (share), then 'resource/share-name' should specify the name of the share.

If **'object-type'** = 400 (registry key), then 'resource/share-name' should be the handle of an open registry key (opened with the **RegOpenKey** function), or a predefined registry handle. (Registration Functions are listed in the WIL Reference Manual under "Registration Database Operations".)

Otherwise, 'resource/share-name' should specify the name of the resource (e.g., "HP LaserJet III" or "C:\UTIL\MYFILE.TXT").

Flags
'Flag' specifies the format of the returned names, and can be one of the following:

0 account (eg, "johndoe")
1 domain\account (eg, "OFFICE\johndoe")

Note: For built-in accounts which are predefined by the system (eg, "Administrators"), only the account name is returned, regardless of the "flag" setting.

Example:

```
;This example sets the share called "Public" on the current machine so
;that any member of the group "Everybody" has full access to the contents
;of the directory associated with the "Public" share.  This function does
;not affect any permissions that may have been set with a NTFS file system
;with respect to the directory associated with the share.
;
AddExtender("WWW9X34i.DLL")
w9xAccessAdd("\\MYSERVER","Public","Everybody",100,"DirShare:Full")
users = w9xAccessList("\\MYSERVER", "Public", 100, 0)
```

See Also:

w9xAccessDel, w9xAccessAdd

w9xFileUsers(server-name, file-pathname)

Lists network users who have a file open.

Syntax:
w9xFileUsers(server-name, file-pathname)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER").
(s) file-pathname	a fully-qualified file name (eg, "C:\DOC\MYFILE.TXT") **Note**: The file name MUST be fully-qualified.

Returns:

(s)	a tab-delimited list of user names.

Example:

```
AddExtender("WWWNT34i.DLL")
users=w9xFileUsers( "\\Server\L4","C:\DOC\MYFILE.TXT")
AskItemList("List of network users who have a file open", users, @TAB, @SORTED,
@SINGLE)
exit
```

See Also:

w9xGetDc(server-name, domain-name, flag)

Returns the domain controller for a domain.

Syntax:
w9xGetDc(server-name, domain-name, flag)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER").
(s) domain-name	is the name of a domain, or a blank string ("") to indicate the primary domain.
(i) flag	(see below)

Returns:

(s)	Returns a domain controller name in UNC format (eg, "\\MYSERVER").

"flag" must be set to 1:

Flag	Meaning
1	Return the primary domain controller for the specified domain

Note: You ***might*** be able to specify the local machine name as the server-name parameter in **w9xGetDc**. This is contrary to Microsoft's claims, that this cannot be done.

```
AddExtender("WWW9X34i.DLL")
dc=w9xGetDc( "\\LocalMachine", "WorkGrp", 1)
message("Primary Domain controller for Workgrps",dc)
```

Example:

```
AddExtender("WWW9X34i.DLL")
dc=w9xGetDc( "\\MYSERVER", "WorkGrp", 1)
message("Primary Domain controller for Workgrps",dc)
```

See Also:

w9xGroupAdd(server-name, group-name, group-type, comment)

Creates a user group.

Syntax:
w9xGroupAdd(server-name, group-name, group-type, comment)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (eg, "\\MYSERVER").
(s) group-name	the name of the group to be created.
(i) group-type	**@GLOBALGROUP**.
(s) comment	is an optional description of the group, or "" for none.

Returns:

(s)	always 1.

Example:

```
AddExtender("WWW9X34i.DLL")
w9xGroupAdd("\\Server\L4", "Everybody", @GLOBALGROUP, "")
Message("Everybody","Group added")
exit
```

See Also:

w9xListGroups, w9xGroupInfo

w9xGroupDel(server-name, group-name, group-type)

Deletes a user group.

Syntax:
w9xGroupDel(server-name, group-name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (eg, "\\MYSERVER").
(s) group-name	is the name of a group to be deleted.
(i) group-type	**@GLOBALGROUP**.

Returns:

(s)	always 1.

Example:

```
AddExtender("WWW9X34i.DLL")
w9xGroupDel("\\Server\L4", "DaGroup", @GLOBALGROUP)
Message("DaGroup","Group Deleted")
exit
```

See Also:

w9xGroupAdd

w9xGroupInfo(server-name, group, group-type, request)

Returns information about a group.

Syntax:
w9xGroupInfo(server-name, group, group-type, request)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) group	name of a group.
(s) group-type	**@GLOBALGROUP**.
(s) request	specifies the information to be returned, (see below).

Returns:

(s)	Returns a string.

"request" specifies the information to be returned, and can be one of the following:

request	type	description
0	(s)	group name
1	(s)	group comment
2	(i)	group's RID (relative identifier) This request is valid only with global groups.

Example:

```
AddExtender("WWW9X34i.DLL")
comment = w9xGroupInfo("\\Server\L4", "Administrators", @GLOBALGROUP, 1)
Message("Comment for 'Administrators'", comment)
```

See Also:
w9xListGroups

w9xListGroups(server-name, group-type)

Returns tab-delimited list of all user groups on a specified server.

Syntax:
w9xListGroups(server-name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(i) group-type	**@GLOBALGROUP**.

Returns:

(i)	a tab-delimited list of user groups.

Example:

```
AddExtender("WWW9X34i.DLL")
globalgroups=w9xListGroups("\\Server\L4",@GLOBALGROUP)
AskItemList("Global Groups",globalgroups,@tab,@sorted,@single)
```

See Also:
w9xMemberList

w9xMemberDel(server-name, group-name, user-name, group-type)

Deletes the specified user from the specified group on the specified server.

Syntax:

w9xMemberDel(server-name, group-name, user-name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) group-name	name of the group.
(s) user-name	name of the current user.
(i) group-type	**@GLOBALGROUP**.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if the user is not a member of the group.

Assuming that the person running this script has sufficient authority to delete users from the specified group, this function will delete the specified user from the group.

Example:

```
AddExtender("WWW9X34i.DLL")
rslt=w9xMemberDel("\\Server\L4","testgroup","jdoe",@GLOBALGROUP)
if rslt
     Message("jdoe","removed from testgroup")
else
     Message("jdoe","is not a member of testgroup")
endif
```

See Also:

w9xMemberGet, w9xMemberSet

w9xMemberGet(server-name, group-name, user-name, group-type)

Determines if the specified user is a member of the specified group on the specified server.

Syntax:
w9xMemberGet(server-name, group-name, user-name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) group-name	name of the group.
(s) user-name	name of the current user.
(i) group-type	**@GLOBALGROUP**

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

Assuming that the person running this script has sufficient authority to query members of the specified group, this function will allow the person to determine if the user is a member of the specified group or not.

Example:

```
AddExtender("WWW9X34i.DLL")
rslt=w9xMemberGet("\\Server\L4","testgroup","jdoe",@GLOBALGROUP)
if rslt
     Message("jdoe","is a member of testgroup")
else
     Message("jdoe","is not a member of testgroup")
endif
```

See Also:
w9xMemberSet, wntMemberDel

w9xMemberGrps(server-name, user-name, group-type, flags)

Returns a list of groups to which the specified user belongs.

Syntax:
w9xMemberGrps(server-name, user-name, group-type, flags)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) user-name	name of the current user.
(i) group-type	**@GLOBALGROUP**
(i) flags	0.

Returns:

(i)	a tab delimited list of groups.

Example:

```
AddExtender("WWW9X34i.DLL")
groups=w9xMemberGrps("\\Server\L4","jdoe",@GLOBALGROUP,0)
AskItemList("jdoe is associated with",groups,@tab,@sorted,@single)
```

See Also:
w9xMemberSet, wntMemberDel

w9xMemberList(server-name, group-name, group-type)

Returns a list of the members of a user group.

Syntax:
w9xMemberList(server-name, group-name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) group-name	name of the group.
(i) group-type	**@GLOBALGROUP**

Returns:

(i)	a tab delimited list of users.

Example:

```
AddExtender("WWW9X34i.DLL")
people=w9xMemberList("\\Server\L4","Everybody",@GLOBALGROUP)
AskItemList("Member of group Everybody",people,@tab,@sorted,@single)
```

See Also:
w9xMemberSet, wntMemberDel

w9xMemberSet(server-name, group-name, user-name or group-name, group-type)

Sets the specified user /or group as a member of the specified group on the specified server.

Syntax:
w9xMemberSet(server-name, group-name, user-name or group name, group-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) group-name	name of the group.
(s) user-name/group name	name of the current user or group.
(i) group-type	**@GLOBALGROUP.**

Returns:

(i)	**@TRUE** if successful; **@FALSE** if the user is already a member of the group.

The person running this script must have sufficient authority to add users/or groups to the specified group.

Example:

```
AddExtender("WWW9X34i.DLL")
rslt=w9xMemberSet("\\Server\L4","testgroup","jdoe",@GLOBALGROUP)
if rslt
     Message("jdoe","is a NEW member of testgroup")
else
     Message("jdoe","is ALREADY a member of testgroup")
endif


;To add a domain globalgroup to a local machine's group
AddExtender("WWW9X34i.DLL")
rslt=w9XMemberSet("\\server", "localgroup", "domain\globalgroup", @LOCALGROUP)
if rslt
     Message("domain\globalgroup","is a NEW member of localgroup")
else
     Message("domain\globalgroup","is ALREADY a member of localgroup")
endif
```

See Also:
w9xMemberDel, w9xMemberGet

w9xOwnerGet(server-name, reg-key, resource-name, object-type, flag)

Returns the owner of an object.

Syntax:

w9xOwnerGet(server-name, reg-key, resource-name, object-type, flag)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(i) reg-key	handle of an open registry key or 0. See below.
(s) resource-name	identifies the object to be accessed.
(i) object-type	indicates the type of object identified by 'Resource-name'.
(i) flag	specifies the format of the returned string.

Returns:

(s)	the name of the account that owns the object, or a blank string ("") if the object has no owner.

'Server-name' can be the name of the server on which the function will execute. This is used for looking up user/group names.

'Reg-key' is used only if the object is a registry key. If 'Object-type' = 400 (registry key), then 'Reg-key' should be the handle of an open registry key (opened with the "RegOpenKey" function), or a predefined registry handle (listed in the WIL Reference under "Registration Database Operations"). Otherwise, 'Reg-key' is ignored, and should be specified as 0.

'Resource-name' identifies the object to be accessed. if 'Object-type' = 400 (registry key), then 'Resource-name' can be a subkey string relative to 'Reg-key', or a blank string ("") if 'Reg-key' represents the actual object to be accessed. Otherwise, 'Resource-name' should specify the name of the object (eg, "C:\UTIL\MYFILE.TXT").

'Object-type' indicates the type of object identified by 'Resource-name', and can be one of the following:

Req #	Meaning
300	file or directory in an NTFS partition
400	registry key

'Flag' specifies the format of the returned string, and can be one of the following:

Req #	Meaning
0	account (eg, "johndoe")
1	domain\account (eg, "OFFICE\johndoe")

Note: For built-in accounts which are predefined by the system (eg, "Administrators"), only the

account name is returned, regardless of the 'Flag' setting.

Example:

```
;For a file:
AddExtender("WWWNT34i.DLL")
ErrorMode(@OFF)
owner = w9xOwnerGet("\\Server\L4", 0, "f:\test\myfile.txt", 300, 0)
ErrorMode(@ON)
Message("Owner is", owner)


;For a registry key:
AddExtender("WWWNT34i.DLL")
ErrorMode(@OFF)
owner = w9xOwnerGet("\\Server\L4", @REGMACHINE, "Software\Test", 400, 0)
ErrorMode(@ON)
Message("Owner is", owner)
```

See Also:

w9xRemoteTime(server-name, format)

Gets the time of day from a server.

Syntax:
w9xRemoteTime(server-name, format)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(i) format	specifies the format in which the time will be returned. See below.

Returns:

(s) server time	format of return value depends on which format was specified.

"format" can be one of the following values:

Value	Format
1	UTC/GMT time, in YYYY:MM:DD:hh:mm:ss format
2	local time (on local machine), in YYYY:MM:DD:hh:mm:ss format

Example:

```
AddExtender("WWW9x34i.DLL")
srvtime = w9xRemoteTime( "\\server", 1)
message("Current server time", srvtime )
```

See Also:

w9xServiceAt(server, domain, server-type, service-name, flags)

Lists all servers in a domain which contain a specified service.

Syntax:

w9xServiceAt(server, domain, server-type, service-name, flags)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) domain	the name of the domain which will be used (e.g., "SALES"), or ("") for the primary domain.
(i) server-type	identifies the type of servers which will be examined. See below.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies information on the service being looked for. See below.

Returns:

(i)	a tab-delimited list of server UNC names (e.g., "\\MYSERVER").

Note: An NT workstation can be considered to be a "server".

Server-type

Specify **-1** for all servers. Or, specify one or more of the following flags, combined using the binary OR ("|") operator.

Req#	Server Type
1	All LAN Manager workstation
2	All LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT (either workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT Non-DC server
65536	Server that can run the browser service
131072	Server running a browser service as backup

262144	Server running the master browser service
524288	Server running the domain master browser
4194304	Windows 95 or newer
-2147483648	Domain announcement

Service-Name

'Service name' is the name of a service (e.g., "Spooler") or driver (e.g., "Atdisk"). The name can be specified either as the "display name" which is listed in Control Panel (name-type = 0) or the "service name" which is the actual registry key for the service (name-type = 1000). The SDK documentation describes them as:

DisplayName = a string that is to be used by user interface programs to identify the service.
ServiceName = a string that names a service in a service control manager database.

So, the following two commands will yield identical results:

```
   servers = w9xServiceAt("\\Server\L4", "", -1, "Browser", 101)  ; display name
   servers = w9xServiceAt("\\Server\L4","", -1,"Computer Browser", 1001) ;service
name
```

Flags

'Flags' specifies information on the service being looked for. It consists of one entry from each of the following three groups, added together:

Req#	service type
1	services
2	drivers
3	both

Req#	service state
100	active services
200	inactive services
300	both

Name type indicates what the **'service-name'** parameter represents.

Req#	name type
0	display name (the name shown in Control Panel)
1000	service name (the actual registry key name)

Note: This function can take a while to run, depending on how many servers are in the domain. Also, it will only return the names of servers which it is able to access, which requires that the user have browse access to their service control managers.

Example:

```
;return a list of all servers running the "Spooler" service
```

```
servers = w9xServiceAt("\\Server\L4", "", -1, "Spooler", 101)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)

;return a list of all NT machines with an "Atdisk" driver installed
servers = w9xServiceAt("\\Server\L4", "", 4096, "Atdisk", 302)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)
```

w9xServerList(server, domain, server-type)

Lists all servers in a domain.

Syntax:

w9xServerList(server, domain, server-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) domain	the name of the domain which will be used (e.g., "SALES"), or ("") for the primary domain.
(i) server-type	identifies the type of servers which will be examined. See below.

Returns:

(i)	Returns a tab-delimited list of server UNC names (eg, "\\MYSERVER").

Note: An NT workstation can be considered to be a "server".

Server-type

Specify **-1** for all servers. Or, specify one or more of the following flags, combined using the binary OR ("|") operator.

Req#	Server Type
1	All workstations
2	All servers
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT/Windows 2000 (workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT/Windows 2000 Non-domain controller
65536	Server that can run the browser service
131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser
1048576	OSF Server
2097152	VMS Server
4194304	Windows 95 or later
8388608	Root of DFS tree
16777216	Server clusters available in the domain
33554432	Terminal Server
268435456	IBM DSS (Directory & Security Services) or equivalent

536870912	return list for alternate transport
1073741824	Servers maintained by the browser
-2147483648	Primary Domain

Note: This function can take a while to run, depending on how many servers are in the domain. Also, it will only return the names of servers which it is able to access, which requires that the user have browse access to their service control managers.

Example:

```
AddExtender("www9x34i.dll")
;return a list of all servers
servers = w9xServerlist("\\Server\L4", "", -1)  ;Specify -1 for all servers
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)

;return a list of all NT machines
servers = w9xServerlist("\\Server\L4", "", 4096)
AskItemList("NT machines",servers,@tab,@sorted,@single)
```

w9xShareAdd(server-name, resource, share-name,share-type, max-users)

Shares a resource.

Syntax:

w9xShareAdd(server-name, resource, share-name,share-type, max-users)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource	identifies the object to be shared. Resource can be a directory name (e.g., "c:\util"), a printer object (e.g., "HP LaserJet III"), or the name of a sharable device.
(s) share-name	name by which other users will access the resource.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(i) max-users	the maximum number of users allowed, or -1 for the highest possible number.

Returns:

(s)	1.

If the 'share-type' is 1 for printer both the 'server-name' and 'max-users' are ignored. Set 'server-name' to a blank string ("") and 'max-users' to -1.

Example:

```
; This example adds a share called "Public" to the C:\TEMP
; directory.  It sets max-users to -1 to provide unlimited
; concurrent access to the directory.
AddExtender("WWW9X34i.DLL")
w9xShareAdd("\\Server\L4","C:\TEMP","Public",0,-1)
```

See Also:

w9xShareDel, w9xShareSet

w9xShareDel(server-name, resource/share-name, share-type)

UN-shares a resource.

Syntax:
w9xShareDel(server-name, resource/share-name, share-type)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.

Returns:

(s)	**@TRUE** if the share was deleted; **@FALSE** if there was no share with the specified name.

If 'share-type' = 1 (printer), then 'resource/share-name' should specify the object's resource name (e.g., "HP LaserJet III"). Otherwise, it should specify the object's share name.

Example:

```
; This example removes the "Public" share from the system.
AddExtender("WWW9X34i.DLL")
rslt=w9xShareDel("\\Server\L4","Public",0)
if rslt
     Message("w9xShareDel","Share PUBLIC deleted")
else
     Message("w9xShareDel","Share PUBLIC not found")
endif
```

See Also:
w9xShareAdd, w9xShareSet

w9xShareInfo(server-name, resource/share-name, share-type, request)

Returns information about a shared resource.

Syntax:
w9xShareInfo(server-name, resource/share-name, share-type, request)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(i) request	specifies the information to be returned. See below.

Returns:

(s)/(i)	a string or integer, depending on "request".

"request" specifies the information to be returned, and can be one of the following:

0 (s) share name
1 (s) resource
2 (s) comment
3 (s) location
6 (i) share type
8 (i) max users
9 (i) current users

Note: This function can only be performed by members of the Administrators or Account Operators local group, or those with Communication, Print, or Server operator group membership.

Example:

```
AddExtender("WWW9X34i.DLL")
location= w9xShareInfo("\\Server", "\\Server\Laser5", 1, 3)
Message("location of printer", location)
```

See Also:
w9xShareAdd, w9xShareSet

w9xShareSet(server-name, resource/share-name, share-type, comment, description)

Sets additional share information for a resource.

Syntax:

w9xShareSet(server-name, resource/share-name, share-type, comment, location)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) resource/share-name	identifies the object to be modified.
(i) share-type	identifies the type of the 'resource' object. 0 = directory, 1 = printer, and 2 = device.
(s) comment	text string used to describe the share or a blank string ("") if no comment is desired.
(s) location	a text string that can be used to describe the location of the printer, 'share-type' 1, or a blank string ("") if no location description is desired. A blank string ("") must be specified for any other 'share-type'.

Returns:

(s)	1.

Example:

```
; This example adds a comment to the Public share.  Other network users can
; see the comment when they browser the network.
AddExtender("WWW9X34i.DLL")
w9xShareSet("\\Server\L4","Public",0,"Public Access Directory on my machine","")
```

See Also:

w9xShareAdd, w9xShareDel

w9xUserAdd(server-name)

Adds a user account.

Syntax:

w9xUserAdd(server-name)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER").

Returns:

(i)	always 1.

Before calling this function, you must use **w9xUserAddDat** to set parameters for the new user account. At a minimum, you must set the "name" element. The other elements will receive default values.

Calling this function does not reset elements in the user parameter structure. So, you can set various elements, add a user, then change just the "name" element and add another user. All other elements will retain their previous values. To clear all elements, call **w9xUserAddDat** specifying blank strings for "variable" and "value".

Example:

```
AddExtender("WWW9X34i.DLL")
w9xUserAddDat("name", "jdoe")
w9xUserAddDat("full_name", "John Doe")
w9xUserAddDat("flags", 1)
w9xUserAdd("\\Server\L4")
exit
```

See Also:

w9xUserAddDat

w9xUserAddDat(element, value)

Sets parameter information for **w9xUserAdd**. (This function sets values for elements in a user parameter structure, which is used by the w9xUserAdd function.)

Syntax:

w9xUserAddDat(element, value)

Parameters:

(s) element	see below.
(s / i) value	see below.

Returns:

(i)	**@TRUE** on success, **@FALSE** if there was a problem.

Note: "value" must contain a 4-digit year, and must appear in the precise format "YYYY:MM:DD:hh:mm:ss" (ie, exactly 19 characters long, with colons in exactly the right positions).

Element

Can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):

Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"password" (s):

The password for the user specified in the "name" element. The length cannot exceed 256 bytes. By convention, Windows NT limits the length of passwords to 14 characters. This convention allows LAN Manager, Windows 3.x, Windows for Workgroups 3.x, and Windows 95 clients to access a Windows NT server using the account.

"home_dir" (s):

Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):

Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):

Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
1	Normal Account	This flag is **REQUIRED** for new accounts.
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored in Windows NT.
16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.
64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.

65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE _ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST _ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST _ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST _ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

"script_path" (s):
Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):
Points to a 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.

"usr_comment" (s):
Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means

there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

If "variable" and "value" are both set to blank strings (""), all values will be cleared from the user parameter structure.

You can specify a value of "*NULL*" to set a string element to a NULL pointer, which is not the same as a NULL string (").

Example:

```
AddExtender("WWW9X34i.DLL")
w9xUserAddDat("name", "jdoe")
w9xUserAddDat("full_name", "John Doe")
w9xUserAddDat("flags", 1)
w9xUserAdd("\\Server\L4")
exit
```

See Also:

w9xUserAdd

w9xUserDel(server-name, user-name)

Deletes a user account.

Syntax:
w9xUserDel(server-name, user-name)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER").
(s) user-name	is the name of the user to be deleted.

Returns:

(i)	always 1.

Example:

```
AddExtender("WWW9X34i.DLL")
w9xUserDel("\\Server1", "jdoe")
```

See Also:

w9xUserAdd, w9xUserAddDat

w9xUserExist(server-name, user-name)

Determines whether a user exists.

Syntax:

w9xUserExist(server-name, user-name)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER")
(s) user-name	is the name of a user who may have an account on "server-name".

Returns:

(i)	**@TRUE** if the specified user exists, **@FALSE** otherwise.

Example:

```
AddExtender("WWW9X34i.DLL")
f=w9xUserGet("\\Server1", "jdoe")
if f == @True
   Message("User Exist?","Yes, user exist")
else
   Message("User Exist?","No, user does NOT exist")
endif
exit
```

See Also:

w9xUserAdd

w9xUserGetDat(server-name, user-name, element)

Returns parameter information for a user account.

Syntax:
w9xUserGetDat(server-name, user-name, element)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER").
(s) user-name	name of a user who has an account on "server-name".
(s) element	specifies the element to be returned. See below.

Returns:

(i)	Returns a string or integer value, depending on "element"..

Element

Can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):
Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"home_dir" (s):
Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):
Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):
Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is ignored in Windows NT.
16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.
64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.
65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE _ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST _ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST _ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST _ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

"script_path" (s):
Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):
Points to a 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.

"usr_comment" (s):
Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

"user_id" (i):
User's RID (relative identifier). Note: This element cannot be set using **w9xUserAddDat** or **w9xUserSetDat**.

"primary_group_id" (i):
RID (relative ID) of the user's primary global group. You can determine a group's RID using **w9xGroupInfo** with request = 2. **Note:** This element cannot be set using **w9xUserAddDat**.

Example:

```
AddExtender("WWW9X34i.DLL")
user="joe"
theflags=w9xUserGetDat("\\SERVER",user,"flags")
if theflags & 2
   Message(user,"Account Disabled")
endif
if theflags & 32
   Message(user,"Password not required")
endif
```

See Also:

w9xUserAddDat

w9xUserInfo(request)

Returns information about the currently logged-on user.

Syntax:

w9xUserInfo(request)

Parameters:

(i) request	specifies the information to be returned. See below.

Returns:

(s)	returns a string.

"request" specifies the information to be returned, and can be one of the following:

Value	Name	Meaning
0	user name	name of the user currently logged on to the workstation
1	logon domain	domain name of the user account of the user currently logged on to the workstation

Example:

```
AddExtender("WWW9X34i.DLL")
domain = w9xUserInfo(1)
Message("Logon domain", domain)
exit
```

w9xUserList(server-name, account-type)

Lists users with accounts on an NT server.

Syntax:
w9xUserList(server-name, account-type)

Parameters:

(s) server-name	is the UNC name of the server on which the function will execute (eg, "\\MYSERVER").
(i) account-type	specifies the types of user accounts to include. (see below)

Returns:

(s)	Returns a tab-delimited list of user names.

Note: to list users in a domain, specify a domain controller for that domain as "server-name".

"**account-type**" specifies the types of user accounts to include. You can specify 0 to include all account types, or specify one or more of the following flags, combined using the bitwise OR ('|') operator:

Flag	Meaning
1	Include local user account data on a domain controller.
2	Include global user account data on a computer.
8	Include domain trust account data on a domain controller.
16	Include workstation or member server account data on a domain controller.
32	Include domain controller account data on a domain controller.

Example:

```
AddExtender("WWW9X34i.DLL")
users = w9xUserList("\\Server\L4",0)
AskItemList("List of users with accounts on an NT server.",
users,@TAB,@SORTED,@SINGLE)
exit
```

See Also:

w9xUserInfo

w9xUserProps

Returns information about a network user.

Syntax:

w9xUserProps(server-name, user-name, request)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) user-name	is the name of a user who has an account on "servername".
(i) request	specifies the information to be returned, and can be one of the following:

Returns:

(s)	returns a string. See below.

Request codes 2 - 6 will likely return a blank string.

Value	Returns
0	Username
1	Full name
2	Description
3	User profile path
4	Login script name
5	Home directory
6	Home directory logon drive
7	Privilege level ("GUEST", "USER", or "ADMIN")

Example:

```
AddExtender("WWW9X34i.DLL")
server="\\Server"
user="John"

response=w9xUserProps(server, user, 0)
message("The user name is:",response)

response=w9xUserProps(server, user, 1)
message("The full name is:",response)

response=w9xUserProps(server, user, 2)
message("The description is:",response)

response=w9xUserProps(server, user, 3)
message("The user profile path is:",response)

response=w9xUserProps(server, user, 4)
message("The login script name is:",response)

response=w9xUserProps(server, user, 5)
```

```
message("The home directory is:",response)

response=w9xUserProps(server, user, 6)
message("The home directory logon drive is:",response)

response=w9xUserProps(server, user, 7)
message("The Privilege level is:",response)

message("Finished","Retrieving information about a network user")
exit
```

See Also:

w9xuserinfo

w9xUserRename

Renames a user account.

Syntax:

w9xUserRename(server-name, old-username, new-username)

Parameters:

(s) server-name	UNC name of the server on which the function will execute (eg, "\\MYSERVER")
(s) old-username	An existing account name.
(s) new-username	New name to be given to the account.

Returns:

(i)	**@TRUE** on success, **@FALSE** if there was a problem.

See Also:

w9xUserInfo

w9xUserSetDat(server-name, user-name, element, value)

Modifies parameter information for a user account.

Syntax:
w9xUserSetDat(server-name, user-name, element, value)

Parameters:

(s) server-name	the UNC name of the NT server on which the function will execute (e.g., "\\MYSERVER").
(s) user-name	is the name of a user who has an account on "servername".
(s) element	See below.
(s) value	See below.

Returns:

(s)	**@TRUE** on success, **@FALSE** if there was a problem.

Note: "value" must contain a 4-digit year, and must appear in the precise format "YYYY:MM:DD:hh:mm:ss" (ie, exactly 19 characters long, with colons in exactly the right positions).

"element" can be one of the following elements in the structure. Its type (string or integer) is shown in parentheses, followed by a description of its corresponding "value":

"name" (s):
Specifies the name of the user account. The number of characters in the name cannot exceed 256.

"password" (s):
The password for the user specified in the "name" element. The length cannot exceed 256 bytes. By convention, Windows NT limits the length of passwords to 14 characters. This convention allows LAN Manager, Windows 3.x, Windows for Workgroups 3.x, and Windows 95 clients to access a Windows NT server using the account.

"home_dir" (s):
Points to a string containing the path of the home directory of the user specified in "user_name". The string can be null.

"comment" (s):
Points to a string that contains a comment. The string can be a null string, or it can have any number of characters before the terminating null character.

"flags" (i):
Contains values that determine several features. This element can be any of the following values:

Value	Name	Meaning
2	UF_ACCOUNTDISABLE	The user's account is disabled.
8	UF_HOMEDIR_REQUIRED	The home directory is required. This value is

		ignored in Windows NT.
16	UF_LOCKOUT	The account is currently locked out.
32	UF_PASSWRD_NOTREQD	No password is required.
64	UF_PASSWRD_CANT_CHANGE	The user cannot change the password.
65536	UF_DONT_EXPIRE_PASSWD	Don't expire password.

The following values describe the account type. Only one value can be set.

Value	Name	Meaning
256	UF_TEMP_DUPLICATE_ACCOUNT	This is an account for users whose primary account is in another domain. This domain, but not to any domain that trusts this domain. The User Manager refers to this account type as a local user account.
512	UF_NORMAL_ACCOUNT	This is a default account type that represents a typical user.
2048	UF_INTERDOMAIN_TRUST_ACCOUNT	This is a permit to trust account for a Windows NT domain that trusts other domains.
4096	UF_WORKSTATION_TRUST_ACCOUNT	This is a computer account for a Windows NT Workstation or Windows NT Server that is a member of this domain.
8192	UF_SERVER_TRUST_ACCOUNT	This is a computer account for a Windows NT Backup Domain Controller that is a member of this domain.

"script_path" (s):
Points to a string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be null.

"full_name" (s):
Points to a 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.

"usr_comment" (s):
Points to a 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.

"workstations" (s):
Points to a 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 (2) value in the "flags" element.

"acct_expires" (i):
Specifies when the account will expire. To indicate that the account should have no expiration date, specify "0000:00:00:00:00:00".

"max_storage" (i):
Specifies the maximum amount of disk space the user can use. Use -1 to use all available disk space.

"logon_hours" (s):
Points 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. 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. A null pointer in this element means there is no time restriction.

Note: 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 PST).

"country_code" (i):
Specifies the country code for the user's language of choice.

"code_page" (i):
Specifies the code page for the user's language of choice.

"profile" (s):
Specifies a path to the user's profile. This value can be a null string, a local absolute path, or a UNC path.

"home_dir_drive" (s):
Specifies the drive letter assigned to the user's home directory for logon purposes.

"primary_group_id" (i):
RID (relative ID) of the user's primary global group. You can determine a group's RID using **w9xGroupInfo** with request = 2. **Note:** This element cannot be set using **w9xUserAddDat**.

"password_expired" (i):
Determines whether the password of the user has expired. Specify nonzero to indicate that the user must change password at next logon.

If "variable" and "value" are both set to blank strings (""), all values will be cleared from the user parameter structure.

You can specify a value of "*NULL*" to set a string element to a NULL pointer, which is not the same as a NULL string ("").

Note: This function cannot rename an account. It can only modify parameter information for a user account. To rename a account use the function w9xUserRename.

Example:
```
AddExtender("WWW9X34i.DLL")
w9XUserSetDat("\\Server\L4","joed","full_name", "John Doe")
exit
```

See Also:
w9xUserAddDat, w9xUserRename, w9xUserInfo

w9xVersion( )

Returns the version of this Extender DLL.

Syntax:
w9xVersion( )

Parameters:
none

Returns:
(i) the version of number of this extender Dll.

This function is used to check the version number of this Dll in cases where older DLL's exist and alternate processing is desirable. Version numbers of newer versions will be larger than that of older versions.

Example:

```
AddExtender("WWW9X34i.DLL")
a=w9xVersion()
Message("Dll Version",a)
```

w95AccessAdd(server-name, resource, user-name, access-rights, flags)

Adds or updates an access (permission) record for a resource.

Syntax:

w95AccessAdd(server-name, resource, user-name, access-rights, flags)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	identifies the object to be shared. Resource can be a directory name (e.g., "c:\util"), a printer object (e.g., "\PRINT\HP LaserJet III"), or the name of a sharable device. Printers must be specified in the form, "\PRINT\<share-name>", where "<share-name>" is substituted with the actual share name.
(s) user-name	name of a user or of a group to whom access is being granted.
(i) access-rights	the type of access that is being granted to "user-name". (determined by the operating system in use)
(i) flags	0 - reserved for future use.

Returns:

(s)	1.

Access-rights

Under Windows 95/98/Me, **'access-rights'** can be one of the following standard permission types:

Access request	Meaning
@ACC_READ_95	Read-only access
@ACC_FULL_95	Full access

If 'resource' specifies a printer, **'access-rights'** should be @ACC_FULL_95.

'Access rights' can also be a combination of the following custom permission flags, combined using the binary "OR" ("|") operator:

@ACC_READ	Read
@ACC_WRITE	Write
@ACC_CREATE	Create
@ACC_DELETE	Delete
@ACC_ATTRIB	Change attributes

@ACC_LIST	List files
@ACC_CONTROL	Change access control

If an access record already exists for 'resource' for the specified user, it will be updated to the specified 'access rights'.

Note: This function requires that user-level access control be used. (Check the configuration under "Network" settings.)

Example:

```
; This example shows how to add access records for jdoe to a particular
; directory.  Because of an idiosyncrasy of the 95 OS, the directory name
; is specified instead of a more logical share name.  C'est la vie.
; This example allows full access rights to the directory for jdoe
AddExtender("WWW9534i.DLL")
w95AccessAdd("","C:\TEMP","jdoe",@ACC_FULL_95,0)
```

See Also:

w95AccessDel

w95AccessDel(server-name, resource, user-name)

Removes an access (permission) record from a resource.

Syntax:
w95AccessDel(server-name, resource, user-name)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	identifies the object to be shared. Resource can be a directory name (e.g., "c:\util"), a printer object (e.g., "\PRINT\HP LaserJet III"), or the name of a sharable device. Printers must be specified in the form, "\PRINT\<share-name>", where "<share-name>" is substituted with the actual share name.
(s) user-name	name of a user or the name of a group.

Returns:

(i)	**@TRUE** if records were deleted, **@FALSE** if no records were found.

Note: This function requires that user-level access control be used. (Check the configuration under "Network" settings.)

Example:

```
; This example shows how to remove access records for jdoe to a particular
; directory.  Because of an idiosyncrasy of the 95 OS, the directory name
; is specified instead of a more logical share name.  C'est la vie.
; Note that removing access records could remove access records
; that either specifically permit or deny access to the resource.
AddExtender("WWW9534i.DLL")

rslt=w95AccessDel("","C:\TEMP","jdoe")
if rslt
   Message("w95AccessDel","jdoe records to D:\TEMP via net shares removed.")
else
   Message("w95AccessDel","No access records found")
endif
```

See Also:
w95AccessAdd

w95AddDrive(user-id, pswd, net-resource, local-drive, persist)

Maps a drive.

Syntax:
w95AddDrive(user-id, pswd, net-resource, local-drive, persist)

Parameters:

(s) user-id	**@DEFAULT** for current user
(s) pswd	password of network share (if required), **@DEFAULT** for current password or **@NONE** for nopassword
(s) net-resource	UNC netname of net resource
(s) local drive	local drive id e.g. ("K:") or **@NONE** for connect only
(s) persist	**@TRUE** Specifies persistent connection, one that will automatically reconnect when you reboot windows. **@FALSE** Specifies a temporary connection.

Returns:

(i)	always 1

This function allows a connection to be made to a net resource, and, optionally, a drive to be mapped to the net resource.

Windows 95 and 98 only allow the currently logged in user credentials, therefore you must specify @DEFAULT for user-id and password.

Example:

```
AddExtender("WWW9534i.DLL")
w95AddDrive(@default,@default,"\\SERVER\PUB","E:",@false)
RunWait("E:\EXCEL\EXCEL.EXE","\E")
w95CancelCon("E:",@TRUE,@TRUE)
```

See Also:
w95DirDialog, w95CancelCon

w95AddPrinter(user-id, pswd, net-resource, local device, persist)

Maps a printer resource to a local port.

Syntax:
w95AddPrinter(user-id, pswd, net-resource, local device, persist)

Parameters:

(s) user-id	user-id or **@DEFAULT** for current user
(s) pswd	password or **@DEFAULT** for current password or **@NONE** for no password
(s) net-resource	UNC netname of net resource
(s) local device	local printer port e.g. ("lpt1") or **@NONE** for connect only
(s) persist	**@TRUE** - Specifies persistent connection, one that will automatically reconnect when you reboot windows. **@FALSE** - Specifies a temporary connection.

Returns:

(i)	**@TRUE** if the port was mapped; **@FALSE** the port was not mapped.

This function allows a connection to be made to a net resource, and, optionally, a local device to be mapped to the net resource.

Example:

```
AddExtender("WWW9534i.DLL")
w95AddPrinter(@default,@default,"\\SERVER\LJ4","lpt2",@false)
```

See Also:
w95DirDialog, w95CancelCon

w95CancelCon(local drive, persist, forceflag)

Breaks a network connection.

Syntax:

w95CancelCon(local drive, persist, forceflag)

Parameters:

(s) local drive	the mapped device. drive letter or UNC path.
(s) persist	**@TRUE** - update persistent connection table **@FALSE** - do not update persistent connection table to remove this device
(i) forceflag	**@TRUE** - breaks the connection regardless of open files. **@FALSE** - if files are open, connection will not be broken.

Returns:

(i)	**@TRUE** if successful; **@FALSE** if unsuccessful.

You can use EITHER the drive letter OR UNC path name in the netcancelcon, wntcancelcon, w95cancelcon functions,in the 32-bit network extenders. Note that when you specify the UNC path name in these 3 functions, that any drive letters that are mapped to that UNC path will be broken when you cancel the connection.

When a mapped local drive is specified, only that connection will be closed.

If persist is set to **@TRUE**, then the persistent connection will be updated to remove this drive mapping from the list of persistent connections.

NOTE: It is important to specify **@TRUE** for the persist parameter in wntCancelCon, if you plan to map this drive letter again using the function **wntAddDrive**.

If forceflag is set to **@FALSE**, **w95CancelCon** will not break the connection if any files on that connection are still open. If forceflag is set to **@TRUE**, the connection will be broken regardless.

Example:

```
AddExtender("WWW9534i.DLL")
w95AddDrive(@default,@default,"\\SERVER\PUB","E:",@false)
RunWait("E:\EXCEL\EXCEL.EXE","\E")
w95CancelCon("E:",@TRUE,@TRUE)
```

See Also:

W95AddDrive, W95DirDialog

w95DirDialog(flag)

Brings up a network drive connect/disconnect dialog box

Syntax:
w95DirDialog(flag)

Parameters:

(i) flag	**@FALSE**=disconnect dialog **@TRUE**=connect dialog

Returns:

(i)	**1**

This function prompts the user with either a standard Connect or Disconnect dialog box. The user may make or break network drive mappings via the dialog box.

Example:

```
AddExtender("WWW9534i.DLL")
err=w95DirDialog(@TRUE)
runwait("excel.exe", "/e")
w95DirDialog(@FALSE)
```

See Also:
W95AddDrive

w95FileClose(server-name, file-pathname)

Close all network connections to a file.

Syntax:

w95FileClose(server-name, file-pathname)

Parameters:

(s) server-name	"server-name" is the name of a remote server on which the function will execute, or a blank string ("") to indicate the local computer.
(s) file-pathname	"file-pathname" is a fully-qualified file name (eg,"C:\DOC\MYFILE.TXT"). NOTE: The file name MUST be fully-qualified.

Returns:

(i)	**@TRUE** on success, or **@FALSE** if the specified file was not open.

Note: this function will not work to do a w95FileClose on a local file that was opened by yourself on the local machine, even if it's done over a mapped drive with a UNC.

Note: you will get the 597 error if you are trying to do a w95FileClose on a file to which you do not have administrator access rights .

Example:

```
;;the "C:\DOC\MYFILE.TXT" on the local machine needs to be opened by
;;someone else over the network.

AddExtender("WWW9534i.DLL")
Dirchange(origdir)

run("excel.exe", strcat(origdir,"TestBook1.xls")) ;or any Excel file you've created
WinWaitExist("~Excel",3)

flocate=strcat(origdir,"TestBook1.xls")

Display(2,"The File is Located in:", flocate)

b=fileexist(strcat(origdir,"TestBook1.xls"))
Display(2,"FileExist for TestBook1.xls", b);this should display a 2 if open

a=w95FileClose("", flocate)
Display(2,"w95FileClose =", a)
```

See Also:

W95AddDrive

w95FileUsers(server-name, file-pathname)

Lists network users who have a file open.

Syntax:

w95FileUsers(server-name, file-pathname)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (eg, "\\MYSERVER") or a blank string ("") to indicate the current machine.
(s) file-pathname	a fully-qualified file name (eg, "C:\DOC\MYFILE.TXT") **Note**: The file name MUST be fully-qualified.

Returns:

(s)	a tab-delimited list of user names.

Example:

```
AddExtender("WWW9534i.DLL")
users=w95FileUsers( "\\Server\L4","C:\DOC\MYFILE.TXT")
AskItemList("List of network users who have a file open", users, @tab, @sorted,
@single)
exit
```

See Also:

w95GetCon(local name)

Returns the name of a connected network resource.

Syntax:
w95GetCon(local name)

Parameters:

(s) local name	local drive name or printer resource (ie.,LPT1).

Returns:

(i)	name of a network resource.

w95GetCon returns the name of the network resource currently connected to a 'local name'. If the resource is not mapped a null string will be returned.

Example:

```
AddExtender("WWW9534i.DLL")
netrsrc=w95GetCon("K:")
if netrsrc==="" then Message("Drive K: is","not mapped")
else Message("Drive K: is mapped to",netrsrc)
```

See Also:
W95AddDrive, W95DirDialog

w95GetDrive(net-resource)

Lists local drives mapped to a UNC.

Syntax:
w95GetDrive(net-resource)

Parameters:

(s) net-resource	specifies a UNC, in the form "\\SERVER\SHARE". It is not case-sensitive, but must otherwise EXACTLY match the UNC name to which the drive(s) are mapped (eg, must not have a trailing backslash).

Returns:

(s)	Returns a tab-delimited list of drives (eg, "H: W:").

Example:

```
AddExtender("WWW9534i.DLL")
drvltr=w95GetDrive("\\Server\Share")
Askitemlist("Server is mapped to",drvltr,@tab,@unsorted,@single)
```

See Also:

w95GetCon

w95GetUser(netname)

Returns the name of the user currently logged into the network.

Syntax:
w95GetUser(netname)

Parameters:

(s) netname -	name of network or **@DEFAULT** for default network.

Returns:

(s)	the user name.

This function will interrogate the network and return the current user name. **@default** will return the user id of the local user.

"netname" can be the name of a local device, or the name of a remote network resource that the user has made a connection to, or the name of a network provider (i.e, "Microsoft Windows Network" or "NetWare Services")

Example:

```
AddExtender("WWW9534i.DLL")
username=w95GetUser(@default)
Message("Current User is",username)
```

See Also:
w95GetCon

w95Resources(net-resource, scope, type, usage)

Itemizes network resources.

Syntax:
w95Resources(net-resource, scope, type, usage)

Parameters:

(s) net-resource	a UNC (e.g., "\\FredsPC"), a domain (e.g., "SALES"), or ("") for the root of the network.
(i) scope	see below.
(i) type	see below.
(i) usage	see below.

Returns:

(s)	a tab-delimited list of network resources.

This function returns a tab-delimited list of network resources which are located immediately below the specified 'net-resource'.

'Scope' can be one of the following:

Req #	Meaning
1	All currently connected resources
2	All resources on the network
3	All remembered (persistent) connections

'Type' can be one of the following:

Req #	Meaning
0	All resources
1	All disk resources
2	All print resources
3	All disk and print resources

'Usage' can be one of the following:

Req #	Meaning
0	All resources
1	All connectable resources
2	All container resources
3	All connectable and container resources

Note: 'usage' is ignored unless 'scope' == 2.

Example:

```
ResourceRoot=""
ScopeConnectted=1
ScopeAll=2
ScopePersistant=3
TypeAll=0
TypeDisk=1
TypePrint=2
TypeDiskAndPrint=3
UsageAll=0
UsageConnectable=1
UsageContainer=2
UsageConnAndCont=4
oncancel="exit"
AddExtender(strcat(DirHome(),"WWW9534i.DLL"))
aaa=ResourceRoot
aaalist=w95Resources(aaa, ScopeAll, TypeAll, UsageAll)
level=0

while 1
   aaa%level%=aaa
   if level==0 then ButtonNames("Down","Exit")
               else ButtonNames("Down","Up")
   level=level+1
   goto downalevel

   :upalevel
   if level==0 then exit
   level=level-1
   if level==0 then ButtonNames("Down","Exit")
               else ButtonNames("Down","Up")
   aaa=aaa%level%

   :downalevel
   oncancel="goto upalevel"
   aaalist=w95Resources(aaa, ScopeAll, TypeAll, UsageAll)
   aaa=AskItemList("Resource Viewer",aaalist,@tab,@sorted,@single)

endwhile
exit
:CANCEL
%oncancel%
exit
```

w95ServiceAt(server, domain, server-type, service-name, flags)

Lists all servers in a domain which contain a specified service.

Syntax:

w95ServiceAt(server, domain, server-type, service-name, flags)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(s) domain	the name of the domain which will be used (e.g., "SALES"), or ("") for the primary domain.
(i) server-type	identifies the type of servers which will be examined. See below.
(s) service-name	the name of the service to be looked for.
(i) flags	specifies information on the service being looked for. See below.

Returns:

(i)	a tab-delimited list of server UNC names (e.g., "\\MYSERVER").

Server-type

Specify **-1** for all servers. Or, specify one or more of the following flags, combined using the binary OR ("|") operator.

Req#	Server Type
1	All LAN Manager workstation
2	All LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT (either workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT Non-DC server
65536	Server that can run the browser service
131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser

4194304	Windows 95 or newer
-2147483648	Domain announcement

Service-Name

'Service name' is the name of a service (e.g., "Spooler") or driver (e.g., "Atdisk"). The name can be specified either as the "display name" which is listed in Control Panel (name-type = 0) or the "service name" which is the actual registry key for the service (name-type = 1000). The SDK documentation describes them as:

DisplayName = a string that is to be used by user interface programs to identify the service.
ServiceName = a string that names a service in a service control manager database.

So, the following two commands will yield identical results:

```
servers = wntServiceAt("", "", -1, "Browser", 101)  ; display name
servers = wntServiceAt("","", -1,"Computer Browser", 1001) ;service name
```

Flags

'Flags' specifies information on the service being looked for. It consists of one entry from each of the following three groups, added together:

Req#	service type
1	services
2	drivers
3	both

Req#	service state
100	active services
200	inactive services
300	both

Name type indicates what the **'service-name'** parameter represents.

Req#	name type
0	display name (the name shown in Control Panel)
1000	service name (the actual registry key name)

Note: This function can take a while to run, depending on how many servers are in the domain. Also, it will only return the names of servers which it is able to access, which requires that the user have browse access to their service control managers.

Example:

```
;return a list of all servers running the "Spooler" service
servers = w95ServiceAt("", "", -1, "Spooler", 101)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)
```

```
;return a list of all 95 machines with an "Atdisk" driver installed
servers = w95ServiceAt("", "", 4096, "Atdisk", 302)
AskItemList("Lan Manager Servers",servers,@tab,@sorted,@single)
```

w95ServiceInf

Returns information about a server's type.

Syntax:

w95ServiceInf(server-name)

Parameters:

(s) server-name — the UNC name of a server (eg, "\\SERVER1"), or a blank string ("") to indicate the local machine.

Returns:

(i) — a bitmask indicating the type of server, or 0 on error. The individual flag bits in the bitmask can be extracted using the binary AND ("&") operator. See below.

Req#	Server Type
1	All LAN Manager workstation
2	All LAN Manager server
4	Any server running with Microsoft SQL Server
8	Primary domain controller
16	Backup domain controller
32	Server running the timesource service
64	Apple File Protocol servers
128	Novell servers
256	LAN Manager 2.x Domain Member
512	Server sharing print queue
1024	Server running dialin service
2048	Xenix server
4096	Windows NT (either workstation or server)
8192	Server running Windows for Workgroups
16384	Microsoft File and Print for Netware
32768	Windows NT Non-DC server
65536	Server that can run the browser service
131072	Server running a browser service as backup
262144	Server running the master browser service
524288	Server running the domain master browser
4194304	Windows 95 or newer
-2147483648	Domain announcement

Example:

```
AddExtender("WWW9534i.DLL")
servertype=w95ServiceInf("\\SERVER1")
title=strcat("Server Type : ",servertype)
if servertype == 0
  Message("Error","There was an Error with the function w95ServiceInf")
Else
   if (servertype & 4194304)
      message(title, "Windows 95 or newer")
```

```
   Else
      if (servertype & 128)
         message(title, "Novell server")
      Else
         If ( servertype & (8|16))
             message(title,"Primary or Backup Domain Controller")
         Else
             Message(title, "Other")
         EndIf
      Endif
   Endif
Endif
```

See Also:

w95ServerType , wntServerType

w95ServerInfo(server-name, request)

Returns information about a server.

Syntax:

w95ServerInfo(server-name, request)

Parameters:

(s) server-name	the UNC name of the server on which the function will execute (e.g., "\\MYSERVER"), or ("") for the local computer.
(i) request	specifies the information about "server-name" to be returned. See below.

Returns:

(i)	depends on request.

Req#	Description
1	(i) major version
2	(i) minor version
3	(i) type (see wntServiceAt for a list of server types)
4	(s) comment

Example:

```
AddExtender("WWW9534i.DLL")
info=w95ServerInfo( "", 1)
Message("Major Version",info)
exit
```

See Also:

w95ServerType

w95ServerType

Returns a server's platform.

Syntax:
w95ServerType(server-name)

Parameters:

(s) server-name	the UNC name of a server (eg, "\\SERVER1"), or a blank string ("") to indicate the local machine.

Returns:

(i)	See below.

Returns one of the following values:

Value	Meaning
0	Invalid server name
1	Other
2	Windows for Workgroups
3	Windows 95 or later
4	Windows NT

Example:

```
; This example returns a servers platform

AddExtender("WWW9534i.DLL")
type=w95ServerType("\\Server1")
switch type
 case 0
   message("Error","invalid server name")
   break
 case 1
   message("Server Type is:","Other")
   break
 case 2
   message("Server Type is:","Windows for Workgroups")
   break
 case 3
   message("Server Type is:","Windows 95 or later")
   break
 case 4
   message("Server Type is:","Windows NT")
   break
EndSwitch
```

See Also:

wntServerType

w95ShareAdd(server-name, resource, share-name,share-type, flags)

Shares a resource.

Syntax:

w95ShareAdd(server-name, resource, share-name,share-type, flags)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) resource	identifies the object to be shared. Resource can be a directory name (e.g., "c:\util"), a printer object (e.g., "HP LaserJet III"), or the name of a sharable device.
(s) share-name	name by which other users will access the resource.
(i) share-type	see below.
(i) flags	see below.

Returns:

(s)	1.

Share-type

The type of 'resource' is identified by the parameter 'share-type' and can be one of the following.

Req#	share type
0	directory
1	printer
2	device

Flags

'flags' specifies the access type being granted to other users. If share-level access control is being used, then **'flags'** can be one of the following:

100001	read-only
100002	full
100003	depends on password

If user-level access control is being used or if 'resource' specifies a printer, then **'flags'** should be set to 100002.

Note: If you specify a name of a network file server, then that machine must have "user-level security" enabled via the Control Panel. Otherwise, you can only work with shares on the local machine.

Example:

```
; This example adds a share "Public" referencing the C:\TEMP
; directory.  Share access will depend on the specified password

AddExtender("WWW9534i.DLL")
w95ShareAdd("","C:\TEMP","Public",0,100003)
w95ShareSet("","Public","My Public Directory","myfullpswd","myreadonlypswd")
```

See Also:

w95ShareDel, w95ShareSet

w95ShareInfo(server-name, share-name, request)

Returns information about a shared resource.

Syntax:
w95ShareInfo(server-name, share-name, request)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) share-name	name by which other users will access the resource.
(i) request	see below.

Returns:

(s)/(i)	a string or integer, depending on "request".

Note: If you specify a name of a network file server, then that machine must have "user-level security" enabled via the Control Panel. Otherwise, you can only work with shares on the local machine.

Request
Specifies the information to be returned, and can be one of the following:

Req#	Info returned
0	(s) share name
1	(s) resource
2	(s) comment
4	(s) full password
5	(s) read password
6	(i) share type
7	(i) flags

See w95ShareAdd and w95ShareSet for information on these values.

Example:

```
AddExtender("WWW9534i.DLL")
w95ShareAdd("","C:\TEMP","Public",0,100003)
type=w95ShareInfo("","Public",6)
message("share type",type)
```

See Also:
w95ShareDel, w95ShareAdd, w95ShareList

w95ShareList(server-name, share-type, flags)

Returns a list of shares on a server.

Syntax:
w95ShareList(server-name, share-type, flags)

Parameters:

(s) server-name	name of a network file server or empty string.
(i) share-type	a bitmask specifying the type(s) of shares that should be returned. See below.
(i) flags	reserved for future use and should be set to 0.

Returns:

(s)	a tab-delimited list of shared resources on a server, including hidden shares.

"share-type" can be set to 0 to return all shares, or it can be one or more of the following values combined with the bitwise OR ('|') operator:

Value	Share type
0	All shares
1	Disk drive
2	Print queue
4	Communication device
8	Interprocess communication (IPC)

Example:

```
AddExtender("WWW9534i.DLL")
w95ShareList("\\SERVER", 1 | 2, 0)
AskItemList("List of shares",list, @TAB, @UNSORTED, @SINGLE)
exit
```

See Also:
w95ShareDel, w95ShareAdd, w95ShareInfo

w95ShareSet(server-name, share-name, comment, full-password, read-password)

Sets additional share information for a resource.

Syntax:
w95ShareSet(server-name, share-name, comment, full-password, read-password)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) share-name	name by which other users will access the resource.
(s) comment	a text string used to describe the share or an empty string ("").
(i) full-password	password or an empty string (""), see below.
(i) read-password	password or an empty string (""), see below.

Returns:

(s)	**1**.

Note: If you specify a name of a network file server, then that machine must have "user-level security" enabled via the Control Panel. Otherwise, you can only work with shares on the local machine.

The parameters **'full-password'** and **'read-password'** are determined the type of access control and other specified parameters.

If share-level access control is being used, and 'share-name' specifies a directory, then **'full-password'** specifies the password that is required for full access, and **'read-password'** specifies the password that is required for read-only access.

If share-level access control is being used, and 'share-name' specifies a printer, then **'full-password'** specifies the password that is required for access. Set **'read-password'** to ("") an empty string.

If user-level access control is being used, then both **'full-password'** and **'read-password'** should be set with empty strings, ("").

Example:

```
; This example adds a share "Public" referencing the C:\TEMP
; directory.  Share access will depend on the specified password

AddExtender("WWW9534i.DLL")
w95ShareAdd("","C:\TEMP","Public",0,100003)
w95ShareSet("","Public","My Public Directory","myfullpswd","myreadonlypswd")
```

See Also:
w95ShareAdd, w95ShareDel

w95ShareUsers(server-name, share/computer-name, share-type, format, flags)

Returns a list of users connected to a share or server.

Syntax:
w95ShareUsers(server-name, share/computer-name, share-type, format, flags)

Parameters:

(s) server-name	name of a network file server or empty string ("") to indicate the current machine.
(s) share/computer-name	specifies either a share name (eg, "PUBLIC") on "server-name", or a computer name (eg, "\\FRED"). See explanation below.
(i) share-type	a bitmask specifying the type(s) of shares that should be returned. See below.
(i) format	specifies the information that will be returned for each connection. See Below.
(i) flags	reserved for future use and should be set to 0.

Returns:

(s) .

"share-type" is a bitmask specifying the type(s) of shares that should be returned if "share/computer-name" specifies a computer name. It can be set to 0 to return all shares, or it can be one or more of the following values combined with the bitwise OR ('|') operator:

Value	Share type
0	All shares
1	Disk drive
2	Print queue
4	Communication device
8	Interprocess communication (IPC)

"format" specifies the information that will be returned for each connection. It can be one of the following values:

Value	Return format
1	username
2	netname
3	username|netname

This function can return either a list of users connected to a share, or a list of computers connected to a server. If "share/computer-name" specifies a share name, then the function will return a list of users connected to that share. If "share/computer-name" specifies a computer name, then the function will return a list of shares on "server-name" (filtered by "share-type") to which that computer

is connected.

This function returns a tab-delimited list of user names and/or net names, depending on "format". The values returned are as follows:

username:
If "server-name" is running with user-level security, then "username" will be the name of the user who made the connection. If "server-name" is running with share-level security, then "username" will be the computer that made the connection.

netname:
"netname" is the opposite of "share/computer-name". In other words, if "share/computer-name" specifies a share name, then "netname" will be the name of the computer connected to that share, and if "share/computer-name" specifies a computer name, then "netname" will be the name of the share to which that computer is connected.

Example:

```
AddExtender("WWW9534i.DLL")
list=w95ShareUsers("", "PUBLIC", 1, 1, 0)
AskItemList("list of users connected to PUBLIC",list, @TAB, @UNSORTED, @SINGLE)
exit
```

See Also:

w95ShareAdd, w95ShareDel, w95ShareInfo, w95ShareList, w95ShareSet

w95ShareDel(server-name, share-name)

UN-shares a resource.

Syntax:
w95ShareDel(server-name, share-name)

Parameters:

(s) server-name	name of a network file server or an empty string ("") to indicate the current machine.
(s) share-name	name by which other users will access the resource.

Returns:

(s)	**@TRUE** if the share was deleted; **@FALSE** if there was no share with the specified name.

Note: If you specify a name of a network file server, then that machine must have "user-level security" enabled via the Control Panel. Otherwise, you can only work with shares on the local machine.

Example:

```
;This example removes the "Public" share

AddExtender("WWW9534i.DLL")
rslt=w95ShareDel("","Public")
if rslt
   Message("w95ShareDel","Share Public removed")
else
   Message("w95ShareDel","Share Public not found")
endif
```

See Also:
w95ShareAdd, w95ShareSet
w95GetUser

w95Version( )

Returns the version of this Extender DLL.

Syntax:
w95Version( )

Parameters:
none

Returns:
(i) the version of number of this extender Dll.

This function is used to check the version number of this Dll in cases where older DLL's exist and alternate processing is desirable. Version numbers of newer versions will be larger than that of older versions.

Example:

```
AddExtender("WWW9534i.DLL")
a=w95Version()
Message("Dll Version",a)
```