KiXtart 95
Version 3.60

KiXtart 95 is a logon script processor and enhanced batch scripting language for computers running Microsoft® Windows® 2000 or Microsoft® Windows® 95/98 in a Windows Networking environment.

The KiXtart 95 free-format scripting language is used to display information, set environment variables, start programs, connect to network drives, read or edit the registry, and change the current drive and directory.

This latest update of KiXtart 95 offers various new features and enhancements:

New commands, functions, macros

DecToHex()	returns hexadecimal representation of a decimal value.
Dir()	allows enumeration of directories.
Execute()	executes a piece of script.
GetFileAttr()	retrieves attributes of a file.
GetFileSize()	returns the size (in bytes) of a file.
SetFileAttr()	sets attributes of a file.
SetFocus()	sets focus to a specific application.
SendKeys()	sends keystrokes to the application that currently has the focus.
@SITE	name of the site in which the system currently resides.

Enhanced commands, functions

Arrays	KiXtart now supports the concept of single dimension arrays of variables.
Hexadecimal numbers	numbers can now be specified in hexadecimal notation.
Local variables	variables can now be declared as local to a specific script or subroutine.
GetFileVersion()	can now return the language of a file.

Note

Mathematical And and Or	2 new operators have been added (& and |) which enable mathematical And and Or operations.

Enhancements

Windows 2000 support	this version of KiXtart 95 has been enhanced to work both on Windows 2000 clients and in combination with Windows 2000 servers.
GetDiskSpace	enhanced to work correctly on pre-OSR2 Windows 95 (but it is still limited to a maximum of 2 gigabytes (GB)).
AT	fixed bug in AT that could occur when output was redirected to a file.
COPY	corrected handling of target that did not end with a BACKSLASH (\) or file name.
Internationalization	both KIX32 and KXRPC have been 'internationalized'. As a result, all macros are now available in the correct codepage, both on Windows 2000 as well as on Windows 95/98.
'Smart' information retrieval	in previous versions of KiXtart, all user information was automatically retrieved at the startup of Kix32. This behavior has been changed so that at startup, only local information is retrieved, and any information that requires access to the network is only retrieved if and when it is requested. This enhancement greatly speeds up the startup of KiXtart, and makes it possible to make scripts intelligent about network access, for example depending on whether or not the current system is connected via remote access server.

For information about the latest changes to KiXtart 95, see Kix32rel.txt, in the Kix32 subdirectory.

System Requirements

KiXtart 95 is supported on systems with an Intel 80386 or better microprocessor running Windows 2000 (Server and Professional), Windows NT 3.5 and later, and

Windows NT version 4.0 and later (Server and Workstation), Windows 95 or Windows 98 or later (referred to in this document as Windows 95/98).

KiXtart is also available for the MS-DOS platform. Please check the following Web locations for the latest versions available:

http://netnet.net/~swilson/kix.html
http://script.kixtart.to
http://www.scriptlogic.com
http://www.comptrends.com
http://www.cyberramp.net/~musicon/kix/index.html

KiXtart 95 Files

The Kix95 directory contains the following files.

Kix32.doc	This document
Kix32.exe	KiXtart 95 program file
Kxrpc.exe	KiXtart RPC service for Windows 95 clients
Kx95.dll	Dynamic link library (DLL) for KiXtart on Windows
Kx16.dll, Kx32.dll	Support DLLs to connect to Netapi.dll on Windows
Kixplay.exe	16-bit tool to play SPK files on Windows
Delkey.kix, Demo.kix, Demo2.kix, Enumdir.kix, Enumkeys.kix, Fly.kix, Fun.kix, Kick.kix, Kixtart.kix, Recur.kix, Test.kix	Sample script files
Adaams.spk, Bouree.spk, Cabaret.spk, Jbond.spk, Treksong.spk	Sample SPK files
Chimes.wav	Sample WAV files
Kix32rel.txt	Release notes, containing information about the latest changes to KiXtart 95

The subdirectory called ALPHA contains the ALPHA-specific executable extensions (KIX32.ALPHA and KXRPC.ALPHA). To install these executable files, copy the files to your system and change the file extension to .exe.

The subdirectory called Xnet contains Xnet.exe and a sample batch file that demonstrates the use of XNET to install the KiXtart RPC service on a remote computer.

Installing KiXtart

KiXtart consists of five executable components:

- Kix32.exe, the main program file

- Kx16.dll, a 16-bit DLL used to connect to Netapi.dll on Windows clients
- Kx32.dll, a 32-bit DLL used to connect to Netapi.dll on Windows clients
- Kxrpc.exe, a Windows NT service to support Windows clients
- Kx95.dll, a 32-bit dynamic link library (DLL) used to connect to the KiXtart RPC service

All executable components can be installed on and run from the network or from the local hard disk of the client systems.

To install KiXtart 95 on the network

To install KiXtart on the network, copy the required files to the NETLOGON share of the logonserver(s).

To install KiXtart 95 on a client

To install KiXtart on a client, copy the required files to a directory on the local hard disk. Optionally, the dynamic-link libraries (DLLs) can be copied to the windows or the windows\system directory.

To install the KiXtart RPC service

1. Copy Kxrpc.exe to a directory on the server that will run the service.
2. At the command prompt, switch to that directory and type the following command:

   KXRPC –install

The KiXtart RPC service can be installed on a remote server using a tools such as RSERVICE or XNET. KiXtart 95 comes with a batch file called Xinst.cmd, which demonstrates installing the KiXtart RPC service on a remote computer using XNET.

The KiXtart RPC service is available for both Intel and Alpha systems. To install the Alpha version, please copy the file called KXRPC.ALPHA to the Alpha system, and rename the file to KXRPC.EXE.

The KiXtart RPC service should only be installed when necessary. Please see the separate chapter on the KiXtart RPC service for details on when and where to install the service.

Required files for Windows 2000 Clients

Windows 2000 clients need only install Kix32.exe.

Required files for Windows Clients

Windows clients must install both Kix32.exe and two dynamic-link libraries (DLLs) called KX16.DLL and KX32.DLL.

If Windows clients are to communicate with the KiXtart RPC service, an additional DLL, called KX95.DLL, should also be installed. Please see the separate paragraph on the KiXtart RPC service for full details.

KX95.DLL should only be installed if the KiXtart RPC service will be used. Without the KiXtart RPC service, KX95.DLL will generate unnecessary network traffic and delay the start of KiXtart.

Uninstalling KiXtart

To uninstall KiXtart 95, simply delete the executable components and scripts.

The KiXtart RPC service can be removed at the command prompt by typing the following command :

KXRPC –remove

Updating from previous versions

To update KiXtart for Windows 2000 clients, replace Kix32.exe.

To update KiXtart for Windows clients, make sure to replace **all** components: KIX32.EXE, KX32.DLL, KX16.DLL. If the KiXtart RPC service is used, make sure to also replace KX95.DLL and KXRPC.EXE.

Failing to replace all the components can cause unexpected behavior. As a precaution, KiXtart checks for the correct components and will report an error in KIXTART.LOG if it finds an outdated component.

To update the KiXtart RPC service, stop the service (NET STOP KXRPC), replace KXRPC.EXE and restart the service.

KiXtart 95 and the year 2000

There are no known issues with KiXtart 95 and the year 2000.

KiXtart 95 uses standard Win32 API's to retrieve date information that is available through the macros (such as @YEAR and @DAY).

The only calculation in KiXtart involving dates (to determine the @YDAY macro) does take the year 2000 correctly into account.

Note

Thunking and the KiXtart RPC Service

Unlike Windows 2000, Windows does not provide all the Win32 APIs that KiXtart 95 needs to gather information, such as the user's full name and group memberships. KiXtart uses two programming methods to solve this problem: thunking and Remote Procedure Calls (RPCs)

Thunking is the term used when connecting to a 16-bit API from a 32-bit application. The 16-bit APIs required by KiXtart are provided by Netapi.dll. Kx16.dll and Kx32.dll provide the so-called thunking layer required to connect to Netapi.dll.

Unfortunately, Netapi.dll does not provide all the information that is of interest to KiXtart. Most notably, Netapi.dll does not provide access to the logon domain, the security identifier (SID), the primary group, the home drive and local groups. The KiXtart RPC service provides these missing pieces of information to KiXtart using RPCs. The client side of the RPC interface is provided in Kx95.dll

The server side of the RPC interface is provided in Kxrpc.exe, and this should be installed and run on one or more Windows 2000 systems. The KiXtart RPC service can run on any Windows 2000 system: a workstation, a standalone server, or a logon server. The system must be either a member of the logon domain or a member of a resource domain that has a trust relationship with the logon domain.

Using the KiXtart RPC service is optional. However, without it, extended information, such as local groups, is not available to Windows systems.

Choosing Where to Install the KiXtart RPC Service

When considering where to install the KiXtart RPC service, you must decide how KiXtart 95 locates servers running the KiXtart RPC service. The simplest choice is to install the KiXtart RPC service on all the logon servers in the logon domain, which also provides load balancing. KiXtart automatically attempts to connect to the KiXtart RPC service on the logon server that authenticated the user.

If the KiXtart RPC service cannot be installed on all logon servers, KiXtart must be directed to locate a specific server running the service. This can be achieved by:

- Setting an environment variable before running KiXtart 95.
- Adding a subkey to the registry of Windows clients.
- Adding an initialization file to the KiXtart 95 startup directory.

These methods are described in the following sections.

Note

Setting a KXRPC Environment Variable

The KXRPC environment variable is set to a comma-delimited list of the full name of the server running the KiXtart RPC service. For example:
set kxrpc= *MyServer*
– Or –

set kxrpc= *MyServer,\\AnotherServer*

Adding a KiXtart Subkey to the Windows Registry

Another way to direct KiXtart to a server running the KiXtart RPC service is to add the following subkey to the registry of Windows clients:

HKEY_LOCAL_MACHINE\Software\Microsoft\KiXtart
In the new **KiXtart** subkey, add an entry called **KXRPC** with a **REG_SZ** data type. Set the value of **KXRPC** to a comma-delimited list of the full names of the KiXtart RPC servers.

Adding a Kixtart.ini File

KiXtart can also be directed to the KXRPC server by creating a Kixtart.ini file and placing it on the NETLOGON share of the logon server, or in the directory from which KiXtart is started.

Kixtart.ini contains a `[KXRPCMapping]` section, which can include an entry for each domain or workgroup that is to be enabled for use of KiXtart 95. Optionally, a `Default=` entry can be added to refer all unknown workgroups or domains to a specific KXRPC server.

The following is a sample Kixtart.ini file:

```
[KXRPCMapping]
MyDomain=\\MyServer1,\\MyServer2,\\MyServer3
YourDomain=\\YourServer
Default=\\ServerA,\\ServerB
```

If multiple KXRPC servers are specified for one mapping, KiXtart connects to them in the sequence specified.

Starting the KiXtart RPC Service

When it is installed, the KiXtart RPC service is configured to start automatically at system startup. After the initial installation, the service can be started from the Services feature in Control Panel or from the command prompt.

To start the KiXtart RPC service

1. In the Services feature in Control Panel, select the KiXtart RPC service, and then click **Start**.

   – Or –

2. At the command prompt, type the following command:

   net start kxrpc

Running KiXtart

KiXtart can be run manually or automatically during the logon sequence.

To run KiXtart manually

- At the command prompt, type the following command:
  kix32

To run Kix32.exe automatically when a user logs on

1. In User Manager, select the user.
2. On the **File** menu, click **Properties**, and then click **Profile**.
3. In the **Logon Script Name** box, type **"Kix32"**.

For Windows clients, do not specify a KiXtart script in the **Logon Script Name** box in the **User Environment Profile** dialog box in User Manager. To specify a script for Windows clients, use a batch file as the logon script, and start KiXtart from the batch file.

Running KiXtart from a Batch File

Kix32.exe can be run from a batch file that is used as the logon script for the user. For example, if Kix32.exe is in the root directory of the NETLOGON share, the batch file might contain the following commands:

```
@ECHO OFF
%0\..\Kix32.exe
```

Use of the syntax **%0\..** is discussed in *Knowledge Base* article Q121387.

If Kix32.exe was installed on the client's local hard disk, you must refer to the local directory, for example: C:\Kixtart\Kix32.exe. By default, KiXtart automatically looks for a personal script for the current user (*Username*.kix). If it does not find one, it looks for the default script, Kixtart.kix. You can override this behavior by

specifying one or more scripts after Kix32.exe on the commandline. If an extension is not specified, KiXtart attempts to use two default extensions: ".KIX" and ".SCR".

KiXtart also supports declaring variables at the command prompt, demonstrated in the following example:

kix32 Demo.kix $Key=HKEY_LOCAL_MACHINE\Software
For information about valid variable names and values, see "Dynamic Program Variables" later in this document.

On computers running Windows , KiXtart can also be started by using Lmscript emulation. For more information, see "Lmscript Emulation" later in this document.

Locating Files

During the logon sequence, KiXtart 95 automatically tries to locate all files that it is asked to open (SPK, WAV, TXT, and so on) by searching for them first on the NETLOGON drive, then on the drive where KiXtart 95 was started from, and finally in the current directory. This behavior can be overridden by prepending the file name with a drive letter or a UNC path.

For example, the following command:
```
play file "Kbond.spk"
```
causes KiXtart to search for Kbond.spk on the NETLOGON share, in the KiXtart startup directory, and in the current directory.

If this command is used:
```
play file "C:Kbond.spk"
```
KiXtart searches for Kbond.spk only in the current directory of the C drive.

Running KiXtart with Lmscript Emulation

Normally, when a user logs on to a LAN Manager or Windows 2000 domain from Windows , the Windows API responsible for processing the logon request starts a program called Lmscript to run the logon script. The sole responsibility of Lmscript is to inform the logon API when the logon script has finished by creating a semaphore file (also called a cookie).

Unfortunately, the original Lmscript.exe takes up a lot of memory. To solve this issue, KiXtart can be used as a replacement for Lmscript.exe. This not only saves memory, but also means that the Kix32.exe does not have to be read from the network during the logon sequence, as it is automtically run from the local hard disk. The benefit of this is minimal in a normal LAN environment, but can be substantial in a WAN or RAS environment.

Note

To enable Lmscript emulation on computers running Windows

1. In the Windows\System folder, rename the original Lmscript.exe.
2. Rename Kix32.exe to Lmscript.exe and then copy it to the Windows\System folder.
3. In User Manager, in the **Logon Script Name** box, specify a KiXtart script as the logon script for the user (for example, Kixtart).
4. At the end of the specified KiX script, add a line containing the COOKIE1 command to create the semaphore file.

Users who do not use Lmscript emulation (such as users running Windows on the LAN or users running Windows NT Workstation) cannot run the logon script unless there is also a batch file with the same name as the KiX script specified for the user.

The following example illustrates the use of such a batch file for a user named Fred.

User name	Fred
Logon script	Script1
Contents of the Scripts directory on the logon server	Script1.bat Script1.kix Kix32.exe

Contents of Script1.bat	`@ECHO OFF`
	`%0\..\Kix32 Script1`
	`EXIT`

Contents of Script1.kix	`CLS`
	`BIG`
	`? "Hi, @USERID"`
	`SLEEP 10`
	`COOKIE1`
	`EXIT`

If Fred uses a computer running Windows NT to log onto the network, or if he uses a computer running Windows with the original Lmscript.exe, Script1.bat starts and then in turn starts Kix32.exe with Script1.kix as the logon script. If he uses a computer running Windows and logs on with Kix32.exe renamed as Lmscript.exe, Script1.kix runs automatically.

Troubleshooting KiXtart

KiXtart provides extensive logging of system errors, such as failure to locate support DLLs, failure to connect to the RPC service, and so on. On computers running Windows NT, these errors are logged in the system event log. On computers running

Windows , they are logged in a text file named Kixtart.log, which is stored in the Temp or Windows directory.

The following table describes the most common problems encountered by KiXtart.

Error	Meaning	Solution
The macro @ADDRESS returns an empty string ("").	KiXtart failed to find a NetBIOS interface on any of the network bindings.	Make sure a NetBIOS interface is available on one of the bindings.
The macro @FULLNAME returns an empty string ("").	KiXtart cannot retrieve the network information.	On Windows NT, make sure the Workstation service is running. On Windows , make sure that the support DLLs are available. Also check the event log or kixtart.log for any errors.
KiXtart seems to hang when at startup.	KiXtart cannot connect to the logon server to retrieve user information.	Determine the name of the logon server (using @LSERVER) and try to connect to it manually.
KiXtart does not recognize certain commands.	Although KiXtart is a free-format language, some literals, such server names that contain a hyphen (-), can cause errors.	Enclose literals in quotation marks.
Errors such as “Label not found” or “Unknown command” appear in an otherwise faultless script.	There is probably an unmatched quotation mark or similar error somewhere in the script.	Proofread your script.
Failed to initialize RASMAN.DLL.	Caused by a ‘half-installation’ of the RAS client software. Installation either wasn’t completed, or the uninstallation left RASAPI32.DLL on the system.	Remove RASAPI32.DLL from the system, or complete the installation of the RAS client.
Application error c0000006H / ‘IN_PAGE_ERROR’ / ‘SWAP_ERROR’ or an Invalid Page Fault is generated intermittently.	The operating system has failed to read code from executable file(s) because the KiXtart startup drive has become unavailable.	Make sure that you do **not**, in any way, disconnect or re-redirect the drive from which KIX32.EXE was started. Also, these faults can be caused by antivirus software. If you use antivirus software, make sure you are using the latest

version and if the problem persists, test if disabling the antivirus software solves the problem.

Because KiXtart 95 is a true 32-bit application, it writes to the screen using the Win32 Console API. For this reason, screen output may seem slower than the direct screen output used by MS-DOS – based versions of the program.

To include quotation marks in a string, either use the **CHR** function, or enclose the entire string in quotation marks. For example,

```
"String with a quote (') in it."    String with a quote (') in it.
```

Known Problems of KiXtart on Windows

The following is a list of known bugs and other issues that may be encountered when using KiXtart on Windows :

- If KiXtart is used on systems that are configured to run both Microsoft Networking client software and Novell Netware client software, compatibility issues can cause KiXtart to fail to retrieve network information and/or find any script. If these problems occur, make the following change in the registry of the affected clients:

HKEY_LOCAL_MACHINE
  System
    CurrentControlSet
      Services
        MSNP32
          Network Provider CallOrder [00 00 00 40] >change to> [00 00 00 20]
        NOVELLNP
           Network ProviderCallOrder [00 00 00 20] >change to> [00 00 00 40]

- When text is output to bottom-right position of the screen, the screen scrolls.
- This issue is related to the Console API on Windows .
- Color is sometimes garbled when the screen is scrolled.
- This problem is caused by the way Windows handles color attributes.
- The LockLog utility (shipped with the MS-DOS – based version of KiXtart) does not work on Windows .
- The **SET** and **SETM** commands do not work on Windows .
- As a work around run Winset.exe (available on the Windows CD) from within KiX32 using the **SHELL** command.
- On Windows , **SAVEKEY** produces a hidden, read-only system file in the Windows System directory. On Windows 2000, the same command produces a normal file in the current directory.

- In either operating system, the file can be used with **LOADKEY** (after it has been made visible using **ATTRIB**).

- On Windows , if a network drive is removed that was redirected from My Computer or Windows Explorer, the drive remains visible in the Windows interface as a disabled or ghosted drive, and the drive is reconnected when the user clicks it.

- This scenario can be prevented with an additional step. After the drive has been removed, delete the corresponding subkey from the registry. For example:
  ```
  USE E: /d
  DELKEY("HKEY_CURRENT_USER\Network\Persistent\E")
  ```

- The logon script is sometimes skipped completely.

- This problem can be caused by a sharing bug in Msnet32.dll. The bug was fixed in version 4.00.951 of Msnet32.dll.
- Another reason for the logon script to be skipped on Windows is a space in the logon script field in NT User Manager. Although NT User Manager accepts multiple strings (and spaces) in the logon script field, Windows fails to run the logon script.
- The **ShutDown** function does not work reliably.
- This problem is caused by the underlying Windows API. It may be fixed in a future version of Windows . As a workaround, try the following command :
- `SHELL "%windir%\RUNDLL32.EXE user.exe,ExitWindows"`

The 'MAP ROOT' issue.

The Windows redirector software on Windows systems does not support the concept of socalled 'deep' redirections (ie: redirecting a drive to a directory below the sharelevel, eg: "\\SERVER\SHARE\USER"). As such, Novell's MAP ROOT feature cannot be emulated. This is a limitation of the redirector software, and unfortunately, KiXtart cannot work around this.

Deep redirections ***are*** possible on Windows 2000 systems, either by using the SUBST command, or by installing the Distributed FileSystem (DFS) client software.

Debug mode

KiXtart provides a simple debug mode. In debug mode, a script can be stepped through statement by statement, and the value of a variable or macro can be displayed. To run a script in debug mode, specify '/d' on the command line.

In debug mode, the top line of the screen is used to display the current line in the script starting at the current statement. Optionally, the second line of the screen is used to display the value of a specific variable or macro.

Whilst debugging, the following keys are available to control script execution :

F5	Run (deactivates debug mode, runs rest of script)
F8, <Space>, <Enter>	Step into (run a single statement, follow thread into subroutines and secondary scripts)
F10	Step over (run a single statement, executes, but skips over subroutines and secondary scripts as far as the debugger is concerned)
<Esc>, 'q'	Exit

Additionally, the value of a variable or macro can be queried simply by typing its name and pressing <Enter>.

General Syntax Rules

KiXtart is a free-format scripting language. It is not case-sensitive. This means that

```
IF @PRIV="ADMIN" DISPLAY "ADMIN.TXT" ELSE DISPLAY "USER.TXT"
ENDIF
```

is equivalent to

```
If @PRIV = "ADMIN"
      Display "ADMIN.TXT"
Else
      Display "USER.TXT"
Endif
```

When using KiXtart, note the following rules:

- Strings can contain any characters, except the **\0** (NULL) and **\x1a** (end of file) characters.
- Script commands should be separated by white space — that is, any combination of spaces, tabs, or new line characters.
- If a string contains delimiters (-, +, *, and so on), the string must be enclosed in quotation marks. For example

```
'String with a dash (-) in it.'      ; String with a dash (-) in it.
```

Dynamic Program Variables

Introduction

In KiXtart, variables are used to temporarily store values during the execution of a script. Variables have a name (the word you use to refer to the value the variable contains) a type (which determines the kind of data the variable can store) and a scope (which determines where in the script you can reference the variable). You can think of a variable as a placeholder in memory for an unknown value.

Storing Data in Variables

Variables can be assigned a particular value by using an assignment statement :

```
$Variable = 10
```

or by using a GET or GETS statement :

```
GET $Variable
```

Optionally, variables can be created and assigned a value on the command line with which KiXtart is started. To do this, type the variable name followed by an equal sign (=) and the value the variable should have. For example:

```
KIX32 Demo.kix $Key=Value
```

On the command line, do not include spaces between the equal sign (=) and the value. If you want to specify a value that contains spaces, enclose it in quotation marks (for example, `KIX32 Demo.kix $Key="Hi there"`).

Declaring Variables

To declare a variable is to tell the program about it in advance. You declare a variable with the Dim or the Global statement, supplying a name for the variable:

```
DIM variablename
```

Variables declared with the Dim statement exist only as long as the script is executing. When the script finishes, the variable, and its value, disappear. Variables declared with the Global statement exist during the entire KiXtart session.

A variable name:

- Can't contain operator characters (+,-,*,/,&,<,>,=)
- Must not exceed 14 characters.
- Must be unique within the same *scope,* which is the range from which the variable can be referenced in a script, or script segment.

You can use the same name for variables in different scopes, and if you do, you will only be able to reference the variable in the current scope. Please see the example below for more details:

```
$Var = 10

IF InGroup( "Admins" )
```

```
   DIM $Var             ; local variable with same name
   $Var = 20
   ? $Var               ; this will display '20'
ENDIF
? $Var ; this will display '10'
```

Implicit declaration

Variables don't have to be declared before they can be used. You can also implicitly declare them simply by assigning a value to them. Note that all variables that are declared in this way have a global scope (see below for details on scope).

Scope of variables

Depending on how and where they are declared, variables can have a local or a global scope. Variables with a global scope are visible anywhere in any script during the entire KiXtart session. Variables with a local scope are only visible to the script or script segment in which they were created.

Examples:

```
$GlobalVariable = 10
```

Assuming this is the first reference to '$GlobalVariable', this variable is implicitly declared and will become a global variable, visible everywhere in every script during this KiXtart session.

```
DIM $LocalVariable
$LocalVariable = 10
```

This variable will become a local variable and will be visible only in the current script.

```
IF $X = 1
   DIM $LocalVariable
   $LocalVariable = 10
ENDIF
```

In this example, $LocalVariable will only be visible inside the IF statement.

```
GOSUB Demo
EXIT 1
```

In this example, $LocalVariable will only be visible inside the subroutine 'Demo'.

```
:Demo

DIM $LocalVariable

$LocalVariable = 10

RETURN
```

Variable types

There are two types of variables: string and integer. String variables can contain up to 32,000 characters. Integer variables can contain any value between -2,147,483,648 and 2,147,483,647. The type of a variable is automatically changed to the result of the expression that is assigned to it. This means that if you assign a string to an integer, the integer is changed to a string.

There is no limit on the number of variables that can be defined, other than the amount of memory available to KiXtart.

Arrays

KiXtart supports single dimension arrays. Arrays allow you to refer to a series of variables by the same name and to use a number (an index) to tell them apart. This helps you create smaller and simpler code in many situations, because you can set up loops that deal efficiently with any number of cases by using the index number. Arrays have both upper and lower bounds, and the elements of the array are contiguous within those bounds. Because KiXtart allocates space for each index number, avoid declaring an array larger than necessary.

Unlike normal variables, arrays must be declared explicitly before they can be used. When declaring an array, follow the array name by the upper bound in square brackets. The upper bound cannot exceed 2,147,483,647.

Examples:

```
Dim $Counters[14]
Dim $Sums[20]
```

The first declaration creates an array with 15 elements, with index numbers running from 0 to 14. The second creates an array with 21 elements, with index numbers running from 0 to 20.

Immediately after the declaration, all elements of an array have the same type: integer. Individual elements of an array can subsequently be changed to the string type (just as any other variable) simply by assigning a string expression to them.

Unlike regular variables, arrays can not be used inside strings and can also not be assigned a value on the command line.

Expressions

KiXtart supports two types of expressions: string and numeric.
A string expression can consist of any combination of the following:

- Literals (a sequence of characters enclosed in quotation marks)
- Functions that return a string
- Plus signs (+), which indicate concatenated sub-expressions

Numeric expressions can consist of any combination of:

- Sub-expressions
- Numeric values (in decimal or hexadecimal notation)
- Functions that return a numeric value
- Numeric operators (+, – , *, /, &, |)

KiXtart support the following numeric operators:

+	Used to sum two numbers.
-	Used to find the difference between two numbers or to indicate the negative value of a numeric expression.
*	Used to multiply two numbers.
/	Used to divide two numbers and return an integer result.
&	The & operator performs a bitwise mathematical AND operation on two numbers.
|	The | operator performs a bitwise mathematical OR operation on two numbers.

To specify a number in hexadecimal notation, prepend it with an ampersand (&).

Both string and numeric expressions can contain the following conditional and logical operators:

- <
- >
- =
- <>
- <=
- >=
- **And**
- **Or**

A string expression can contain up to 32,000 characters. Any macros, or references to environment strings within a string (for example: “String with the macro @USERID in it.”) are resolved before the string is evaluated. For compatibility reasons, references to variables inside strings (for example: “String with a $Var in

it.") are also resolved before the string is displayed. The only exception to this rule are arrays, which can not be used inside strings.

The characters **@**, **%**, or **$** are normally used to indicate macros, environment strings, or variables. If you want to use these characters in a string, use **@@**, **%%**, or **$$**.

The following examples show the correct use of expressions in KiXtart:

```
$X = 1 + "20"           ; $X type = integer / value = 21.

$X = &10 + &A           ; $X type = integer / value = &1A (26).

$X = "1" + "20"         ; $X type = string / value = '120'.

$X = @USERID + "1"      ; $X type = string / value = 'USER1'.

"Current time =  " + @time      ; prints: "Current time = 12:34:00"

"Use @@time to print the time"      ; prints: "Use @time to print the
time "

$Y = "And this is how you access environment variables: %USERNAME%..."

IF @Day='Sunday' AND @USERID = 'RuudV'

$X = (@MONTHNO=3 AND @MDAYNO>=20) OR @MONTHNO=4

IF @WKSTA="VLEERBEER" OR @WKSTA="PALOMINE"

$X = ((@YDAYNO + 7) / 7) + 1

; Old style use of variables inside a string:
"Use of a variable $Var inside a string."

New, preferred style to use variables in combination with strings:
"Use of a variable " + $Var + " inside a string."
```

Strings in the script are displayed on the screen in the current character size starting from the current cursor position. For information about character size, see the **BIG** and **SMALL** commands.

A string can be enclosed in single or double quotation marks. To specify quotation marks in a string, either use the **CHR** function or enclose the entire string in the opposite type of quotation marks — .that is, if you want to include single quotation marks in a string, enclose the string in double quotation marks, and vice versa.

The following examples show the correct use of string expressions in KiXtart:

Code	Output
"Hi "+ @userid	Hi Ruudv
'Double quote in a string: (")'	Double quote in a string: (")
"Single quote in a string: (')"	Single quote in a string: (')
"More double quote: " + Chr(34)	More double quote: "

KiXtart determines the type of the expression from the first element of the expression.

KiXtart Command Reference

KiXtart accepts the commands described in the following sections.

In this documentation, square brackets ([ ]) indicate optional arguments, and angle brackets (< >) indicate required arguments.

:

Defines a label within the script file to which you can transfer control.

:label

Labels must be unique within the script. You can define a label in an **IF** statement, but you cannot jump to one from outside that **IF** statement.

;

Indicates a comment. Subsequent characters on the script line are ignored.

;

?

Indicates a new line. This moves the cursor position to the beginning of the next line.

?

BEEP

Causes the system to beep.

BEEP

BIG

Changes the character mode to large characters.

BIG

When **BIG** is used, subsequent screen output is 8 characters wide and 8 characters high. Use **SMALL** to reset the character mode to normal.
BIG is ignored when screen output is redirected to a file.

BREAK

Enables (**BREAK ON**) or disables (**BREAK OFF**) the CTRL+C/BREAK keys and the **Close** command. This effectively allows control over whether a script run by KiXtart can be interrupted or not.
BREAK <**ON** | **OFF**>

By default, to prevent users from inadvertently interrupting a script, KiXtart automatically disables the CTRL+C/BREAK keys, disables the **Close** command in the **System** menu of the current command-prompt window, and hides the **Please wait while your logon script executes** message box on Windows .

In a multi-tasking environment such as Windows NT, you cannot fully prevent users from interrupting a program. (They can end programs by using the Task List, for example.) As an additional protection, on computers running Windows NT Workstation only, when **BREAK** is **OFF** (the default) KiXtart also installs a special event handler for the current console. The effect of this handler is that whenever a user forcibly terminates KiXtart, the user is automatically logged off. This means that you must be careful when testing scripts.

Tip : the **Please wait while your logon script executes** message box contains a **Cancel** button, which you can hide by opening a copy of Msnet32.dll in Visual C 4.00 WorkBench, selecting the LMWINSCRIPTDLG resource, and making the **Cancel** button invisible.

CALL

Runs a separate KiXtart script.

CALL "*script name*"

When the called script ends or when a **RETURN** statement is encountered, script execution continues at the statement following the **CALL** statement in the calling script.

Theoretically, there is no limit to the number of scripts that can be nested. Obviously, the practical limit on the number of scripts you can call is determined by the amount of available memory at the time KiXtart runs, the size of the scripts, the number of variables defined, and so on.

CD

Changes the current working directory to the directory specified.

CD "*directory*"

Check the value of @ERROR to see if **CD** was successful.

CLS

Clears the screen and moves the cursor to position 0,0.

CLS

The **CLS** command is ignored if all output has been redirected to a file using the **REDIRECTOUTPUT** function.

COLOR

Sets the color attribute to be used in subsequent display statements.

COLOR *Xx*/*Yy*

X	Foreground color
x	Optional intensity indication
Y	Background color
y	Optional blink indication

Possible values for the foreground and background colors are:

n	Normal (black)
b	Blue
g	Green
c	Cyan
r	Red
m	Magenta

y	Yellow/brown
w	White

If the foreground color is followed by a plus sign (+), the color is displayed with high intensity.

Specifying a plus sign (+) with the background color causes the color to be displayed blinking.

`COLOR w+/b`	Bright white text on a blue background
`COLOR g/r+`	Green text on a blinking red background

COOKIE1

Creates a *cookie*, or semaphore-file, that the Windows Logon API uses to determine whether the script has finished running. This command is only useful when KiXtart is being used to emulate Lmscript.exe. For more information, see "Lmscript Emulation," earlier in this document.

COOKIE1

COPY

Copies one or more files.

COPY "*source*" "*destination*" [/h]

Wildcard characters are supported.
If a file already exists at the destination, it is overwritten without warning.
The /h option can be used to include files with the hidden or system attribute set in the copy.

DEL

Deletes a file.

DEL "*file name*"

DEL does not prompt the user to confirm the deletion.
Wildcard characters are supported.

DIM

Declare one or more local variables.

DIM "*variable1*" [<,>"*variablex*"]

Local variables are visible only in the current script or script segment.

```
DIM $Variable

DIM $Array[9]   ; Note : declaration of an array of 10 elements.

IF $X = 1
   DIM $Var1, $Var2, $Var3
ENDIF
```

DISPLAY

Displays the contents of a file on the screen, starting at the current cursor position.
DISPLAY "*file name*"

DO UNTIL

Loops until an expression becomes true.

DO ... **UNTIL** "*expression*"

DO UNTIL loops can be nested as many times as memory allows.

EXIT

Exits the current KiXtart script, or, if used at the topmost level, exits KiXtart.

EXIT [*error level* / *exit code*]

If **EXIT** is followed by a numeric expression, then @ERROR is set to the value of that expression and you can check it in the calling script or batch file.

FLUSHKB

Flushes all pending characters from the keyboard buffer.

FLUSHKB

GET

Accepts a single character from the keyboard and stores the character in a variable.

GET $*x*

The character is stored in the specified script variable. If a function key, such as F1, is pressed, **GET** returns 0, and @ERROR returns the key code of the function key.

GETS

Reads a line of characters from the keyboard until the <ENTER> key is pressed, and stores the result in a variable.

GETS $*x*

GLOBAL

Declare one or more global variables.

GLOBAL "*variable1*" [<,>"*variablex*"]

Global variables are visible everywhere in every script during the current KiXtart session.

```
GLOBAL $X
GLOBAL $X, $Y, $Z
```

[GO]

Changes the current drive.

[**GO**] *drive*

Use **GO** if you want to specify a variable as the drive to change to.

```
GO A:
A:
GO $2
```

GOSUB

Causes script execution to continue at the first statement after a label.

GOSUB <*label*>

Label can be an expression.
When a **RETURN** statement is encountered, script execution continues at the statement following the **GOSUB** statement.

```
? "This demonstrates calling a subroutine"
GOSUB "Demo"
? "End of demonstration…"
EXIT 1
:Demo
? "We are in the subroutine now…"
RETURN
```

GOTO

Causes script execution to continue at the first statement after a label.

GOTO <*label*>

Label can be an expression.

```
GOTO "end"
$string = "end"
GOTO $string
```

IF ELSE ENDIF

Conditionally runs statements.

IF *expression*
statement1
....
[**ELSE**
statement2
....]
ENDIF

The body of an **IF** statement is executed selectively depending on the value of the expression. If *expression* is true, then statement1 is executed. If *expression* is false and the **ELSE** clause is specified, then statement2 is executed.

IF statements can be nested as many times as memory allows.

If the *expression* does not contain any relational operators, the condition is considered to be true if it is numeric and it evaluates to a value other than zero, or if it is alphanumeric and it evaluates to a string containing at least one character.

Comparisons are not case-sensitive.

```
IF $X                    ; similar to IF $X <> 0
IF @HOMESHR              ; similar to IF @HOMESHR <> ""
IF INGROUP("Domain Admins")      ; similar to IF INGROUP("Domain Admins")
> 0
IF INGROUP("Domain Admins") = 0  ; true if user NOT a Domain Admin
IF $X*2 < 10
IF (($X*2) < 10) OR ($Y + 100) /3 >120
IF INSTR(%PATH%,"NETLOGON") AND @DOS = "3.51"
IF (SUBSTR(@WKSTA,11,1)="1" AND @USERID = "PETERV") OR  @DOMAIN =
"VleerBeer"
IF @USERID = "RUUDV" OR @USERID = "WIMW"
IF (INGROUP("Domain Users") OR INGROUP("Users"))
```

MD

Creates a new directory.

MD "*directory*"

Check the value of @ERROR to see if **MD** was successful (@ERROR = 0).

PASSWORD

No function; supported only for compatibility with KiXtart 2.3x.

PASSWORD "*password*"

PLAY

Plays ‘music’ on the computer's speaker, by using the SPK file format described below, or on a sound card by playing a WAV file.

PLAY [**FILE** "*path\filename*.spk"] | "*string*" | "*path\filename*.wav"

There are four possible syntax forms:

- **PLAY FILE** "Kbond.spk"
- **PLAY** "0g256t 0g8d247f 4d165f 247f 8d262f 4d165f 262f 8d277f 4d165f"
- **PLAY FILE** "Ding.wav"
- **PLAY** "Chimes.wav"

The string or file consists of a sequence of commands indicating the frequency and duration of the tones to play. The following commands are available:

- **F** or **f** - *frequency*
  This command causes a tone to be produced at the current frequency. The initial current frequency is 1000Hz. To change the value, indicate the desired frequency immediately followed by the f character. For example, to produce a tone at 1500Hz, specify **1500F**.
- **G** or **g** - *gap*
  This command sets the number of timer ticks (1 second = 18 ticks) of silence between individual tones. The number of timer ticks between tones is specified as a number immediately followed by **G**. The initial value is 0.
- **D** or **d** - *duration*
  This command sets the length (in timer ticks) of each tone. For example, to make each tone last about a third of a second, use the command **6d**.
- **T** or **t** - *tempo*
  This command scales the duration of each tone. This allows you to change the duration of a series of tones globally, without having to change each of the individual duration commands.

- A tempo value of **256** indicates normal tempo. A value of **4df** lasts:
- 2 timer ticks, when the tempo is set to 128
- 4 timer ticks, when the tempo is set to 256
- 8 timer ticks, when the tempo is set to 512

KiXtart automatically selects the appropriate action based on the file name extension you provide.

```
PLAY"0g256t 0g8d247f 4d165f 247f 8d262f 4d165f 262f 8d277f 4d165f
    277f   8d262f   4d165f 262f 8d247f 4d165f 247f 8d262f 4d165f
    262f   8d277f   4d165f 277f 8d262f"
```

QUIT

Exits KiXtart.

QUIT [*error level* / *exit code*]

If **QUIT** is followed by a numeric expression, then the value of that expression is used as the exit code of KiXtart, and you can check it using a batch file.

RD

Removes the directory specified.

RD "*directory*"

Check the value of @ERROR to see if **RD** was successful.

RETURN

Causes script execution to continue at the statement following the last **CALL** or **GOSUB** statement.

RETURN

If **RETURN** is specified in the main script, KiXtart stops.

RUN

Runs a command.

RUN "*command*"

Command can be any 16-bit or 32-bit application. To run command interpreter commands, specify the correct command interpreter as part of the command. **RUN** does not wait for the program to complete. Script execution continues immediately. This behavior is different from the MS-DOS – based version of KiXtart, where the **RUN** command also terminates the script. If you want to emulate the MS-DOS – based version, you must add an **EXIT** command after the **RUN** command.

```
RUN @LDRIVE + "\UPDATE.EXE"
RUN "%COMSPEC% /e:1024 /c DIR C:"
```

SELECT CASE … ENDSELECT

A **SELECT** statement is an efficient way to write a series of **IF ELSE** statements.

SELECT
CASE *expression*
statement1
....
CASE *expression*
statement2
....
ENDSELECT

A **SELECT** statement consists of one or more conditions (**CASE**) each of which is followed by one or more statements that are executed only if the condition evaluates to TRUE. The **SELECT** statement is processed from top to bottom. If an expression evaluates to TRUE, the statements immediately following it are executed, up to the next **CASE** statement.

Only one **CASE** statement is executed, regardless of how many statements evaluate to TRUE.

If *expression* does not contain any relational operators, the condition is considered to be true if it is numeric and if it evaluates to a value other than zero, or if it is alphanumeric and it evaluates to a string containing at least one character. **SELECT** statements can be nested as many times as memory allows.

```
SELECT
CASE InGroup("Domain Admins") AND @DAY = 1
    ? "Whatever…"
CASE InGroup("Office Users")
    ? "Etc…"
    ? "Etc…"
CASE 1      ; this is a nice way to provide a default CASE; if all other
; CASEs fail, this one will always be run
    ? "Hmm, you're not in one of our groups?"
ENDSELECT
```

SET

Sets environment variables in the environment of the current user (**HKEY_CURRENT_USER\Environment**).

SET "*variable=string*"

After any change to the environment, KiXtart informs running programs that the change was made, prompting them to regenerate their environments. Programs that support this feature (such as Program Manager, Task Manager, and Windows Explorer) update their environments when they receive the WM_SETTINGCHANGE message.

The environment of the current process (KiXtart 95) is not affected.
This command does not work on Windows . As an alternative, use the **SHELL** command to run Winset.exe. (Winset.exe is included on the Windows CD.)

SETL

Sets environment variables in the local environment that you see when you start a program from within a KiXtart script.

SETL "*variable=string*"

This command does not affect the current environment. If you start KiXtart from a batch file, any commands in the batch file that are run after KiXtart exits do not see changes made by the **SET** or **SETL** commands. If you want to run batch files or programs that depend on settings set by KiXtart, start them from KiXtart using **SHELL** or **RUN**.

SETL sets the value of @ERROR.

SETM

Sets environment variables in the environment of the local computer (**HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment**).

SETM "*variable=string*"

After any change to the environment, KiXtart informs running programs that the change was made, prompting them to regenerate their environments. Programs that support this feature (such as Program Manager, Task Manager, and Windows Explorer) update their environments when they receive the WM_SETTINGCHANGE message.

The environment of the current process (KiXtart 95) is not affected.

This command does not work on Windows . As an alternative, use the **SHELL** command to run Winset.exe. (Winset.exe is included on the Windows CD.)

SETTIME

Synchronizes the system clock of the local computer with the time on a specified source.

SETTIME "*source*"

Source can be one of the following:

A server name expressed in UNC format	KiXtart connects to the server specified to retrieve the time.
A domain name	KiXtart browses the domain for a server running the Time Source service.
"*"	KiXtart browses the local domain for any server running the Time Source service.

On Windows NT, SETTIME requires the current user to have the 'Change the system time' privilege.
For more information on running the Windows NT Time Source Service, see *Knowledge Base* article Q131715 (also available on TechNet).

```
SETTIME "*"
SETTIME "\\MYTIME"
SETTIME "TIMEDOMAIN"
```

SHELL

Loads and runs a program.

SHELL "*command*"

Command can be any 16-bit or 32-bit application. To run command interpreter commands, specify the correct command interpreter as part of the command.
Script execution is stopped until the program exits.
If the program you want to run needs to set environment variables (as is the case with Smsls.bat, for example), you may need to specify additional environment space by using the **/E** parameter.

SHELL sets the value of @ERROR to the exit code of the program that is run.

```
SHELL @LDRIVE + "\UPDATE.EXE"
SHELL "%COMSPEC% /e:1024 /c DIR C:"
SHELL "SETW USERNAME=@USERID"
```

```
SHELL "CMD.EXE /C COPY " + @LDRIVE + "\FILE.TXT C:\"
SHELL "%COMSPEC% /C COPY Z:\FILE.TXT C:\"
SHELL "C:\WINNT\SYSTEM32\CMD /E:1024 /C " + @LDRIVE + "\SMSLS.BAT"
```

SLEEP

Halts script execution for the number of seconds specified.

SLEEP <*seconds*>

SMALL

Changes the character mode to small (normal) characters.

SMALL

After using **SMALL**, subsequent screen output is normal. For more information, see **BIG** earlier in this section.

USE

Lists the current connections. Can also be used to connect a device, such as a drive or a printer, to a network resource; or to disconnect a device from a network resource.

USE LIST
USE <* | "*device*" | "*resource*"> /**DELETE** [/**PERSISTENT**]
USE ["*device*"] <"*resource*"> [/**USER**:*user*] [/**PASSWORD**:*password*] [/**PERSISTENT**]

Use **USE "*" /DELETE** to delete all current connections except those to a NETLOGON share and those to the drive or share from which KiXtart was started.

If a resource name contains non-alphanumeric characters (such as - or +), enclose the name in quotation marks.

On Windows 2000 only, the /**USER** and /**PASSWORD** parameters enable overriding the security context of the current user.

Check the value of @ERROR to see if **USE** was successful (a value of 0 indicates success).

```
USE E:    "\\SERVER\PUBLIC" /PERSISTENT
USE * /DELETE
USE E:    "\\SERVER\PUBLIC" /user:Yogi  /password:Bear
USE E:    "\\SERVER\PUBLIC"
```

```
USE LPT1: "\\SERVER\LASER"   /user:testlan\USER1
USE L: /DEL
USE LIST
USE H: @HOMESHR ; connect to user's home share
IF @ERROR = 0
    H:          ;
    CD @HOMEDIR ; change directory to user's home directory
ENDIF
```

WHILE - LOOP

Runs a set of statements as long as an expression is true.

WHILE "*expression*" ... **LOOP**

WHILE loops can be nested as many times as memory allows.

KiXtart Function Reference

Most functions take one or more string or numeric expressions as parameters. String parameters are indicated by double quotation marks around the parameter name. Certain functions allow for optional parameters. If you omit these parameters, the function uses a default value instead.

Return Values

Most functions return either a string or a numeric value, and can thus be used anywhere an expression is expected. Most functions also set the value of @ERROR, which allows you to check whether the function was successful.

Registry Functions

All registry functions use the following format to specify registry subkeys:
[*remote_computer_name*\][*Key*\]*Subkey*

Remote_computer_name can be any valid computer name in UNC format (preceded by two backslashes). If you do not specify a *remote_computer_name*, the program defaults to the local registry.

Key can be any of the four main registry trees: **HKEY_LOCAL_MACHINE**, **HKEY_USERS**, **HKEY_CLASSES_ROOT**, or **HKEY_CURRENT_USER**. If you do not specify a root key, KiXtart uses **HKEY_CURRENT_USER**.

Subkey can be any valid registry subkey. If the name of a subkey contains spaces, enclose the entire expression in quotation marks.

The following examples show correct syntax for registry functions:

```
\\VLEERBEER\HKEY_LOCAL_MACHINE\CONTROL
"HKEY_CURRENT_USER\Program Groups\Games"
"Control Panel\International\Sorting Order"
```

When gaining access to a remote registry, you can only specify either **HKEY_LOCAL_MACHINE** or **HKEY_USERS**. Also, if you want to gain access to a remote registry from Windows , you must enable remote registry access. For more information, see the instructions in the Admin\Nettools\ Remotreg directory on the Windows CD.

Caution:
KiXtart does not ask for confirmation when registry values are overwritten or when subkeys are deleted. Always be very careful when changing the registry, and preferably back up your system before changing registry values.

ADDKEY

Adds the specified subkey to the registry.

ADDKEY ("*subkey*")
Parameter
Subkey
A string that specifies the name of the subkey you want to add to the registry.

0	Subkey added
Error code	Function failed

```
$ReturnCode = AddKey("HKEY_CURRENT_USER\EZReg")
If $ReturnCode = 0
   ? "Key added...."
Endif
```

ADDPRINTERCONNECTION

Adds a connection to the specified printer for the current user.

ADDPRINTERCONNECTION ("*printer name*")

Printer name
The name of the printer to which to connect.

This function is available only on Windows 2000 and Windows NT, and can be used only to connect to printers on a server running under Windows 2000 and Windows NT.

When Windows 2000 connects to the printer, it may copy printer driver files to the local computer. If the user does not have permission to copy files to the appropriate location, **ADDPRINTERCONNECTION** fails, and @ERROR returns ERROR_ACCESS_DENIED.

0	Printer connection established
Error code	Function failed

```
If ADDPRINTERCONNECTION ("\\vleerbeer\hp laserjet 4") = 0
   ? "Added printer connection...."
Endif
```

ADDPROGRAMGROUP

Instructs Program Manager to create a new program group.

ADDPROGRAMGROUP ("*group name*", *common group flag*)

Group name
Identifies the group window to be added.

Common group flag
Optional numeric parameter. This parameter is available only on Windows 2000 and Windows NT. *Common group flag* can have the following values:

0	Creates a personal group.
1	Creates a common group. The current user must have administrative privileges, or the function fails.

0	Program group added
Error code	Function failed

```
If AddProgramGroup("NewGroup", 0) = 0
   ? "NewGroup has created...."
Endif
```

ADDPROGRAMITEM

Instructs Program Manager to add an icon to the active program group.

ADDPROGRAMITEM ("*command line*", "*name*", "*icon path*", *icon index*, "*default directory*", *minimize*, *replace*, *run in own space*)

Command line
Specifies the command line required to run the application. This parameter is a string containing the name of the executable file for the application. It can also include the path of the application and any required parameters.

Name
Specifies the title that is displayed below the icon in the group window.

Icon path
Identifies the file name for the icon to display in the group window. This string identifies a Windows-based executable file or an icon file. If no *icon path* is specified, Program Manager uses the first icon in the file specified by *command line* if that file is an executable file.

If *command line* specifies a file that has been associated with a program, Program Manager uses the first icon provided in the executable file of that program.

Association information is obtained from the registry. If *command line* specifies neither an executable file nor an associated program, Program Manager uses a default icon.

Icon index
This parameter is an integer that specifies the index of the icon in the file identified by the *icon path* parameter. Program Manager includes five default icons that can be used for programs not written for Windows.

Default directory
Specifies the name of the default (or working) directory. This parameter is a string.

Minimize
Optional numeric parameter. Specifies whether an application window is minimized when first displayed. Possible values for this parameter are:

0	Default system setting
1	Minimize

Replace
Optional numeric parameter. Specifies whether **ADDPROGRAMITEM** replaces an existing program item with the same name. Possible values for this parameter are:

0	Adds a new program item without replacing the existing one. This is the default.
1	Replaces any existing program item.

Run in own space
Optional numeric parameter. Specifies whether the program runs in its own address space. This parameter applies only to 16-bit Windows applications running on Windows NT. This parameter can have the following values:

0	Does not run in separate address space. This is the default.
1	Runs in separate address space.

There is a limit of 50 items that can be added to each program group.

0	Program item added
Error code	Function failed

```
If AddProgramItem("c:\windows\regedit.exe","RegEdit","",0,"c:\",0,0) = 0
   ? "Added program item 'RegEdit' to current group..."
Endif
```

ASC

Returns the ASCII code of the character specified.

ASC (*character*)

Character
Character you want to know the ASCII code of.

Numeric value representing the ASCII code of the character.

```
$ASCII = Asc( "H" )
```

AT

Places the cursor in the position indicated.

AT (*row*, *column*)
Row
Specifies the row at which to position the cursor.

Column
Specifies the column at which to position the cursor.

The cursor position is expressed in screen coordinates. A value of 0,0 represents the top left corner of the screen.

The **AT** command is ignored if all output has been redirected to a file using the **REDIRECTOUTPUT** function.

Nothing.

BOX

Draws a box.

BOX (*top_left_row*, *top_left_column*, *bottom_right_row*, *bottom_right_column*, "*line style*")

Top_left_row, *top_left_column*, *bottom_right_row*, *bottom_right_column*
The four corners of the box to be drawn, expressed in screen coordinates. A value of 0,0 represents the top left corner of the screen.

Line style

Possible values for *line style* are:

single	Single line outline, space as filler

double	Double line, space as filler
full	Full line, space as filler
grid	Single line, cross as filler

You can also create a custom box by using a string value for *line style*. The string can contain as many as 9 characters, which are defined as follows.

This character in the string	**Represents this portion of the box**
1st	Top-left corner
2nd	Top horizontal
3rd	Top -right corner
4th	Right vertical
5th	Bottom -right corner
6th	Bottom horizontal
7th	Bottom -left corner
8th	Left vertical
9th	Filler

The **BOX** command is ignored if all output is redirected to a file using the **REDIRECTOUTPUT** function.

Nothing.

```
BOX (10, 10, 12, 15, "+-+|+-+| ")   ;
```

produces the following box:

```
+---+
|   |
+---+
```

CHR

Insert special characters, such as carriage returns, in a string.

CHR (*character code*)

Character code
A numeric expression representing the character code to insert.

The string representation of the character code.

```
$Message  = "Hello " + @USERID + chr(13) + chr(10) + "Welcome to our
network."
```

CLOSE

Closes a file previously opened by the **OPEN** function.

CLOSE (*file number*)

File number
A numeric expression indicating the file number of the file to close. Possible values range from 1 to 10.

-2 Not valid file number specified

0 File closed

```
IF Close(3)
    Beep
    ? "Error closing file!"
ENDIF
```

COMPAREFILETIMES

Compares the date and time of two files.

COMPAREFILETIMES ("*file1*", "*file2*")

File1
Identifies the first file you want to compare.

File2
Identifies the second file you want to compare.

-3 *File2* could not be opened (see @ERROR for more information).

-2 *File1* could not be opened (see @ERROR for more information).

-1 *File1* is older than *file2*.

0 *File1* and *file2* have the same date and time.

1 *File1* is more recent than *file2*.

```
$Result = CompareFileTimes(@LDRIVE + "\USER.INI", "C:\WINDOWS\USER.INI")
IF $Result = 1 OR $Result = -3
    COPY @LDRIVE + "\USER.INI" "C:\WINDOWS\USER.INI"
ENDIF
```

DECTOHEX

Returns the hexadecimal representation of a decimal value.

DECTOHEX (Decimal value)

Decimal value
The value you want to have the hexadecimal representation of.

A string representing the hexadecimal value of the input value.

```
$Result = DexToHex(123)
```

DELKEY

Deletes the specified subkey from the registry.

DELKEY ("*subkey*")

Subkey
A string that specifies the name of the subkey you want to delete.

This call fails if any subkeys exist within the specified subkey. Use **DELTREE** if you want to delete a subkey that contains subkeys.

0	Subkey deleted
Error code	Function failed

```
$ReturnCode = DelKey("HKEY_CURRENT_USER\EZReg")
If $ReturnCode = 0
   ? "Key deleted...."
Endif
```

DELPRINTERCONNECTION

Deletes a connection to a printer that was established by using **ADDPRINTERCONNECTION**.

DELPRINTERCONNECTION ("*printer name*")

Printer name
A string that specifies the name of the printer connection to delete.

.

The **DELPRINTERCONNECTION** function does not delete any printer driver files that were copied from the server on which the printer resides when the printer connection was established.

0	Printer connection deleted
Error code	Function failed

```
If DelPrinterConnection ("hplaser4") = 0
   ? "Deleted printer connection...."
Endif
```

DELPROGRAMGROUP

Instructs Program Manager to delete an existing program group.

DELPROGRAMGROUP ("*group name*", *common group flag*)

Group name
Identifies the group to be deleted.

Common group flag
Optional numeric parameter. This parameter is available only on Windows 2000 and Windows NT. *Common group flag* can have the following values:

0	Deletes a personal group.
1	Deletes a common group. The current user must have administrative privileges, otherwise the function fails.

When this function runs, no confirmation is asked nor warning given.

0	Program group deleted
Error code	Function failed

```
If DelProgramGroup("NewGroup", 0) = 0
   ? "NewGroup deleted...."
Endif
```

DELPROGRAMITEM

Instructs Program Manager to delete an item from the active program group.

DELPROGRAMITEM ("*item name*")

Item name
Specifies the item to be deleted from the active program group.

0 Program item deleted

Error code Function failed

```
If DelProgramItem("Whatever") = 0
   ? "ProgramItem 'Whatever' deleted from the current group...."
Endif
```

DELTREE

Deletes a subkey from the registry, including all the subkeys contained in the specified subkey.

DELTREE ("*subkey*")

Subkey
Specifies the subkey to be deleted from the registry.

When this function runs, no confirmation is asked nor warning given.

0 Subkey deleted

Error code Function failed

```
$ReturnCode = DelTree("HKEY_CURRENT_USER\EZReg")
If $ReturnCode = 0
   ? "Key deleted...."
Endif
```

DELVALUE

Deletes a value entry from the registry.

DELVALUE ("*subkey*", "*entry*")

Subkey
A string that specifies the name of the subkey from which you want to delete an entry.

Entry
A string that specifies the name of the entry you want to delete.

0 Value entry deleted

Error code Function failed

```
$ReturnCode  =DelValue("HKEY_CURRENT_USER\EZReg", "Test")
If $ReturnCode = 0
   ? "Value deleted...."
Endif
```

DIR

Dir can be used to enumerate the files in a directory. Dir returns a string representing the name of a file, directory, or folder that matches a specified pattern. To retrieve subsequent entries in a directory, specify an empty string (“”) as the path.

DIR ("path", index)

Path

Optional string that specifies a file name — may include directory or folder, and drive. If path is empty (“”), Dir will return the next file of the previously opened enumeration handle. Wildcard characters (‘*’ and ‘?’) are supported.

Index

Optional number indicating which enumeration handle to use. The Dir function can enumerate two directories at the same time. To open the second enumeration handle, specify 1 for the index.

Returns a string representing the name of a file, directory, or folder that matches a specified pattern. An empty string ("") is returned if *path* is not found or to indicate that the end of the current enumeration was reached. Dir also sets the value of @ERROR :

0	Dir successful.
Error code	Function failed.

```
$FileName = Dir("C:\TEMP")
While $FileName <> "" and @ERROR = 0
    ? $FileName
    Dir()   ; retrieve next file
Loop
```

ENUMGROUP

Enumerates the global groups of which the current user is a member.

ENUMGROUP (*Index*)

Index

A numeric value representing the group whose name you want to discover (where 0 is the first subkey).

String	Global group name
Error code	Function failed

```
$Index = 0
DO
```

```
    $Group = ENUMGROUP($Index)
UNTIL Len($Group) = 0
```

ENUMKEY

Lists the names of the subkeys contained in a registry key or subkey.

ENUMKEY ("*subkey*", *index*)

Subkey

Specifies the key or subkey for which you want to enumerate the subkeys.

Index

A numeric value representing the position of the subkey whose name you want to discover. Zero (0) represents the first subkey in the key.

0	Function returns a string representing the subkey in the specified key
Error code	Function failed
259	Subkey does not exist

```
$Index = 0
:Loop1
$KeyName = ENUMKEY("HKEY_CURRENT_USER\Console\ ", $Index)
If @ERROR = 0
    ? "Name found: $KeyName"
    $Index = $Index + 1
    goto Loop1
Endif
```

ENUMLOCALGROUP

Enumerates the local groups of which the current user is a member.

ENUMLOCALGROUP (*index, "source"*)

Index

A numeric value representing the group whose name you want to discover (where 0 is the first subkey).

Source

Optional string value representing the server or domain whose local groups you want to query.

String	Local group name
Error code	Function failed

```
$Index = 0
DO
    $Group = ENUMLOCALGROUP($Index)
UNTIL Len($Group) = 0
```

- Or -

```
$Index = 0
DO
    $Group = ENUMLOCALGROUP($Index, "\\MyServer")
UNTIL Len($Group) = 0
```

ENUMVALUE

Lists the names of the registry entries contained in a specific key or subkey.

ENUMVALUE ("*subkey*", *index*)

Subkey

Specifies the key or subkey for which you want to enumerate the value entries.

Index

A numeric value representing the position of the entry whose name you want to discover. Zero (0) represents the first entry in the subkey.

0	Function returns a string representing the entry in the specified key or subkey
Error code	Function failed
259	Entry does not exist

```
$Index = 0
:Loop1
$ValueName = ENUMVALUE("HKEY_CURRENT_USER\Console\Configuration",
$Index)
If @ERROR = 0
    ? "Name found: $ValueName"
    $Index = $Index + 1
    goto Loop1
Endif
```

EXECUTE

Executes a piece of KiXtart script code.

EXECUTE (script code)

Script code

A string expression representing the code to execute.

The exitcode of the executed script.

```
Execute( '? "This is a demo of the Execute() function"' )

Execute( '$$X = 10' )    ; note the extra '$'

Execute( '$$X = ' + @USERID )
```

EXIST

Checks for the existence of one or more files.

EXIST ("*file name*")

File name
Identifies the file(s) you want to locate.

Supports wildcards.

0	File not found
1	File found

```
IF EXIST (@LDRIVE + "\users.txt")
    DISPLAY @LDRIVE + "\users.txt"
ENDIF
IF EXIST (@LDRIVE + "\*.INI")
    ; Etc, etc.
ENDIF
```

EXISTKEY

Checks for the existence of a registry subkey.

EXISTKEY ("*subkey*")

Subkey
Identifies the subkey you want to locate.

The values returned by **EXISTKEY** have the opposite meaning of the values returned by **EXIST**.

0	Subkey found
Error code	Subkey not found

```
$ReturnCode = ExistKey("HKEY_CURRENT_USER\Console\Configuration")
If $ReturnCode = 0
   ? "Key exists...."
Endif
```

GETDISKSPACE

Returns the number of kilobytes (KB) available to the current user on a specific drive.

GETDISKSPACE ("*drive*")

Drive

String that specifies a directory on the disk of interest. On Windows 2000 and Windows NT and on Windows 95 OSR2 and later versions, this string can be a UNC name. If this parameter is a UNC name, you must follow it with an additional backslash. For example, you would specify \\MyServer\MyShare as \\MyServer\MyShare\.

If Drive is an empty string, GetDiskSpace obtains information about the disk that contains the current directory.

On Windows 2000 and Windows NT and on Windows 95 OSR2 and later versions, *Drive* does not have to specify the root directory on a disk. On these platforms, the function accepts any directory on a disk.

A number representing the number of kilobytes (KB) available to the current user on the drive specified.
On Windows 95 OSR1 and earlier versions, the function can only return correct values for volumes that are smaller than 2 gigabytes in size. On Windows 2000 and Windows NT and Windows 95 OSR2 and later versions, the function always returns correct values, regardless of the size of the volume.

```
$Result = GetDiskSpace( "C:\" )

$Result = GetDiskSpace( "X:\MARKETING" )
```

GETFILEATTR

Returns the attributes of a file.

GETFILEATTR ("*file name*")

File name

Identifies the file for which you want to retrieve the attributes.

Zero to indicate the function failed. If the function failed, check @ERROR for details on the error. Otherwise, the return value represents the attributes of the file. The attributes can be one or more of the following values:

1	Read only	The file or directory is read-only. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it.
2	Hidden	The file or directory is hidden. It is not included in an ordinary

		directory listing.
4	System	The file or directory is part of, or is used exclusively by, the operating system.
16	Directory	The file name identifies a directory.
32	Archive	The file or directory is an archive file or directory. Applications use this attribute to mark files for backup or removal.
64	Encrypted	The file or directory is encrypted. For a file, this means that all data streams are encrypted. For a directory, this means that encryption is the default for newly created files and subdirectories.
128	Normal	The file or directory has no other attributes set. This attribute is valid only if used alone.
256	Temporary	The file is being used for temporary storage. File systems attempt to keep all of the data in memory for quicker access rather than flushing the data back to mass storage. A temporary file should be deleted by the application as soon as it is no longer needed.
512	Sparse file	The file is a sparse file.
1024	Reparse point	The file has an associated reparse point.
2048	Compressed	The file or directory is compressed. For a file, this means that all of the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories.
4096	Offline	The data of the file is not immediately available. Indicates that the file data has been physically moved to offline storage.

```
$Result = GetFileAttr(@LDRIVE + "\Kix32.exe")
IF GetFileAttr( "C:\TEMP" ) & 16
   ? "C:\temp is a directory !"
ENDIF
```

GETFILESIZE

Returns the size of a file in bytes.

GETFILESIZE ("*file name*")

File name
Identifies the file for which you want to retrieve the size.

Size of the file in bytes.
The maximum size of files that GetFileSize can correctly report the size of is 2,147,483,647 bytes.

```
$Result = GetFileSize(@LDRIVE + "\Kix32.exe")
```

GETFILETIME

Returns the date and time information of a file.

GETFILETIME ("*file name*")

File name
Identifies the file for which you want to retrieve the date and time information.

A string representing the date and time of the file in the format “YYYY/MM/DD HH:MM:SS”.
The information returned represents the time the file was last written to.

```
$Result = GetFileTime(@LDRIVE + "\Kix32.exe")
```

GETFILEVERSION

Returns a version information string of a file.

GETFILEVERSION ("*file name*",”versionfield”)

File name
Identifies the file for which you want to get the version string.

Versionfield
Optional parameter identifying the specific version information field that should be retrieved. By default, the FileVersion field is returned. Possible values for this field are :

Comments	This field contains any additional information that should be displayed for diagnostic purposes.
CompanyName	This field identifies the company that produced the file. For example, "Microsoft Corporation."
FileDescription	This field describes the file in such a way that it can be presented to users. This string may be presented in a list box when the user is choosing files to install. For example, "Keyboard driver for AT-style keyboards" or "Microsoft Word for Windows".
FileVersion	This field member identifies the version of this file. For example, "3.00A" or "5.00.RC2".
InternalName	This field identifies the file's internal name, if one exists. For example, this string could contain the module name for a dynamic-link library (DLL), a virtual device name for a

	Windows virtual device, or a device name for an MS-DOS device driver.
Language	Full English name of the language of the file specified in the format defined by ISO Standard 639. (example : “0413Dutch (Standard)”).
LegalCopyright	This field describes all copyright notices, trademarks, and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, copyright dates, trademark numbers, and so on. In English, this string should be in the format "Copyright Microsoft Corp. 1990–1994".
LegalTrademarks	This field describes all trademarks and registered trademarks that apply to the file. This should include the full text of all notices, legal symbols, trademark numbers, and so on. In English, this string should be in the format "Windows is a registered trademark of Microsoft Corporation".
OriginalFilename	This field identifies the original name of the file, not including a path. This enables an application to determine whether a file has been renamed by a user. This name may not be MS-DOS 8.3-format if the file is specific to a non-FAT file system.
PrivateBuild	This field describes by whom, where, and why this private version of the file was built. For example, "Built by OSCAR on \OSCAR2".
ProductName	This field identifies the name of the product with which this file is distributed. For example, this string could be "Microsoft Windows".
ProductVersion	This field identifies the version of the product with which this file is distributed. For example, "3.00A" or "5.00.RC2".
SpecialBuild	This field describes how this version of the file differs from the normal version. For example, "Private build for Microsoft solving mouse problems on M250 and M250E computers".

A string representing the file version field.
The information returned by this function is the same as the version information displayed in Windows Explorer.
This function applies only to 32-bit Windows – based executable files.

```
$Result = GetFileVersion(@LDRIVE + "\Kix32.exe")

$Result = GetFileVersion(@LDRIVE + "\Kix32.exe", “ProductVersion” )
```

INGROUP

Checks whether the current user is a member of a group.

INGROUP ("*group name*")

Group name
Identifies the group in which to check the user's membership.

INGROUP can be used to check for groupmembership of groups that exist on the domain or server where the user is logged on, or to check for groupmembership of groups on a specific domain or server.

When checking for a local group, **INGROUP** identifies that the user is indirectly a member of the group by virtue of being a member of a global group which, in turn, is a member of the local group.

If you want to check for membership in a group on a specific domain or server, use the following format:

```
"OtherDomain\group"
```

– Or –

```
"\\SomeServer\group"
```

For Windows clients, **INGROUP** works on local groups only if the KiXtart RPC service is running.

0 The user is not a member of a group with this name.

1 The user is a member of a global group with this name.

2 The user is a member of a local group with this name.

```
IF INGROUP("Domain Users")
    DISPLAY "z:\users.txt"
ENDIF
IF INGROUP("Developers") = 2
    ? "Member of local group Developers"
ENDIF
IF INGROUP("\\" + @WKSTA + "\Developers") = 2
    ? "Member of local group Developers on local system"
ENDIF
```

INSTR

Scans a string for the presence of a second string.

INSTR ("*string1*", "*string2*")

String1
The string to search in.

String2
The string to search for.

? Offset of the first character of *string2* found in *string1*

0 *String2* not present in *string1*

```
$x = INSTR(@DOMAIN, "TEST")   ; check if domain contains the string
"TEST"
```

LCASE

Returns a string in lowercase.

LCASE ("*string*")

String
The string you want to change to lowercase.

The input string in lowercase.

```
$x = LCASE(@USERID)
```

LEN

Returns the length of a string.

LEN ("*string*")

String
The string whose length you want to discover.

The number of characters contained in the specified string.

```
$x = LEN(@USERID)
```

LOADHIVE

Creates a subkey under HKEY_USERS or HKEY_LOCAL_MACHINE and stores registration information from a specified file into that subkey. This registration information is in the form of a hive. A hive is a discrete body of keys, subkeys, and values that is rooted at the top of the registry hierarchy. A hive is backed by a single file and .LOG file.

LOADHIVE ("key", "*file name*")

Key
The key you want to load the information in. This key must reside under HKEY_LOCAL_MACHINE or HKEY_USERS.

File name
Identifies the file you want to load the information from. This file specified needs to be a legal registry hive (created by SAVEKEY, or from REGEDT32.EXE).

On Windows 2000 and Windows NT, using **LOADHIVE** requires Backup and Restore privileges.

0	Hive loaded
Error code	Function failed

```
$ReturnCode = LoadHive("HKEY_USERS\EZReg", "c:\temp\tst.reg")
If $ReturnCode = 0
   ? "Hive loaded...."
Endif
```

LOADKEY

Loads a registry key (including its subkeys and values) from a file.

LOADKEY ("*subkey*", "*file name*")

Subkey
The subkey in which you want to load the information. This subkey must exist for the call to be successful.

File name
Identifies the file from which to import the information. This file must be a valid registry hive file created by using the **SAVEKEY** function, or by using a registry editor.

On Windows NT, using **LOADKEY** requires Backup and Restore privileges. Caution: **LOADKEY** imports information into the registry and overwrites any existing subkey. This replaces all the subkeys and values that might already exist in the subkey you are loading. Any existing values and subkeys are lost.

0	Subkey loaded
Error code	Function failed

```
$ReturnCode = LoadKey("HKEY_CURRENT_USER\EZReg", "c:\temp\tst.reg")
If $ReturnCode = 0
   ? "Key loaded...."
Endif
```

LOGEVENT

Logs an event in the Windows 2000 or Windows NT event log.

LOGEVENT (type, ID, message, target, source)

Type

Number representing the type of the event. Possible values :

0	SUCCESS
1	ERROR
2	WARNING
4	INFORMATION
8	AUDIT_SUCCESS
16	AUDIT_FAILURE

ID

Number representing the event that occurred.

Message

Message text of the event.

Target

Optional parameter representing the UNC name of the system where the event should be logged. By default, all events are logged on the local system.

Source

Optional parameter representing the source of the event. If this parameter is not specified, Kixtart will assume the KIX32.EXE as the source of the event.

0	Event logged
Error code	Function failed

clients.

```
$RC = LogEvent( 0 , 1 , "Logon script completed successfully" , "",
"MyEvent" )
$RC = LogEvent( 0 , 1 , "Logon script completed successfully")
$RC = LogEvent( 1 , 1 , "Logon script failed!" , @LSERVER )
```

LOGOFF

Logs the current user off and ends the Windows session.

LOGOFF (*force*)

Force
During a logoff operation, applications that are shut down are allowed a specific amount of time to respond to the logoff request. If the time expires, Windows displays a dialog box that allows the user to forcibly shut down the application, to retry the logoff, or to cancel the logoff request. If the Force value is true (for example : non-zero), Windows always forces applications to close and does not display the dialog box.

0	Windows does not force applications to close.
1	Windows always forces applications to close and does not display the dialog box.

0	User logged off
Error code	Function failed

```
$RC = LogOff(0)
```

LTRIM

Strips leading spaces from an input string and returns the result.

LTRIM ("*string*")

String
The string from which to strip leading spaces.

The input string without leading spaces.

```
$x = LTRIM(SUBSTR(@IPADDRESS0, 1, 3));   192
```

MESSAGEBOX

Displays a standard dialog box in Windows.

MESSAGEBOX ("*message*", "*title*", *style*, *time-out*)

Message
The message to display in the dialog box.

Title
The title of the dialog box.

Style
Optional numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality. The following table illustrates the values used and the meaning of each group of values.

Buttons to display

Value	Meaning
0	Display **OK** button only.
1	Display **OK** and **Cancel** buttons.
2	Display **Abort**, **Retry**, and **Ignore** buttons.
3	Display **Yes**, **No**, and **Cancel** buttons.
4	Display **Yes** and **No** buttons.
5	Display **Retry** and **Cancel** buttons.

Icon to display

Value	Meaning
16	Stop symbol
32	Question mark
48	Exclamation mark
64	Information symbol

Default button

Value	Meaning
0	First button is default.
256	Second button is default.
512	Third button is default.

Modality

Value	Meaning
0	Application-modal. The user must respond to the message box before continuing work in the application.
4096	System-modal. All applications are suspended until the user responds to the message box.

When adding numbers to create a final value for the argument type, use only one number from each group. If *style* is omitted, a default value of 0 is assumed.

Time-out

Optional numeric expression representing the number of seconds after which to close the dialog box.

The time-out feature only works if the **MESSAGEBOX** dialog box is the active window for the duration of the time-out. If the user switches away from KiXtart and activates another application, the **MESSAGEBOX** dialog box is not closed.

MESSAGEBOX displays a maximum of 1024 characters in application-modal dialog boxes. Longer messages are truncated after the 1024th character. Message strings longer than 255 characters with no intervening spaces are truncated after the 255th character. For system-modal dialog boxes, the number of characters you can display depends on screen resolution and number of lines in the message.

MESSAGEBOX breaks lines automatically at the right edge of the dialog box. If you want to set line breaks yourself, place a linefeed (ANSI character 10) before the first character of the text that is to begin each new line.

The value returned by **MESSAGEBOX** indicates which button was selected, as shown in the following table.

Value	Meaning
-1	User did not respond to the dialog box within the specified time-out period.
1	**OK** button selected.
2	**Cancel** button selected.
3	**Abort** button selected.
4	**Retry** button selected.
5	**Ignore** button selected.
6	**Yes** button selected.
7	**No** button selected.

If the dialog box contains a **Cancel** button, pressing ESC has the same effect as choosing **Cancel**.

```
$Selection = MessageBox("Do you want to continue ?", "KiXtart", 36)
If $Selection = 6
   ? "Yes selected, continuing...."
Endif
```

OptionalArgument1
Optional string value representing an argument for the method. Note : all optional arguments must be specified as a string. The actual type of the argument is determined automatically based on the corresponding TypeCharacter in TypeList.

Returns

A string representing the return value of the function. If the call fails, @ERROR will be set to the relevant error code.

OLECALLFUNC

Accesses a method of an OLE Automation object that returns a value.

OLECALLFUNC (*objecthandle*, "*methodname*", *“typelist”, “optionalargument1”, “optionalargument2” , ...*)

ObjectHandle
The handle of the object you want to access. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

Methodname
The name of the method you want to access.

TypeList
TypeList is a **casesensitive** series of characters that define the type of each optional argument. Based on the type specified, KiXtart will convert the argument(s) to the correct type before calling the OLE function. This parameter can have the following values:

b	Boolean
c	Currency
D	Date
i	Short integer
I	Long integer
o	Object handle
r	4 byte real
R	8 byte real

Optionalargument
Optional string value representing an argument for the method. Note : all optional arguments must be specified as a string. The actual type of the argument is determined automatically by the corresponding TypeCharacter in TypeList.

A string representing the return value of the function. If the call fails, @ERROR will be set to the relevant error code.

OLECALLPROC

Accesses a method of an OLE Automation object that does not return a value.

OLECALLPROC (*objecthandle*, "*methodname*", *“typelist”, “optionalargument1”, “optionalargument2” , ...*)

ObjectHandle
The handle of the object you want to access. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

Methodname
The name of the method you want to access.

TypeList
TypeList is a **casesensitive** series of characters that define the type of each optional argument. Based on the type specified, KiXtart will convert the argument(s) to the correct type before calling the OLE function. This parameter can have the following values:

b	Boolean
c	Currency
D	Date
i	Short integer
I	Long integer
o	Object handle
r	4 byte real
R	8 byte real

Optionalargument
Optional string value representing an argument for the method. Note : all optional arguments must be specified as a string. The actual type of the argument is determined automatically by the corresponding TypeCharacter in TypeList.

If the function succeeds, the return value is 0. If the function fails, the return value represents an error code.

OLECREATEOBJECT

OLECreateObject launches (if necessary) the OLE Automation server and returns a handle through which the OLE Automation object can be manipulated.

OLECREATEOBJECT (“serverclassname")

ServerClassName
The handle of the object you want to create.

If the function succeeds it returns the handle to the object. If the function fails, it returns 0.

OLEGETOBJECT

OLEGetObject gets an object either from a file stored on disk, or from a class name, and returns a handle to the object.

OLEGETOBJECT (mode, “objectname" , “classname”)

Mode

0	ObjectName specifies a file on disk
1	ObjectName specifies a classname
2	ObjectName specifies a file on disk, and ClassName specifies the class of the object

ObjectName
The filename or the classname of the object you want to create.

ClassName
Optional parameter (only required if Mode = 2) indicating the class of the object to get.

If the function succeeds it returns the handle to the object. If the function fails, it returns 0.

OLEGETPROPERTY

Returns the value of a specific property of an OLE Automation object.

OLEGETPROPERTY (objecthandle , “propertyname”)

ObjectHandle
The handle of the object you want to access. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

PropertyName
The name of the property you want to retrieve the value of.

If the function succeeds, the return value is the value of the property. If the function fails, the return value will be empty, and @ERROR will be set to the relevant error code.

OLEGETSUBOBJECT

Some methods return handles to sub-objects. OLEGetSubObject is a way to retrieve these handles.

OLEGETSUBOBJECT (*objecthandle*, "*methodname*", *“typelist”, “optionalargument1”, “optionalargument2” , ...*)

ObjectHandle
The handle of the object you want to access. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

Methodname
The name of the method you want to access.

TypeList
TypeList is a **casesensitive** series of characters that define the type of each optional argument. Based on the type specified, KiXtart will convert the argument(s) to the correct type before calling the OLE function. This parameter can have the following values:

b	Boolean
c	Currency
D	Date
i	Short integer
I	Long integer
o	Object handle
r	4 byte real
R	8 byte real

Optionalargument
Optional string value representing an argument for the method. Note : all optional arguments must be specified as a string. The actual type of the argument is determined automatically by the corresponding TypeCharacter in TypeList.

If the function succeeds, the return value is 0. If the function fails, the return value represents an error code.

OLEPUTPROPERTY

Sets the value of a specific property of an OLE Automation object.

OLEPUTPROPERTY (objecthandle , “propertyname”, “value”)

ObjectHandle
The handle of the object you want to access. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

PropertyName
The name of the property you want to retrieve the value of.

Value
String representing the value that the property should be set to.

If the function succeeds, the return value is 0. If the function fails, the return value represents an error code.

OLERELEASEOBJECT

OLEReleaseObject frees up any resources obtained when the OLE object was created. If you do not explicitly release an object, all its resources will remain allocated until KiXtart exits.

OLERELEASEOBJECT (objecthandle)

ObjectHandle
The handle of the object you want to release. This handle must have been obtained by a call to OLECreateObject, OLEGetObject, OLEGetSubObject, OLEGetProperty or OLECallFunc.

If the function succeeds, the return value is 0. If the function fails, the return value represents an error code.

OPEN

Opens a text file.

OPEN (*file number*, "*file name*", *mode*)

File number
A numeric expression indicating the file number of the file to open. Possible values range from 1 to 10.

File name
A string expression indicating the path and name of the ASCII file to open.

Mode
Optional parameter that indicates what should happen if the file does not exist. This parameter can have the following values:

0	If the file does not exist, OPEN fails with return code 2 (default).
1	If the file does not exist, OPEN will create a new file.
2	Opens the file for read access (default).
4	Opens the file for write access.

These values are cumulative. So if you want to open a file for write access, and create it if it does not yet exist, you should specify 5. Notice however that a file can not be opened for read and write access at the same time.

OPEN opens the ASCII file specified by *file name*, for the internal buffer indicated by *file number*. KiXtart supports a maximum of ten open files, so *file number* must be within the range of 1 to 10.

The file-I/O functions in KiXtart (**OPEN**, **READLINE**, and **CLOSE**) process small configuration files. They are not intended for larger operations, such as scanning long files. For the sake of simplicity and speed, **OPEN** reads an entire ASCII file into memory, and subsequent **READLINE** commands read lines stored in memory.

Although this design is memory-intensive, it is also very fast and simple.

-3	File number already in use
-2	Incorrect file number specified
-1	Incorrect file name specified
0	File opened successfully
>0	System error

```
IF Open(3, @LDRIVE + "\CONFIG\SETTINGS.INI")  = 0
$x = ReadLine(3)
WHILE @ERROR = 0
   ? "Line read: [" + $x + "]"
   $x = ReadLine(3)
LOOP
ENDIF
```

READLINE

Reads a line from a file.

READLINE (*file number*)

File number

A numeric expression indicating the file number of the file to open. Possible values range from 1 to 10.

READLINE reads a string ending in a carriage return. If successful, the function returns the string without a carriage return. If it encounters an error, @ERROR returns an error code.

-4	File not open for reading
-3	File number not open
-2	Incorrect file number specified

-1	End of file
0	Line read successfully

```
IF Open(3, @LDRIVE + "\CONFIG\SETTINGS.INI")  = 0
$x = ReadLine(3)
WHILE @ERROR = 0
? "Line read: [" + $x + "]"
$x = ReadLine(3)
LOOP
Close (3)
ELSE
BEEP
? "Config file not opened, error code: [" + @ERROR + "]"
ENDIF
```

READPROFILESTRING

Retrieves a string from an initialization file.

READPROFILESTRING ("*file name*", "*section*", "*key*")

File name
A string that names the initialization file. If this parameter does not include a full path, Windows searches for the file in the Windows directory.

Section
A string that specifies the section containing the key name. If this parameter is empty, **READPROFILESTRING** returns all section names in the file.

Key
A string containing the key name whose associated string is to be retrieved. If this parameter is empty, all key names in the section specified by *section* are returned.

This function is provided for compatibility with 16-bit Windows – based applications. Win32 – based applications store initialization information in the registry.

0	Function returns a string representing the value of the specified key
Error code	Function failed

```
$dev = ReadProfileString("win.ini", "Windows", "Device")
If @ERROR = 0
   ? "Windows device = " + $dev
Endif
```

READTYPE

Returns the ASCII representation of a registry entry data type (for example, **REG_SZ**).

READTYPE ("*subkey*", "*entry*")

Subkey
Identifies the subkey containing the entry.

Entry
Identifies the entry whose data type you want to discover.

0	Function returns ASCII representation of data type for specified registry entry
Error code	Function failed

The following data types can be returned:

- **REG_NONE**
- **REG_SZ**
- **REG_EXPAND_SZ**
- **REG_BINARY**
- **REG_DWORD**
- **REG_DWORD_LITTLE_ENDIAN**
- **REG_DWORD_BIG_ENDIAN**
- **REG_LINK**
- **REG_MULTI_SZ**
- **REG_RESOURCE_LIST**
- **REG_FULL_RESOURCE_DESCRIPTOR**

```
$RowsType = ReadType("HKEY_CURRENT_USER\Console\Configuration",
"WindowRows")
If @ERROR = 0
   ? "Type of  WindowRows: $RowsType"
Endif
```

READVALUE

Reads the value of a registry entry.

READVALUE ("*subkey*", "*entry*")
Subkey
Identifies the subkey containing the entry.

Entry
Identifies the entry whose value you want to discover. To read the default entry of a key, specify an empty string as the entry name (“”).

0	Function returns ASCII representation of the specified registry entry
Error code	Function failed

REG_MULTI_SZ (multi-string) variables are returned with the pipe symbol (|) used as the separator between strings. If a string contains a pipe symbol character, it is represented by two pipe symbol characters (||).

REG_DWORD variables are returned in decimal format.

```
$Rows = ReadValue("HKEY_CURRENT_USER\Console\Configuration",
"WindowRows")
If @ERROR = 0
   ? "Number of window-rows: $Rows"
Endif
```

REDIRECTOUTPUT

Redirects all screen output to a file.

REDIRECTOUTPUT ("*file name*", *overwrite*)

File name
A string that names the file to which to redirect the output. If this parameter is an empty string (""), output is redirected to the screen.

Overwrite
Optional numeric value indicating whether to clear the output file before writing any data to it. This parameter can have the following values:

0	New output data appended to the existing contents of file.
1	All data in file overwritten when the output is redirected to the file.

If all output is redirected to a file, the **AT**, **BIG**, **BOX**, and **CLS** commands are ignored.

0	Output redirected
Error code	Function failed

```
IF RedirectOutput("logon.log") = 0
    ? "Opened 'logon.log' at " + @TIME ?
ENDIF
```

RND

Returns a pseudo random number.

RND (Range)

Range
Optional parameter indicating the range for the return value (maximum and default value = 32767).

Pseudo random number.
The RND function returns a pseudo-random integer ranging from 0 to the maximum specified. Use the SRND function to seed the pseudo-random-number generator before calling RND.

```
$x = RND()
$x = RND(10)
```

RTRIM

Strips trailing spaces from an input string and returns the result.

RTRIM ("*string*")

String
The string from which to strip trailing spaces.

The input string without trailing spaces.

```
$x = RTRIM(SUBSTR(@IPADDRESS0, 1, 3));   192
```

SAVEKEY

Saves a registry key (including its subkeys and entries) to a file.

SAVEKEY ("*subkey*", "*file name*")

Subkey
Identifies the subkey you want to save.

File name
Identifies the file in which to save the information.

When this function runs, the destination file is overwritten without warning.
On Windows 2000 and Windows NT, running **SAVEKEY** requires Backup and Restore privileges.

0	Subkey saved

Error code Function failed

```
$ReturnCode = SaveKey("HKEY_CURRENT_USER\EZReg", "c:\temp\tst.reg")
If $ReturnCode = 0
   ? "Key saved...."
Endif
```

SENDKEYS

Sends one or more keystrokes to the active window as if typed at the keyboard.

SENDKEYS ("*keys*")

Keys

String specifying the keystrokes to send.

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use `"A"` for *string*. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use `"ABC"` for *string*. The plus sign (+), caret (^),tilde (~), and parentheses ( ) have special meanings to SendKeys. To specify one of these characters, enclose it within braces (`{}`). For example, to specify the plus sign, use `{+}`. To specify brace characters, use `{{}` and `{}}`.

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes shown below:

BACKSPACE	{BACKSPACE}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL	{DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER}
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS	{INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}

PAGE UP	{PGUP}
PRINTSCREEN	{PRTSC}
RIGHT ARROW	{RIGHT}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

SHIFT	+
CTRL	^
ALT	~

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form `{key number}`. You must put a space between `key` and `number`. For example, `{LEFT 42}` means press the LEFT ARROW key 42 times; `{h 10}` means press H 10 times.

SendKeys cannot be used to send keystrokes to an application that is not designed to run in Microsoft Windows. Sendkeys also can't send the PRINT SCREEN key {PRTSC} to any application.

0	Keystrokes sent
Error code	Function failed

```
Shell( "Notepad.exe" )
SetFocus( "Untitled - Notepad" )
$ReturnCode = SendKeys("Hello World")
Sleep( 2 )
$ReturnCode = SendKeys("~{F4}Y")
```

SENDMESSAGE

Sends a message across the network to another user or workstation.

SENDMESSAGE ("*recipient*", "*message*")

Recipient
Identifies the user or workstation to which to send the message.

Message
The message to send.

0	Message sent
Error code	Function failed

```
$ReturnCode = SendMessage("ADMIN" , @USERID + " logged in at " + @TIME)
If $ReturnCode = 0
   ? "Message has been sent.."
Endif
```

SETCONSOLE

Changes the display state of the command-prompt window in which KiXtart is running.

SETCONSOLE("*mode*")

Mode
String that specifies the new display state. The following table shows the display states that are supported by this function.

SHOW	Show window
HIDE	Hide window

FOREGROUND	Move window to foreground
ALWAYSONTOP	Bring window to top
MINIMIZE	Minimize window
MAXIMIZE	Maximize window

If a window is hidden, it does not disappear from the system, but remains active.

0	Display state changed
Error code	Function failed

```
If SetConsole ("FOREGROUND") = 0
   ? "Console moved to foreground......"
Endif
```

SETDEFAULTPRINTER

Sets the default printer to which applications send print jobs.

SETDEFAULTPRINTER ("*printer name*")

Printer name
String that specifies the name of the printer to set as the default printer.

0	Default printer set
Error code	Function failed

```
If SetDefaultPrinter ("hplaser4") = 0
   ? "Set default printer to HP LaserJet 4...."
Endif
```

SETFILEATTR

Sets the attributes of a file.

SETFILEATTR ("*file name*", attributes)

File name
Identifies the file of which you want to set the attributes.

Attributes
Attributes to set for the file. The attributes can be one or more of the following values:

1	Read only	The file or directory is read-only. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it.

2	Hidden	The file or directory is hidden. It is not included in an ordinary directory listing.
4	System	The file or directory is part of, or is used exclusively by, the operating system.
32	Archive	The file or directory is an archive file or directory. Applications use this attribute to mark files for backup or removal.
128	Normal	The file or directory has no other attributes set. This attribute is valid only if used alone.
256	Temporary	The file is being used for temporary storage. File systems attempt to keep all of the data in memory for quicker access rather than flushing the data back to mass storage. A temporary file should be deleted by the application as soon as it is no longer needed.
4096	Offline	The data of the file is not immediately available. Indicates that the file data has been physically moved to offline storage.

0	Attributes set
Error code	Function failed

```
$Result = SetFileAttr(@LDRIVE + "\Kix32.exe", 32)
```

SETFOCUS

Sets the input focus to the application specified. This function is very useful in combination with the SendKeys function.

SETFOCUS ("*Title*")

Title
String specifying the title in the title bar of the application window you want to activate. In determining which application to activate, *title* is compared to the title string of each running application. If there is no exact match, any application whose title string begins with *title* is activated. If there is more than one instance of the application named by *title*, one instance is arbitrarily activated.

0	Focus set to specified application.
Error code	Function failed

```
If SetFocus ("Untitled - Notepad") = 0
   ? "Focus set to Notepad...."
Endif
```

SETWALLPAPER

Sets the current wallpaper.

SETWALLPAPER("*wallpaper name*")

Wallpaper name
String that specifies the name of the bitmap to set as the default wallpaper.

The file specified must be a valid BMP file.

0	Wallpaper set
Error code	Function failed

```
If SetWallpaper ("kixtart.bmp") = 0
   ? "Set current wallpaper to KiXtart.bmp...."
Endif
```

SHOWPROGRAMGROUP

Instructs Program Manager to minimize, maximize, or restore the window of an existing program group.

SHOWPROGRAMGROUP ("*group name*", *show command*, *common group flag*)

Group name
Identifies the group window to minimize, maximize, or restore.

Show command
Specifies the action Program Manager is to perform on the group window. This parameter is an integer and it must have one of the following values.

Value	Action
1	Activates and displays the group window. If the window is minimized or maximized, Windows restores it to its original size and position.
2	Activates the group window and displays it as an icon.
3	Activates the group window and displays it as a maximized window.
4	Displays the group window in its most recent size and position. The active window remains active.
5	Activates the group window and displays it in its current size and position.

6 Minimizes the group window.

7 Displays the group window as an icon. The active window remains active.

8 Displays the group window in its current state. The active window remains active.

Common group flag

Optional numeric parameter. This parameter is available only on Windows 2000 and Windows NT. *Common group flag* can have the following values:

0 Acts upon a personal group.

1 Acts upon a common group. To manipulate a common group, the user must have administrative privileges, or the function fails.

0	Program group maximized, minimized, or restored
Error code	Function failed

```
If ShowProgramGroup("NewGroup", 6, 0) = 0
   ? "NewGroup has been minimized...."
Endif
```

SHUTDOWN

Shuts down or restarts a computer.

SHUTDOWN ("*computer*", "*message*", *wait*, *force*, *reboot*)

Computer

The name of the computer that is to be shut down or restarted. An empty string("") indicates the local computer.

Message

String that specifies a message to display in the **Shutdown** dialog box.

Wait

Optional parameter specifying the time in seconds that the dialog box is displayed. While the dialog box is displayed, system shutdown can be stopped by using the Win32 **AbortSystemShutdown** function.

If *wait* is not zero, **SHUTDOWN** displays a dialog box on the specified computer. The dialog box, which displays the name of the user who called the function and the message specified by *message*, prompts the user to log off. The system beeps when the dialog box is created.

The dialog box remains on top of other windows and can be moved but not closed. A timer counts down the time remaining before a forced shutdown. If the

user logs off, the system shuts down immediately. Otherwise, the computer is shut down when the timer expires.

If *wait* is zero, the computer shuts down without displaying the dialog box, and the shutdown cannot be stopped by **AbortSystemShutdown**.

Force

Specifies whether applications with unsaved changes are forcibly closed. If *force* is not zero, applications are closed. If *force* is zero, a dialog box is displayed prompting the user to close the applications.

Reboot

Optional parameter specifying whether the computer is to restart immediately after shutting down. If *reboot* is 1, the computer restarts. If *reboot* is 0, the computer does not start.

0	Computer shut down
System error code	Function failed

SHUTDOWN does not work reliably on Windows due to an issue in the underlying Windows API. As a workaround, try the following command :

```
SHELL "%windir%\RUNDLL32.EXE user.exe,ExitWindows"
```

```
$RC = Shutdown("", "System is being rebooted to enable new settings.",
60, 0, 1)
```

SRND

The SRND function sets the starting point for generating a series of pseudo-random integers. To reinitialize the generator, use 1 as the seed argument. Any other value for seed sets the generator to a random starting point. RND retrieves the pseudo-random numbers that are generated. Calling RND before any call to SRND generates the same sequence as calling SRND with seed passed as 1.

SRND (seed)

Seed

Numeric value to seed the generator with.

Nothing.

```
SRND( 10 )
```

SUBSTR

Returns part of a string.

SUBSTR ("*string*", *start*, *length*)

String
The string from which to extract a substring.

Start
Numeric value representing the offset in the string where the substring begins.

Length
Numeric value representing the length of the substring.

The substring indicated by *start* and *length*.

```
$x = SUBSTR(@USERID, LEN(@USERID) - 2, 2)   ; get the last 2 chars of
the userid
```

UCASE

Returns a string in uppercase.

UCASE ("*string*")

String
The string you want to change to uppercase.

The input string in uppercase.

```
$x = UCASE(@USERID)
```

UNLOADHIVE

Unloads the specified key and subkeys from the registry.

UNLOADHIVE ("key")

Key
The key you want to unload. This key must have been created using LoadHive.

On Windows NT, using **UNLOADHIVE** requires Backup and Restore privileges.

0	Key loaded
Error code	Function failed

```
$ReturnCode = UnLoadHive( "HKEY_USERS\Fiets" )

If $ReturnCode = 0
  ? "Hive unloaded...."
Endif
```

VAL

Returns the numeric value of a string.
VAL ("*string*")
String
The string whose numeric value you want to discover. By default, Val expects the string to be in decimal format. To determine the numeric value of a hexadecimal string, start the string with an ampersand '&'.

The numeric value of the input string.

```
$x = VAL(SUBSTR(@IPADDRESS0, 1, 3))
$x = VAL("&A34")
```

WRITELINE

Appends a line to the end of the file indicated by FileNumber. If WriteLine encounters an error, @ERROR is set to the relevant errorcode.

WRITELINE (*file number, "linetowrite"*)

File number
A numeric expression indicating the file number of the file to open. Possible values range from 1 to 10.

LineToWrite
The string you want to write to the file.

WriteLine does not automatically append a <Carriage Return>, so if you want to write a <Carriage Return>, you should add it to the string (as in : $LineToWrite + Chr(13) + Chr(10)).

-4	File not open for writing
-3	File number not open
-2	Incorrect file number specified
-1	End of file
0	Line written successfully

```
IF Open( 3 , "C:\TEMP\LOG.TXT" , 5 )  = 0
$x = WriteLine( 3 , "KiXtart started at " +  @TIME + Chr(13) + Chr(10) )
ELSE
BEEP
? "failed to open file, error code : [" + @ERROR + "]"
```

```
ENDIF
```

WRITEPROFILESTRING

Copies a string to an initialization file.

WRITEPROFILESTRING ("*file name*", "*section*", "*key*", "*string*")

File name
String identifying the initialization file.

Section
String containing the name of the section of the initialization file where *string* is copied. If the section does not exist, it is created. The section name is not case-sensitive, and can contain any combination of uppercase and lowercase letters.

Key
String containing the name of the key to associate with *string*. If the key does not exist in the specified section, it is created. If this parameter is empty, the entire section, including all entries within the section, is deleted.

String
String to write to the file. If this parameter is empty, the key identified by *key* is deleted.

On Windows , use of the tab character (\t) is not supported as part of this parameter.

This function is provided for compatibility with 16-bit Windows-based applications. Win32-based applications store initialization information in the registry.

0	Profile string written
Error code	Function failed

WRITEVALUE

Assigns a value to a registry entry.

WRITEVALUE ("*subkey*", "*entry*", "*expression*", "*data type*")

Subkey
Identifies the subkey where you want to write a value entry.

Entry
The name of the entry. To write to the default entry of a key, specify an empty string as the entry name (“”).

Expression
The data to store as the value of the entry.

REG_MULTI_SZ (multi-string) variables are returned with the pipe symbol (|) used as the separator between strings. If a string contains a pipe symbol character, it is represented by two pipe symbol characters (||).

Data type
Identifies the data type of the entry.

The following data types are supported:

- **REG_NONE**
- **REG_SZ**
- **REG_EXPAND_SZ**
- **REG_BINARY**
- **REG_DWORD**
- **REG_DWORD_LITTLE_ENDIAN**
- **REG_DWORD_BIG_ENDIAN**
- **REG_LINK**
- **REG_MULTI_SZ**
- **REG_RESOURCE_LIST**
- **REG_FULL_RESOURCE_DESCRIPTOR**

0	Value entry written
Error code	Function failed

```
WriteValue("EZReg\Test", "A MultiString variable", "Line 1|Line 2|Line 3
with a || in it|" "REG_MULTI_SZ")
If @ERROR = 0
   ? "Value written to the registry"
Endif
```

KiXtart Macro Reference

Macros can be used anywhere an expression is expected. Supported macros are defined in the following table.

Macro	Definition
ADDRESS	Address of the network adapter
COMMENT	User comment
CURDIR	Current directory
DATE	Date (in the format YYYY/MM/DD)
DAY	Day of the week (Monday, Tuesday, and so on)
DOMAIN	Domain or workgroup the computer belongs to

DOS	Version of Windows 2000 or Windows NT
ERROR	Return code of the most recent command or function. A return code of 0 means the command or function was successful. Any other value indicates an error.
FULLNAME	Full name of current user
HOMEDIR	Short name of the directory part of home directory
HOMEDRIVE*	Drive letter of drive containing home directory
HOMESHR	Server and share name part of home directory
HOSTNAME	Fully qualified TCP/IP host name (including TCP/IP domain name)
INWIN	Operating system: 1 = Windows NT; 2 = Windows
IPADDRESS*x*	TCP/IP address (possible values for *x* are 0 - 3). **Note** Addresses are padded so that the resulting string always consists of four sets of three characters separated by periods. For example, if your IP address is 123.45.6.7, @IPADDRESS0 is 123. 45. 6. 7.
KIX	Version of KiXtart
LANROOT	Directory where network software resides (usually *Systemroot*\System32)
LDOMAIN*	Logon domain
LDRIVE	Drive that is redirected to \\logonserver\NETLOGON
LM	Version of network software
LONGHOMEDIR	Long name of the directory part of home directory
LSERVER	Logon server
MAXPWAGE	Maximum password age
MDAYNO	Day of the month (1-31)
MONTHNO	Months since January (1-12)
MONTH	Name of the month
PRIMARYGROUP*	Current user's primary group
PRIV	User's privilege level (GUEST, USER, ADMIN)
PWAGE	Password age
RAS	Number of active Remote Access Service (RAS) connections
RSERVER*	KXRPC server used for the current session
SCRIPTDIR	Directory of current script
SERROR	Error text corresponding with @ERROR
SID*	Current user's security identifier (SID)
SITE**	Name of the site in which the system resides
STARTDIR	Directory from which KiXtart was started
SYSLANG	Full English name of the language of the operating system

	specified in the format defined by ISO Standard 639. (example : “0413Dutch (Standard)”).
TIME	Current time (in the format HH:MM:SS)
USERID	Current user's user ID
USERLANG	Full English name of the language selected by the current user specified in the format defined by ISO Standard 639. (example : “0413Dutch (Standard)”).
WDAYNO	Days since Sunday (1 – 7)
WKSTA	Computer name
WUSERID	Current user's Windows user ID
YDAYNO	Days since January 1 (1 – 365)
YEAR	Current year

*Available on computers running Windows only if the KiXtart RPC service is running.
** Only available on clients with full Active Directory support.

During the logon sequence, WUSERID is empty on computers running Windows if Windows NT Networking has been configured as the system's primary network provider.

The following examples show the correct use of KiXtart macros:

```
@LM                       "2.10"
@DATE                     "1997/10/03"
DISPLAY @USERID + ".TXT" displays the file "RUUDV.TXT"
CD "\DATA\" + @DOMAIN     changes the current directory to "\DATA\your-
                          domain"
```

KiXtart Error Codes

To find out if a KiXtart command or function is successful, always check the @ERROR and @SERROR macros. Most functions also return the error code. If @ERROR is zero, the previous command or function was successful. If @ERROR is non-zero, the value corresponds to the error code returned by the most recently executed Win32 API. The following list contains all the possible error codes returned by the Win32 API. Only a limited number of error codes are actually relevant to KiXtart.

Code	Error
0L	NO_ERROR, ERROR_SUCCESS
1L	ERROR_INVALID_FUNCTION
2L	ERROR_FILE_NOT_FOUND
3L	ERROR_PATH_NOT_FOUND

4L	ERROR_TOO_MANY_OPEN_FILES
5L	ERROR_ACCESS_DENIED
6L	ERROR_INVALID_HANDLE
7L	ERROR_ARENA_TRASHED
8L	ERROR_NOT_ENOUGH_MEMORY
9L	ERROR_INVALID_BLOCK
10L	ERROR_BAD_ENVIRONMENT
11L	ERROR_BAD_FORMAT
12L	ERROR_INVALID_ACCESS
13L	ERROR_INVALID_DATA
14L	ERROR_OUTOFMEMORY
15L	ERROR_INVALID_DRIVE
16L	ERROR_CURRENT_DIRECTORY
17L	ERROR_NOT_SAME_DEVICE
18L	ERROR_NO_MORE_FILES
19L	ERROR_WRITE_PROTECT
20L	ERROR_BAD_UNIT
21L	ERROR_NOT_READY
22L	ERROR_BAD_COMMAND
23L	ERROR_CRC
24L	ERROR_BAD_LENGTH
25L	ERROR_SEEK
26L	ERROR_NOT_DOS_DISK
27L	ERROR_SECTOR_NOT_FOUND
28L	ERROR_OUT_OF_PAPER
29L	ERROR_WRITE_FAULT
30L	ERROR_READ_FAULT
31L	ERROR_GEN_FAILURE
32L	ERROR_SHARING_VIOLATION
33L	ERROR_LOCK_VIOLATION
34L	ERROR_WRONG_DISK
36L	ERROR_SHARING_BUFFER_EXCEEDED
38L	ERROR_HANDLE_EOF
39L	ERROR_HANDLE_DISK_FULL
50L	ERROR_NOT_SUPPORTED
51L	ERROR_REM_NOT_LIST

52L	ERROR_DUP_NAME
53L	ERROR_BAD_NETPATH
54L	ERROR_NETWORK_BUSY
55L	ERROR_DEV_NOT_EXIST
56L	ERROR_TOO_MANY_CMDS
57L	ERROR_ADAP_HDW_ERR
58L	ERROR_BAD_NET_RESP
59L	ERROR_UNEXP_NET_ERR
60L	ERROR_BAD_REM_ADAP
61L	ERROR_PRINTQ_FULL
62L	ERROR_NO_SPOOL_SPACE
63L	ERROR_PRINT_CANCELLED
64L	ERROR_NETNAME_DELETED
65L	ERROR_NETWORK_ACCESS_DENIED
66L	ERROR_BAD_DEV_TYPE
67L	ERROR_BAD_NET_NAME
68L	ERROR_TOO_MANY_NAMES
69L	ERROR_TOO_MANY_SESS
70L	ERROR_SHARING_PAUSED
71L	ERROR_REQ_NOT_ACCEP
72L	ERROR_REDIR_PAUSED
80L	ERROR_FILE_EXISTS
82L	ERROR_CANNOT_MAKE
83L	ERROR_FAIL_I24
84L	ERROR_OUT_OF_STRUCTURES
85L	ERROR_ALREADY_ASSIGNED
86L	ERROR_INVALID_PASSWORD
87L	ERROR_INVALID_PARAMETER
88L	ERROR_NET_WRITE_FAULT
89L	ERROR_NO_PROC_SLOTS
100L	ERROR_TOO_MANY_SEMAPHORES
101L	ERROR_EXCL_SEM_ALREADY_OWNED
102L	ERROR_SEM_IS_SET
103L	ERROR_TOO_MANY_SEM_REQUESTS
104L	ERROR_INVALID_AT_INTERRUPT_TIME
105L	ERROR_SEM_OWNER_DIED

106L	ERROR_SEM_USER_LIMIT
107L	ERROR_DISK_CHANGE
108L	ERROR_DRIVE_LOCKED
109L	ERROR_BROKEN_PIPE
110L	ERROR_OPEN_FAILED
111L	ERROR_BUFFER_OVERFLOW
112L	ERROR_DISK_FULL
113L	ERROR_NO_MORE_SEARCH_HANDLES
114L	ERROR_INVALID_TARGET_HANDLE
117L	ERROR_INVALID_CATEGORY
118L	ERROR_INVALID_VERIFY_SWITCH
119L	ERROR_BAD_DRIVER_LEVEL
120L	ERROR_CALL_NOT_IMPLEMENTED
121L	ERROR_SEM_TIMEOUT
122L	ERROR_INSUFFICIENT_BUFFER
123L	ERROR_INVALID_NAME
124L	ERROR_INVALID_LEVEL
125L	ERROR_NO_VOLUME_LABEL
126L	ERROR_MOD_NOT_FOUND
127L	ERROR_PROC_NOT_FOUND
128L	ERROR_WAIT_NO_CHILDREN
129L	ERROR_CHILD_NOT_COMPLETE
130L	ERROR_DIRECT_ACCESS_HANDLE
131L	ERROR_NEGATIVE_SEEK
132L	ERROR_SEEK_ON_DEVICE
133L	ERROR_IS_JOIN_TARGET
134L	ERROR_IS_JOINED
135L	ERROR_IS_SUBSTED
136L	ERROR_NOT_JOINED
137L	ERROR_NOT_SUBSTED
138L	ERROR_JOIN_TO_JOIN
139L	ERROR_SUBST_TO_SUBST
140L	ERROR_JOIN_TO_SUBST
141L	ERROR_SUBST_TO_JOIN
142L	ERROR_BUSY_DRIVE
143L	ERROR_SAME_DRIVE

144L	ERROR_DIR_NOT_ROOT
145L	ERROR_DIR_NOT_EMPTY
146L	ERROR_IS_SUBST_PATH
147L	ERROR_IS_JOIN_PATH
148L	ERROR_PATH_BUSY
149L	ERROR_IS_SUBST_TARGET
150L	ERROR_SYSTEM_TRACE
151L	ERROR_INVALID_EVENT_COUNT
152L	ERROR_TOO_MANY_MUXWAITERS
153L	ERROR_INVALID_LIST_FORMAT
154L	ERROR_LABEL_TOO_LONG
155L	ERROR_TOO_MANY_TCBS
156L	ERROR_SIGNAL_REFUSED
157L	ERROR_DISCARDED
158L	ERROR_NOT_LOCKED
159L	ERROR_BAD_THREADID_ADDR
160L	ERROR_BAD_ARGUMENTS
161L	ERROR_BAD_PATHNAME
162L	ERROR_SIGNAL_PENDING
164L	ERROR_MAX_THRDS_REACHED
167L	ERROR_LOCK_FAILED
170L	ERROR_BUSY
173L	ERROR_CANCEL_VIOLATION
174L	ERROR_ATOMIC_LOCKS_NOT_SUPPORTED
180L	ERROR_INVALID_SEGMENT_NUMBER
182L	ERROR_INVALID_ORDINAL
183L	ERROR_ALREADY_EXISTS
186L	ERROR_INVALID_FLAG_NUMBER
187L	ERROR_SEM_NOT_FOUND
188L	ERROR_INVALID_STARTING_CODESEG
189L	ERROR_INVALID_STACKSEG
190L	ERROR_INVALID_MODULETYPE
191L	ERROR_INVALID_EXE_SIGNATURE
192L	ERROR_EXE_MARKED_INVALID
193L	ERROR_BAD_EXE_FORMAT
194L	ERROR_ITERATED_DATA_EXCEEDS_64k

195L	ERROR_INVALID_MINALLOCSIZE
196L	ERROR_DYNLINK_FROM_INVALID_RING
197L	ERROR_IOPL_NOT_ENABLED
198L	ERROR_INVALID_SEGDPL
199L	ERROR_AUTODATASEG_EXCEEDS_64k
200L	ERROR_RING2SEG_MUST_BE_MOVABLE
201L	ERROR_RELOC_CHAIN_XEEDS_SEGLIM
202L	ERROR_INFLOOP_IN_RELOC_CHAIN
203L	ERROR_ENVVAR_NOT_FOUND
205L	ERROR_NO_SIGNAL_SENT
206L	ERROR_FILENAME_EXCED_RANGE
207L	ERROR_RING2_STACK_IN_USE
208L	ERROR_META_EXPANSION_TOO_LONG
209L	ERROR_INVALID_SIGNAL_NUMBER
210L	ERROR_THREAD_1_INACTIVE
212L	ERROR_LOCKED
214L	ERROR_TOO_MANY_MODULES
215L	ERROR_NESTING_NOT_ALLOWED
230L	ERROR_BAD_PIPE
231L	ERROR_PIPE_BUSY
232L	ERROR_NO_DATA
233L	ERROR_PIPE_NOT_CONNECTED
234L	ERROR_MORE_DATA
240L	ERROR_VC_DISCONNECTED
254L	ERROR_INVALID_EA_NAME
255L	ERROR_EA_LIST_INCONSISTENT
259L	ERROR_NO_MORE_ITEMS
266L	ERROR_CANNOT_COPY
267L	ERROR_DIRECTORY
275L	ERROR_EAS_DIDNT_FIT
276L	ERROR_EA_FILE_CORRUPT
277L	ERROR_EA_TABLE_FULL
278L	ERROR_INVALID_EA_HANDLE
282L	ERROR_EAS_NOT_SUPPORTED
288L	ERROR_NOT_OWNER
298L	ERROR_TOO_MANY_POSTS

317L	ERROR_MR_MID_NOT_FOUND
487L	ERROR_INVALID_ADDRESS
534L	ERROR_ARITHMETIC_OVERFLOW
535L	ERROR_PIPE_CONNECTED
536L	ERROR_PIPE_LISTENING
994L	ERROR_EA_ACCESS_DENIED
995L	ERROR_OPERATION_ABORTED
996L	ERROR_IO_INCOMPLETE
997L	ERROR_IO_PENDING
998L	ERROR_NOACCESS
999L	ERROR_SWAPERROR
1001L	ERROR_STACK_OVERFLOW
1002L	ERROR_INVALID_MESSAGE
1003L	ERROR_CAN_NOT_COMPLETE
1004L	ERROR_INVALID_FLAGS
1005L	ERROR_UNRECOGNIZED_VOLUME
1006L	ERROR_FILE_INVALID
1007L	ERROR_FULLSCREEN_MODE
1008L	ERROR_NO_TOKEN
1009L	ERROR_BADDB
1010L	ERROR_BADKEY
1011L	ERROR_CANTOPEN
1012L	ERROR_CANTREAD
1013L	ERROR_CANTWRITE
1014L	ERROR_REGISTRY_RECOVERED
1015L	ERROR_REGISTRY_CORRUPT
1016L	ERROR_REGISTRY_IO_FAILED
1017L	ERROR_NOT_REGISTRY_FILE
1018L	ERROR_KEY_DELETED
1019L	ERROR_NO_LOG_SPACE
1020L	ERROR_KEY_HAS_CHILDREN
1021L	ERROR_CHILD_MUST_BE_VOLATILE
1022L	ERROR_NOTIFY_ENUM_DIR
1051L	ERROR_DEPENDENT_SERVICES_RUNNING
1052L	ERROR_INVALID_SERVICE_CONTROL
1053L	ERROR_SERVICE_REQUEST_TIMEOUT

1054L	ERROR_SERVICE_NO_THREAD
1055L	ERROR_SERVICE_DATABASE_LOCKED
1056L	ERROR_SERVICE_ALREADY_RUNNING
1057L	ERROR_INVALID_SERVICE_ACCOUNT
1058L	ERROR_SERVICE_DISABLED
1059L	ERROR_CIRCULAR_DEPENDENCY
1060L	ERROR_SERVICE_DOES_NOT_EXIST
1061L	ERROR_SERVICE_CANNOT_ACCEPT_CTRL
1062L	ERROR_SERVICE_NOT_ACTIVE
1063L	ERROR_FAILED_SERVICE_CONTROLLER_CONNECT
1064L	ERROR_EXCEPTION_IN_SERVICE
1065L	ERROR_DATABASE_DOES_NOT_EXIST
1066L	ERROR_SERVICE_SPECIFIC_ERROR
1067L	ERROR_PROCESS_ABORTED
1068L	ERROR_SERVICE_DEPENDENCY_FAIL
1069L	ERROR_SERVICE_LOGON_FAILED
1070L	ERROR_SERVICE_START_HANG
1071L	ERROR_INVALID_SERVICE_LOCK
1072L	ERROR_SERVICE_MARKED_FOR_DELETE
1073L	ERROR_SERVICE_EXISTS
1074L	ERROR_ALREADY_RUNNING_LKG
1075L	ERROR_SERVICE_DEPENDENCY_DELETED
1076L	ERROR_BOOT_ALREADY_ACCEPTED
1077L	ERROR_SERVICE_NEVER_STARTED
1078L	ERROR_DUPLICATE_SERVICE_NAME
1100L	ERROR_END_OF_MEDIA
1101L	ERROR_FILEMARK_DETECTED
1102L	ERROR_BEGINNING_OF_MEDIA
1103L	ERROR_SETMARK_DETECTED
1104L	ERROR_NO_DATA_DETECTED
1105L	ERROR_PARTITION_FAILURE
1106L	ERROR_INVALID_BLOCK_LENGTH
1107L	ERROR_DEVICE_NOT_PARTITIONED
1108L	ERROR_UNABLE_TO_LOCK_MEDIA
1109L	ERROR_UNABLE_TO_UNLOAD_MEDIA
1110L	ERROR_MEDIA_CHANGED

1111L	ERROR_BUS_RESET
1112L	ERROR_NO_MEDIA_IN_DRIVE
1113L	ERROR_NO_UNICODE_TRANSLATION
1114L	ERROR_DLL_INIT_FAILED
1115L	ERROR_SHUTDOWN_IN_PROGRESS
1116L	ERROR_NO_SHUTDOWN_IN_PROGRESS
1117L	ERROR_IO_DEVICE
1118L	ERROR_SERIAL_NO_DEVICE
1119L	ERROR_IRQ_BUSY
1120L	ERROR_MORE_WRITES
1121L	ERROR_COUNTER_TIMEOUT
1122L	ERROR_FLOPPY_ID_MARK_NOT_FOUND
1123L	ERROR_FLOPPY_WRONG_CYLINDER
1124L	ERROR_FLOPPY_UNKNOWN_ERROR
1125L	ERROR_FLOPPY_BAD_REGISTERS
1126L	ERROR_DISK_RECALIBRATE_FAILED
1127L	ERROR_DISK_OPERATION_FAILED
1128L	ERROR_DISK_RESET_FAILED
1129L	ERROR_EOM_OVERFLOW
1130L	ERROR_NOT_ENOUGH_SERVER_MEMORY
1131L	ERROR_POSSIBLE_DEADLOCK
1132L	ERROR_MAPPED_ALIGNMENT
1140L	ERROR_SET_POWER_STATE_VETOED
1141L	ERROR_SET_POWER_STATE_FAILED
1150L	ERROR_OLD_WIN_VERSION
1151L	ERROR_APP_WRONG_OS
1152L	ERROR_SINGLE_INSTANCE_APP
1153L	ERROR_RMODE_APP
1154L	ERROR_INVALID_DLL
1155L	ERROR_NO_ASSOCIATION
1156L	ERROR_DDE_FAIL
1157L	ERROR_DLL_NOT_FOUND
1200L	ERROR_BAD_DEVICE
1201L	ERROR_CONNECTION_UNAVAIL
1202L	ERROR_DEVICE_ALREADY_REMEMBERED
1203L	ERROR_NO_NET_OR_BAD_PATH

1204L	ERROR_BAD_PROVIDER
1205L	ERROR_CANNOT_OPEN_PROFILE
1206L	ERROR_BAD_PROFILE
1207L	ERROR_NOT_CONTAINER
1208L	ERROR_EXTENDED_ERROR
1209L	ERROR_INVALID_GROUPNAME
1210L	ERROR_INVALID_COMPUTERNAME
1211L	ERROR_INVALID_EVENTNAME
1212L	ERROR_INVALID_DOMAINNAME
1213L	ERROR_INVALID_SERVICENAME
1214L	ERROR_INVALID_NETNAME
1215L	ERROR_INVALID_SHARENAME
1216L	ERROR_INVALID_PASSWORDNAME
1217L	ERROR_INVALID_MESSAGENAME
1218L	ERROR_INVALID_MESSAGEDEST
1219L	ERROR_SESSION_CREDENTIAL_CONFLICT
1220L	ERROR_REMOTE_SESSION_LIMIT_EXCEEDED
1221L	ERROR_DUP_DOMAINNAME
1247L	ERROR_ALREADY_INITIALIZED
1248L	ERROR_NO_MORE_DEVICES
1300L	ERROR_NOT_ALL_ASSIGNED
1301L	ERROR_SOME_NOT_MAPPED
1302L	ERROR_NO_QUOTAS_FOR_ACCOUNT
1303L	ERROR_LOCAL_USER_SESSION_KEY
1304L	ERROR_NULL_LM_PASSWORD
1305L	ERROR_UNKNOWN_REVISION
1306L	ERROR_REVISION_MISMATCH
1307L	ERROR_INVALID_OWNER
1308L	ERROR_INVALID_PRIMARY_GROUP
1309L	ERROR_NO_IMPERSONATION_TOKEN
1310L	ERROR_CANT_DISABLE_MANDATORY
1311L	ERROR_NO_LOGON_SERVERS
1312L	ERROR_NO_SUCH_LOGON_SESSION
1313L	ERROR_NO_SUCH_PRIVILEGE
1314L	ERROR_PRIVILEGE_NOT_HELD
1315L	ERROR_INVALID_ACCOUNT_NAME

1316L	ERROR_USER_EXISTS
1317L	ERROR_NO_SUCH_USER
1318L	ERROR_GROUP_EXISTS
1319L	ERROR_NO_SUCH_GROUP
1320L	ERROR_MEMBER_IN_GROUP
1321L	ERROR_MEMBER_NOT_IN_GROUP
1322L	ERROR_LAST_ADMIN
1323L	ERROR_WRONG_PASSWORD
1324L	ERROR_ILL_FORMED_PASSWORD
1325L	ERROR_PASSWORD_RESTRICTION
1326L	ERROR_LOGON_FAILURE
1327L	ERROR_ACCOUNT_RESTRICTION
1328L	ERROR_INVALID_LOGON_HOURS
1329L	ERROR_INVALID_WORKSTATION
1330L	ERROR_PASSWORD_EXPIRED
1331L	ERROR_ACCOUNT_DISABLED
1332L	ERROR_NONE_MAPPED
1333L	ERROR_TOO_MANY_LUIDS_REQUESTED
1334L	ERROR_LUIDS_EXHAUSTED
1335L	ERROR_INVALID_SUB_AUTHORITY
1336L	ERROR_INVALID_ACL
1337L	ERROR_INVALID_SID
1338L	ERROR_INVALID_SECURITY_DESCR
1340L	ERROR_BAD_INHERITANCE_ACL
1341L	ERROR_SERVER_DISABLED
1342L	ERROR_SERVER_NOT_DISABLED
1343L	ERROR_INVALID_ID_AUTHORITY
1344L	ERROR_ALLOTTED_SPACE_EXCEEDED
1345L	ERROR_INVALID_GROUP_ATTRIBUTES
1346L	ERROR_BAD_IMPERSONATION_LEVEL
1347L	ERROR_CANT_OPEN_ANONYMOUS
1348L	ERROR_BAD_VALIDATION_CLASS
1349L	ERROR_BAD_TOKEN_TYPE
1350L	ERROR_NO_SECURITY_ON_OBJECT
1351L	ERROR_CANT_ACCESS_DOMAIN_INFO
1352L	ERROR_INVALID_SERVER_STATE

1353L	ERROR_INVALID_DOMAIN_STATE
1354L	ERROR_INVALID_DOMAIN_ROLE
1355L	ERROR_NO_SUCH_DOMAIN
1356L	ERROR_DOMAIN_EXISTS
1357L	ERROR_DOMAIN_LIMIT_EXCEEDED
1358L	ERROR_INTERNAL_DB_CORRUPTION
1359L	ERROR_INTERNAL_ERROR
1360L	ERROR_GENERIC_NOT_MAPPED
1361L	ERROR_BAD_DESCRIPTOR_FORMAT
1362L	ERROR_NOT_LOGON_PROCESS
1363L	ERROR_LOGON_SESSION_EXISTS
1364L	ERROR_NO_SUCH_PACKAGE
1365L	ERROR_BAD_LOGON_SESSION_STATE
1366L	ERROR_LOGON_SESSION_COLLISION
1367L	ERROR_INVALID_LOGON_TYPE
1368L	ERROR_CANNOT_IMPERSONATE
1369L	ERROR_RXACT_INVALID_STATE
1370L	ERROR_RXACT_COMMIT_FAILURE
1371L	ERROR_SPECIAL_ACCOUNT
1372L	ERROR_SPECIAL_GROUP
1373L	ERROR_SPECIAL_USER
1374L	ERROR_MEMBERS_PRIMARY_GROUP
1375L	ERROR_TOKEN_ALREADY_IN_USE
1376L	ERROR_NO_SUCH_ALIAS
1377L	ERROR_MEMBER_NOT_IN_ALIAS
1378L	ERROR_MEMBER_IN_ALIAS
1379L	ERROR_ALIAS_EXISTS
1380L	ERROR_LOGON_NOT_GRANTED
1381L	ERROR_TOO_MANY_SECRETS
1382L	ERROR_SECRET_TOO_LONG
1383L	ERROR_INTERNAL_DB_ERROR
1384L	ERROR_TOO_MANY_CONTEXT_IDS
1385L	ERROR_LOGON_TYPE_NOT_GRANTED
1386L	ERROR_NT_CROSS_ENCRYPTION_REQUIRED
1387L	ERROR_NO_SUCH_MEMBER
1388L	ERROR_INVALID_MEMBER

1389L	ERROR_TOO_MANY_SIDS
1390L	ERROR_LM_CROSS_ENCRYPTION_REQUIRED
1391L	ERROR_NO_INHERITANCE
1392L	ERROR_FILE_CORRUPT
1393L	ERROR_DISK_CORRUPT
1394L	ERROR_NO_USER_SESSION_KEY
1395L	ERROR_LICENSE_QUOTA_EXCEEDED
1400L	ERROR_INVALID_WINDOW_HANDLE
1401L	ERROR_INVALID_MENU_HANDLE
1402L	ERROR_INVALID_CURSOR_HANDLE
1403L	ERROR_INVALID_ACCEL_HANDLE
1404L	ERROR_INVALID_HOOK_HANDLE
1405L	ERROR_INVALID_DWP_HANDLE
1406L	ERROR_TLW_WITH_WSCHILD
1407L	ERROR_CANNOT_FIND_WND_CLASS
1408L	ERROR_WINDOW_OF_OTHER_THREAD
1409L	ERROR_HOTKEY_ALREADY_REGISTERED
1410L	ERROR_CLASS_ALREADY_EXISTS
1411L	ERROR_CLASS_DOES_NOT_EXIST
1412L	ERROR_CLASS_HAS_WINDOWS
1413L	ERROR_INVALID_INDEX
1414L	ERROR_INVALID_ICON_HANDLE
1415L	ERROR_PRIVATE_DIALOG_INDEX
1416L	ERROR_LISTBOX_ID_NOT_FOUND
1417L	ERROR_NO_WILDCARD_CHARACTERS
1418L	ERROR_CLIPBOARD_NOT_OPEN
1419L	ERROR_HOTKEY_NOT_REGISTERED
1420L	ERROR_WINDOW_NOT_DIALOG
1421L	ERROR_CONTROL_ID_NOT_FOUND
1422L	ERROR_INVALID_COMBOBOX_MESSAGE
1423L	ERROR_WINDOW_NOT_COMBOBOX
1424L	ERROR_INVALID_EDIT_HEIGHT
1425L	ERROR_DC_NOT_FOUND
1426L	ERROR_INVALID_HOOK_FILTER
1427L	ERROR_INVALID_FILTER_PROC
1428L	ERROR_HOOK_NEEDS_HMOD

1429L	ERROR_GLOBAL_ONLY_HOOK
1430L	ERROR_JOURNAL_HOOK_SET
1431L	ERROR_HOOK_NOT_INSTALLED
1432L	ERROR_INVALID_LB_MESSAGE
1433L	ERROR_SETCOUNT_ON_BAD_LB
1434L	ERROR_LB_WITHOUT_TABSTOPS
1435L	ERROR_DESTROY_OBJECT_OF_OTHER_THREAD
1436L	ERROR_CHILD_WINDOW_MENU
1437L	ERROR_NO_SYSTEM_MENU
1438L	ERROR_INVALID_MSGBOX_STYLE
1439L	ERROR_INVALID_SPI_VALUE
1440L	ERROR_SCREEN_ALREADY_LOCKED
1441L	ERROR_HWNDS_HAVE_DIFFERENT_PARENT
1442L	ERROR_NOT_CHILD_WINDOW
1443L	ERROR_INVALID_GW_COMMAND
1444L	ERROR_INVALID_THREAD_ID
1445L	ERROR_NON_MDICHILD_WINDOW
1446L	ERROR_POPUP_ALREADY_ACTIVE
1447L	ERROR_NO_SCROLLBARS
1448L	ERROR_INVALID_SCROLLBAR_RANGE
1449L	ERROR_INVALID_SHOWWIN_COMMAND
1500L	ERROR_EVENTLOG_FILE_CORRUPT
1501L	ERROR_EVENTLOG_CANT_START
1502L	ERROR_LOG_FILE_FULL
1503L	ERROR_EVENTLOG_FILE_CHANGED
1700L	RPC_S_INVALID_STRING_BINDING
1701L	RPC_S_WRONG_KIND_OF_BINDING
1702L	RPC_S_INVALID_BINDING
1703L	RPC_S_PROTSEQ_NOT_SUPPORTED
1704L	RPC_S_INVALID_RPC_PROTSEQ
1705L	RPC_S_INVALID_STRING_UUID
1706L	RPC_S_INVALID_ENDPOINT_FORMAT
1707L	RPC_S_INVALID_NET_ADDR
1708L	RPC_S_NO_ENDPOINT_FOUND
1709L	RPC_S_INVALID_TIMEOUT
1710L	RPC_S_OBJECT_NOT_FOUND

1711L	RPC_S_ALREADY_REGISTERED
1712L	RPC_S_TYPE_ALREADY_REGISTERED
1713L	RPC_S_ALREADY_LISTENING
1714L	RPC_S_NO_PROTSEQS_REGISTERED
1715L	RPC_S_NOT_LISTENING
1716L	RPC_S_UNKNOWN_MGR_TYPE
1717L	RPC_S_UNKNOWN_IF
1718L	RPC_S_NO_BINDINGS
1719L	RPC_S_NO_PROTSEQS
1720L	RPC_S_CANT_CREATE_ENDPOINT
1721L	RPC_S_OUT_OF_RESOURCES
1722L	RPC_S_SERVER_UNAVAILABLE
1723L	RPC_S_SERVER_TOO_BUSY
1724L	RPC_S_INVALID_NETWORK_OPTIONS
1725L	RPC_S_NO_CALL_ACTIVE
1726L	RPC_S_CALL_FAILED
1727L	RPC_S_CALL_FAILED_DNE
1728L	RPC_S_PROTOCOL_ERROR
1730L	RPC_S_UNSUPPORTED_TRANS_SYN
1731L	RPC_S_SERVER_OUT_OF_MEMORY
1732L	RPC_S_UNSUPPORTED_TYPE
1733L	RPC_S_INVALID_TAG
1734L	RPC_S_INVALID_BOUND
1735L	RPC_S_NO_ENTRY_NAME
1736L	RPC_S_INVALID_NAME_SYNTAX
1737L	RPC_S_UNSUPPORTED_NAME_SYNTAX
1739L	RPC_S_UUID_NO_ADDRESS
1740L	RPC_S_DUPLICATE_ENDPOINT
1741L	RPC_S_UNKNOWN_AUTHN_TYPE
1742L	RPC_S_MAX_CALLS_TOO_SMALL
1743L	RPC_S_STRING_TOO_LONG
1744L	RPC_S_PROTSEQ_NOT_FOUND
1745L	RPC_S_PROCNUM_OUT_OF_RANGE
1746L	RPC_S_BINDING_HAS_NO_AUTH
1747L	RPC_S_UNKNOWN_AUTHN_SERVICE
1748L	RPC_S_UNKNOWN_AUTHN_LEVEL

1749L	RPC_S_INVALID_AUTH_IDENTITY
1750L	RPC_S_UNKNOWN_AUTHZ_SERVICE
1751L	EPT_S_INVALID_ENTRY
1752L	EPT_S_CANT_PERFORM_OP
1753L	EPT_S_NOT_REGISTERED
1755L	RPC_S_INCOMPLETE_NAME
1756L	RPC_S_INVALID_VERS_OPTION
1757L	RPC_S_NO_MORE_MEMBERS
1758L	RPC_S_NOT_ALL_OBJS_UNEXPORTED
1759L	RPC_S_INTERFACE_NOT_FOUND
1760L	RPC_S_ENTRY_ALREADY_EXISTS
1761L	RPC_S_ENTRY_NOT_FOUND
1762L	RPC_S_NAME_SERVICE_UNAVAILABLE
1764L	RPC_S_CANNOT_SUPPORT
1765L	RPC_S_NO_CONTEXT_AVAILABLE
1766L	RPC_S_INTERNAL_ERROR
1767L	RPC_S_ZERO_DIVIDE
1768L	RPC_S_ADDRESS_ERROR
1769L	RPC_S_FP_DIV_ZERO
1770L	RPC_S_FP_UNDERFLOW
1771L	RPC_S_FP_OVERFLOW
1772L	RPC_X_NO_MORE_ENTRIES
1773L	RPC_X_SS_CHAR_TRANS_OPEN_FAIL
1774L	RPC_X_SS_CHAR_TRANS_SHORT_FILE
1775L	RPC_X_SS_IN_NULL_CONTEXT
1776L	RPC_X_SS_CONTEXT_MISMATCH
1777L	RPC_X_SS_CONTEXT_DAMAGED
1778L	RPC_X_SS_HANDLES_MISMATCH
1779L	RPC_X_SS_CANNOT_GET_CALL_HANDLE
1780L	RPC_X_NULL_REF_POINTER
1781L	RPC_X_ENUM_VALUE_OUT_OF_RANGE
1782L	RPC_X_BYTE_COUNT_TOO_SMALL
1783L	RPC_X_BAD_STUB_DATA
1784L	ERROR_INVALID_USER_BUFFER
1785L	ERROR_UNRECOGNIZED_MEDIA
1786L	ERROR_NO_TRUST_LSA_SECRET

1787L	ERROR_NO_TRUST_SAM_ACCOUNT
1788L	ERROR_TRUSTED_DOMAIN_FAILURE
1789L	ERROR_TRUSTED_RELATIONSHIP_FAILURE
1790L	ERROR_TRUST_FAILURE
1791L	RPC_S_CALL_IN_PROGRESS
1792L	ERROR_NETLOGON_NOT_STARTED
1793L	ERROR_ACCOUNT_EXPIRED
1794L	ERROR_REDIRECTOR_HAS_OPEN_HANDLES
1795L	ERROR_PRINTER_DRIVER_ALREADY_INSTALLED
1796L	ERROR_UNKNOWN_PORT
1797L	ERROR_UNKNOWN_PRINTER_DRIVER
1798L	ERROR_UNKNOWN_PRINTPROCESSOR
1799L	ERROR_INVALID_SEPARATOR_FILE
1800L	ERROR_INVALID_PRIORITY
1801L	ERROR_INVALID_PRINTER_NAME
1802L	ERROR_PRINTER_ALREADY_EXISTS
1803L	ERROR_INVALID_PRINTER_COMMAND
1804L	ERROR_INVALID_DATATYPE
1805L	ERROR_INVALID_ENVIRONMENT
1806L	RPC_S_NO_MORE_BINDINGS
1807L	ERROR_NOLOGON_INTERDOMAIN_TRUST_ACCOUNT
1808L	ERROR_NOLOGON_WORKSTATION_TRUST_ACCOUNT
1809L	ERROR_NOLOGON_SERVER_TRUST_ACCOUNT
1810L	ERROR_DOMAIN_TRUST_INCONSISTENT
1811L	ERROR_SERVER_HAS_OPEN_HANDLES
1812L	ERROR_RESOURCE_DATA_NOT_FOUND
1813L	ERROR_RESOURCE_TYPE_NOT_FOUND
1814L	ERROR_RESOURCE_NAME_NOT_FOUND
1815L	ERROR_RESOURCE_LANG_NOT_FOUND
1816L	ERROR_NOT_ENOUGH_QUOTA
1817L	RPC_S_NO_INTERFACES
1818L	RPC_S_CALL_CANCELLED
1819L	RPC_S_BINDING_INCOMPLETE
1820L	RPC_S_COMM_FAILURE
1821L	RPC_S_UNSUPPORTED_AUTHN_LEVEL
1822L	RPC_S_NO_PRINC_NAME

1823L	RPC_S_NOT_RPC_ERROR
1824L	RPC_S_UUID_LOCAL_ONLY
1825L	RPC_S_SEC_PKG_ERROR
1826L	RPC_S_NOT_CANCELLED
1827L	RPC_X_INVALID_ES_ACTION
1828L	RPC_X_WRONG_ES_VERSION
1829L	RPC_X_WRONG_STUB_VERSION
1898L	RPC_S_GROUP_MEMBER_NOT_FOUND
1899L	EPT_S_CANT_CREATE
1900L	RPC_S_INVALID_OBJECT
1901L	ERROR_INVALID_TIME
1902L	ERROR_INVALID_FORM_NAME
1903L	ERROR_INVALID_FORM_SIZE
1904L	ERROR_ALREADY_WAITING
1905L	ERROR_PRINTER_DELETED
1906L	ERROR_INVALID_PRINTER_STATE
1907L	ERROR_PASSWORD_MUST_CHANGE
1908L	ERROR_DOMAIN_CONTROLLER_NOT_FOUND
1909L	ERROR_ACCOUNT_LOCKED_OUT
2000L	ERROR_INVALID_PIXEL_FORMAT
2001L	ERROR_BAD_DRIVER
2002L	ERROR_INVALID_WINDOW_STYLE
2003L	ERROR_METAFILE_NOT_SUPPORTED
2004L	ERROR_TRANSFORM_NOT_SUPPORTED
2005L	ERROR_CLIPPING_NOT_SUPPORTED
2138L	ERROR_NO_NETWORK
2202L	ERROR_BAD_USERNAME
2250L	ERROR_NOT_CONNECTED
2401L	ERROR_OPEN_FILES
2404L	ERROR_DEVICE_IN_USE
3000L	ERROR_UNKNOWN_PRINT_MONITOR
3001L	ERROR_PRINTER_DRIVER_IN_USE
3002L	ERROR_SPOOL_FILE_NOT_FOUND
3003L	ERROR_SPL_NO_STARTDOC
3004L	ERROR_SPL_NO_ADDJOB
3005L	ERROR_PRINT_PROCESSOR_ALREADY_INSTALLED

3006L	ERROR_PRINT_MONITOR_ALREADY_INSTALLED
4000L	ERROR_WINS_INTERNAL
4001L	ERROR_CAN_NOT_DEL_LOCAL_WINS
4002L	ERROR_STATIC_INIT
4003L	ERROR_INC_BACKUP
4004L	ERROR_FULL_BACKUP
4005L	ERROR_REC_NON_EXISTENT
4006L	ERROR_RPL_NOT_ALLOWED
6118L	ERROR_NO_BROWSER_SERVERS_FOUND

Feedback

For questions or feedback concerning this tool, please contact rkinput@microsoft.com.

If you are interested in joining an international community of KiXtart users to discuss tips and tricks and share sample scripts, please connect to http://www.brainbuzz.com.
Alternatively, you can also join the discussion on: http://script.kixtart.to.

To find the latest versions of KiXtart, and more sample scripts and tips and tricks on KiXtart, please visit one of the following sites:

http://netnet.net/~swilson/kix.html
http://script.kixtart.to
http://www.scriptlogic.com
http://www.comptrends.com
http://www.cyberramp.net/~musicon/kix/index.html
http://www.hockey.net/~kevinv/kixtart/index.shtml
http://cwashington.netreach.net/

If you are interested in a 'KiXtart-aware' script-editor, please visit:
http://versionzero.webjump.com/