

ALTIRIS  
DOCUMENTATION

**BEST PRACTICES  
ARTICLE**



# **Altiris®**

## **Performing System Administration Tasks With WiseScripts**



altiris®



# Introduction

WiseScript™ is a high-level scripting language that consolidates dozens or hundreds of lines of code into predefined script actions. WiseScript has been used for years to create application installations; however, this document focuses on its use as a tool for quickly creating utilities to automate system administration tasks. (Example: Adding, deleting, and moving files and directories; freeing disk space; and mapping network drives.)

You can create WiseScripts in any of the following products:

- WiseScript for NS (Notification Server) is a WiseScript authoring environment that lets system administrators create .EXEs that automate Altiris tasks in a batch mode and manage resources more effectively. WiseScript for NS is part of the Wise Toolkit that is available to users of Altiris® Client Management Suite™ software.
- WiseScript Editor, like WiseScript for NS, is a WiseScript authoring environment. In addition to automating system administration tasks, installation developers can use it to create .EXEs to use as custom actions in Windows Installer installations. These custom actions can extend the capabilities of Microsoft Windows Installer and simplify installation tasks that are difficult to accomplish with Windows Installer. WiseScript Editor is available in the Wise Package Studio, Wise Installation Studio, and Wise Installation Express products.
- WiseScript Package Editor is an application development tool for creating and editing installation packages based on WiseScript. It is available in Wise Package Studio® Professional Edition and Wise Installation Studio. It formerly was named Wise Installation System.

Although the focus of each of these products is different, most of the WiseScripts that they create are interchangeable. Example: You can create a script in WiseScript Package Editor and use it to automate tasks in the Altiris® Notification Server™ environment.

Topics include:

- [Why Use WiseScript™?](#)
- [About the Script Editor Interface](#) (page 5)
- [Process for Creating a Script](#) (page 6)
- [Sample Script Tutorials](#) (page 7)
- [Resources](#) (page 19)

## Why Use WiseScript™?

Use WiseScript to automate system administration tasks. Its easy-to-use, structured interface lets you create useful scripts in a fraction of the time it would take to write them in a free-form scripting language such as VBScript.

WiseScript excels at retrieving information about a computer, prompting for input (example: passwords) if necessary, and taking action based on that information. This is different from using Altiris® Inventory Solution® software, for example, which can collect snapshot information about a computer's state but cannot take immediate action. In addition, the computer's state might change between the time you take the snapshot and when you can analyze the data and schedule a task to take action. A WiseScript collects and analyzes the data and takes action in real time.

When you have a task that you cannot easily perform with your other tools, reach for WiseScript. You can write a WiseScript to quickly solve an urgent problem, and you can create a library of WiseScripts to resolve common problems and perform routine maintenance. Use Altiris® Software Delivery Solution™ software to deploy your WiseScripts on an as-needed or regularly scheduled basis.

### Scenarios for Using WiseScript

- An end user reports that their default browser does not open when they click on an HTML file. You can create a simple WiseScript that reads the default browser from the registry, verifies that the default browser is installed, and either edits the registry key, installs the default browser, or both.
- You need to update a platform-dependent .DLL file on a collection of computers that are running different operating systems. Instead of creating a different Software Delivery task for each version of the file, you can create one Software Delivery task that runs a WiseScript. The WiseScript finds the version of Windows that is running on each computer and installs the correct .DLL file for that version.
- You want all company computers to display a corporate desktop theme. You can write a WiseScript to:
  - Check the Windows version on the computer.
  - Create a theme directory.
  - Register the path to the theme directory in the registry, using WiseScript's conditional logic to create the registry key in a different location depending on the Windows version.
  - Install the theme file in its directory.

### Additional WiseScript Examples

Following are just a few of the tasks you can accomplish with WiseScript. Samples of some of these scripts are included in this document.

- Move files and directories.
- Modify a machine resource (example: registry key or .INI file).
- Locate and delete a file and its directory (example: to remove a spyware program). See [Locate and Delete a File and its Directory](#) (page 8).
- Free disk space by clearing the Temp directory, the Recycle Bin, or the Internet cache. See [Clear the Temp Directory](#) (page 9).
- Find the current Windows version. See [Find the Current Windows Version](#) (page 10).
- Find and report system information and take action depending on the results. See [Find and Report System Information](#) (page 13).
- Map a network drive. See [Map a Network Drive](#) (page 15).
- Create, edit, or manage virtual software layers on computers that have the Software Virtualization Agent. For details, see the article [Using WiseScripts to Manage and Update Virtual Software Packages](#) in the Altiris Knowledgebase (article 27373).
- Assign license numbers from a text file.

## WiseScript Benefits

- WiseScripts are totally self-contained and do not require a scripting engine on the destination computer.
- WiseScript is a proven, well-tested technology.
  - For more than 10 years, WiseScript has been used to create installations for millions of desktops.
  - Large Internet service providers and several instant messenger clients use WiseScript installations for their applications.
- WiseScripts can run on Windows 3.1, 95, 98, ME, NT, 2000, XP, and Windows Server 2003 with little or no modification.
- WiseScripts can be run silently or, unlike a batch file, with end user interaction.
- WiseScripts are small. The script engine is less than 100 KB.
- WiseScript is easy to use.
  - You build WiseScripts by dragging and dropping actions and completing dialogs.
  - You do not have to declare variables before you use them.
  - Control and logic structures are very easy to build.
- WiseScript is well known.

WiseScript was licensed to Microsoft to be the basis of the SMS Installer, which means that WiseScript is also used by thousands of SMS administrators.
- WiseScript is well supported.

Public Web sites provide hundreds of free scripts.
- WiseScript is powerful.

In addition to the dozens of predefined actions, WiseScripts can call VBScripts and DLL functions, making it possible to use any Windows system call.
- WiseScript is fast.

Because the WiseScript engine is written in C++, when you build a WiseScript, you are building a C++ program. A WiseScript executes faster than a VBScript that performs the same operation.
- WiseScript is extensible.

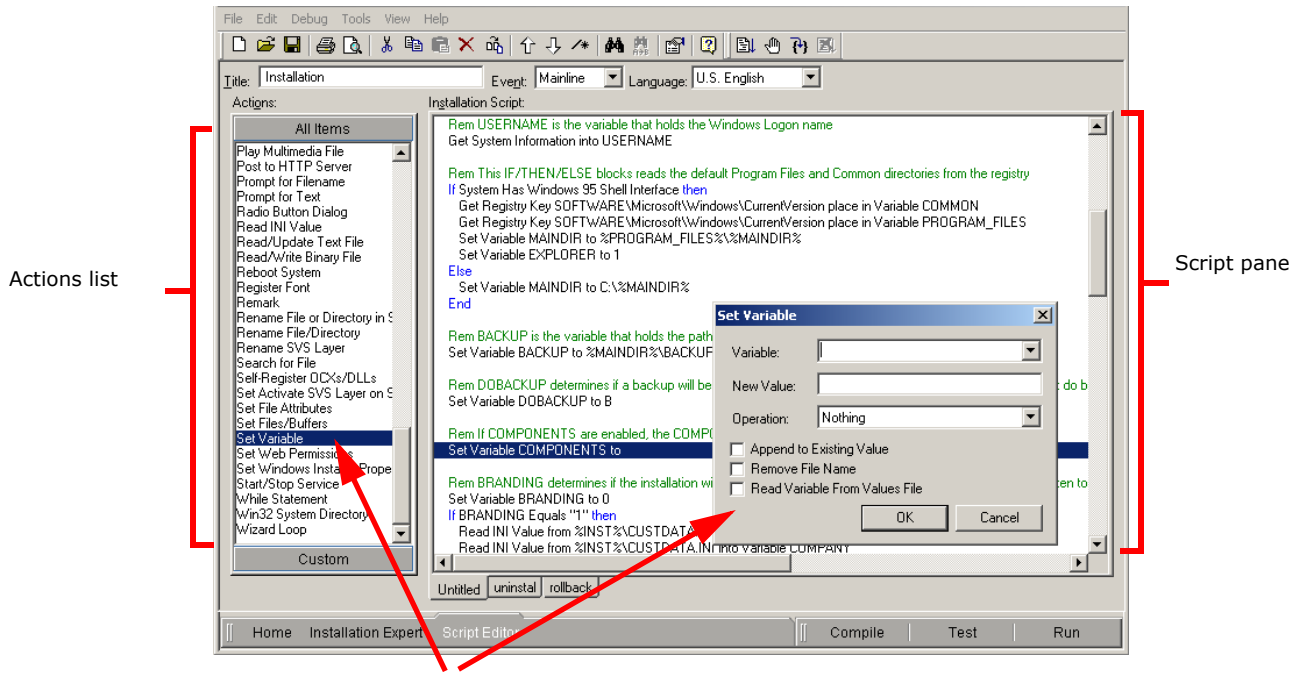
You can streamline your scripting process by creating your own script actions for tasks that you perform frequently. To create a user-defined action, create a WiseScript .WSE (project file) and save it in the Actions subdirectory of the WiseScript product's installation directory. Your action will be available for use in future scripts.

## About the Script Editor Interface

All WiseScript products contain the Script Editor scripting environment. You don't need to memorize commands because Script Editor supports a point-and-click method of scripting. The script you create is displayed in clear, English-like statements.

The script is compiled, along with files and other resources, into an .EXE. When the .EXE is launched, the script runs, executing the actions that are specified in the script.

You create a WiseScript by adding actions to the script pane from a predefined list. When you add an action, a dialog appears that lets you enter the parameters for the action.



Each action opens a dialog that lets you enter parameters.

For more information about the Script Editor interface, see the WiseScript product documentation, which you can open from the Help menu in the WiseScript product.

## Process for Creating a Script

1. Open your WiseScript product.
 

For details, see the WiseScript product documentation, which you can open from the Help menu in the WiseScript product.
2. If a previous WiseScript opens, select File menu > New.
 

If the New Installation File dialog appears, select Blank Script and click OK.
3. Add script actions. See [Adding an Action to a Script](#) (page 7).
  - Each action opens a dialog that lets you enter parameters.

---

**Note**  
Context-sensitive help is available for every script action dialog; press F1 when the dialog appears.

---

  - You can drag actions to rearrange them in the script.
  - You can comment scripts for easier modification later.
4. Save the WiseScript. The project file format is .WSE.

5. Compile the WiseScript by clicking Compile at the lower right of the main window. This creates an .EXE.
6. Test the WiseScript:
  - To test the WiseScript without actually making changes to your computer, click Test at the lower right of the main window. This is most useful for testing any user interface elements of the WiseScript.
  - To run the WiseScript on your computer, click Run at the lower right of the main window. This will make changes to your computer.
7. If you are using WiseScript within the Altiris® Notification Server™ software, you can use the Altiris Console to:
  - a. Create a Software Delivery package that runs the WiseScript program.
  - b. Configure a Software Delivery task to run the package on a specific collection of computers.
  - c. Schedule the task as needed.

For details, see the appropriate Altiris documentation at [www.altiris.com/Support/Documentation.aspx](http://www.altiris.com/Support/Documentation.aspx).

## Adding an Action to a Script

In Script Editor, do any of the following:

- From the **Actions** list in the left pane, drag an action onto a line in the **Installation Script** list in the right pane. The new action appears above the line that is highlighted when you drop the action.
- Click in the script and double-click the action in the **Actions** list to place the new action above the line you clicked.
- Click in the script and start typing the first few letters of the action name. As you type, the current line becomes a drop-down list with all the action names, and the action that most closely matches the letters you typed is the current item in the list. When the action you want is the current item in the list, press Enter.

When you add an action, a dialog appears that lets you set the parameters for the action unless it does not require parameters. When you add a Custom Dialog or Custom Billboard action, the appropriate editing environment opens.

Some actions come in pairs. (Example: When you add an If action, you must also add an End action at the end of the condition block.) Script Editor indents actions inside these pairs.

Context-sensitive help is available for every script action dialog; press F1 when the dialog appears.

## Sample Script Tutorials

This section contains tutorials that provide an introduction to WiseScript while leading you through the creation of several useful scripts. No knowledge of WiseScript is necessary to complete the tutorials.

Typically, you would use follow normal Software Delivery procedures to deliver WiseScripts to and run them on managed computers. However, in this tutorial you will run these sample scripts on your computer.

The tutorials begin with simple scripts that perform useful tasks, such as removing “spyware” and emptying the Temp directory. The later tutorials cover more advanced functions: using conditional logic to guide execution, writing information to dialogs, and calling .DLLs.

To complete the tutorials, you must have access to a WiseScript product. (For a list of WiseScript products, see [Introduction](#) on page 3.) If you do not have a WiseScript product, you can read through the tutorials to get an idea of how easy it is to create WiseScripts.

See:

- [Locate and Delete a File and its Directory](#) (page 8)
- [Clear the Temp Directory](#) (page 9)
- [Find the Current Windows Version](#) (page 10)
- [Find and Report System Information](#) (page 13)
- [Map a Network Drive](#) (page 15)

## Locate and Delete a File and its Directory

Scenario: You need to locate a newly-discovered spyware program on an infected computer and delete that program and its container directory.

In this example, you will take advantage of WiseScript’s extensive file search functionality to locate the spyware file. You also will use WiseScript variables to store the path to the spyware and to provide the path to the files to delete. Then you will remove the file and delete the host directory.

To simulate a spyware program and its related directories and files, you will create several sample directories and harmless text files on your computer.

### To prepare sample directories and files for this WiseScript to delete:

1. Create the directory Program Files\Sample.
2. Create a text file named Sample.txt and add it to the directory you just created.
3. Create one or more subdirectories under Program Files\Sample and add one or more files to each subdirectory. It doesn’t matter what names you give to these subdirectories and files.

### To create the sample script:

1. Open your WiseScript product.
2. If a previous WiseScript opens, select File menu > New.  
If the New Installation File dialog appears, select Blank Script and click OK.
3. In the **Actions** list on the left, double-click the Search for File action.
4. Complete the Search for File Settings dialog as follows and then click OK. (Leave the defaults for any values that are not listed here.)
  - **File Name**  
Enter Sample.txt



- **Variable Name**  
Enter FILELOC  
  
This variable will store the path to the file.
  - **Remove File Name**  
Select this checkbox to strip the filename from the path that is stored in the variable.
5. Drag the Delete File(s) action to the right pane, below the first action, complete its dialog as follows, and then click OK.
- **Pathname**  
Enter %FILELOC%\\*.\*  
  
This uses the path returned from the search function and uses wildcards to specify every file in the directory.
  - **Include Sub-Directories**  
Select this checkbox
  - **Remove Directory Containing Files**  
Select this checkbox

Your script should look like this:

```
Search for file Sample.txt place in Variable FILELOC
Delete File(s) %FILELOC%\*.*
```

**To compile and verify the script:**

1. Save the WiseScript as Delete\_File.wse. The extension .WSE represents a WiseScript project file.
2. Click Compile at the lower right of the main window. This creates an .EXE with the same name as the project file.
3. In the WiseScript Editor, click Run in the lower right of the window.  
  
The WiseScript runs on your computer.
4. In Windows Explorer, verify that all the files and directories have been deleted and that the Sample directory no longer appears under Program Files.

## Clear the Temp Directory

Scenario: You need to clear the current user’s Temp directory to free space for a new application.

In this example, the script reads the TEMP environment variable to get the path to the current user’s Temp directory. Then it uses the Delete File(s) action to clear the directory.

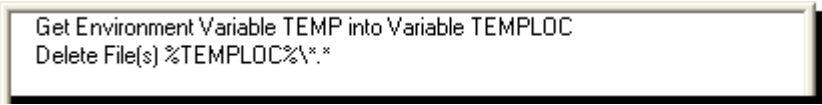
You could use a similar script to empty the Recycle bin or clear the Internet cache.

An advantage of WiseScript’s Delete File(s) action is that it attempts to delete every file in the directory individually. That way, when it encounters a read-only file (example: an Altiris agent XML file) or a file that is in use, it skips that file and proceeds to the next file. This is an improvement over the DOS delete \*.\* command, which stops when it encounters the first read-only file.

### To create the sample script:

1. Open your WiseScript product.
2. If a previous WiseScript opens, select File menu > New.  
If the New Installation File dialog appears, select Blank Script and click OK.
3. In the **Actions** list on the left, double-click the Get Environment Variable action.
4. Complete the Get Environment Variable dialog as follows and then click OK. (Leave the defaults for any values that are not listed here.)
  - **Env. Variable**  
Enter TEMP
  - **Variable Name**  
Enter TEMPLOC
5. Drag the Delete File(s) action to the right pane (below the first action), complete its dialog as follows, and then click OK.
  - **Pathname**  
Enter %TEMPLOC%\\*.\*
  - **Include Sub-Directories**  
Select this checkbox

Your script should look like this:



```
Get Environment Variable TEMP into Variable TEMPLOC
Delete File(s) %TEMPLOC%\*.*
```

### To compile and verify the script:

1. In the WiseScript Editor, click Run in the lower right of the window.
2. In the Save As dialog, name the file Clear\_Temp.wse.  
The file is saved and compiled, and the compiled WiseScript runs on your computer.
3. In Windows Explorer, verify that the current user's Temp directory is empty.

---

#### Note

Read-only files (example: Altiris temp files) or files that are in use might not be deleted.

---

## Find the Current Windows Version

Scenario: You need to update a file on a collection of computers. Because the file is platform-dependent, you need to find the version of Windows that is running on each computer and install the correct file for that version.

Instead of creating a different Software Delivery task for each version of the file, you can create one Software Delivery task to run a WiseScript that determines the operating system on each computer and installs the appropriate file.

In this example, you will create a script that collects the numeric Windows version from the computer. Instead of installing a file, the script displays a message identifying the

Windows version. In a production environment, you would use a combination of If Statements and Install File(s) actions to install the file.

Ideally, this sample test could look like this:

If WindowsVersion is less than 5.1, then Windows = Windows 2000

If WindowsVersion is greater than or equal to 5.1 AND less than 5.2, then Windows = Windows XP

If WindowsVersion is greater than or equal to 5.2, then Windows = Windows Server 2003

However, WiseScript does not use logical ANDs and ORs; a WiseScript If Statement can take one argument only. Therefore, you must nest the conditional statements, resulting in logic that looks like this:

If WindowsVersion is less than 5.1, then Windows = Windows 2000

Else If WindowsVersion is greater than or equal to 5.1, then

If WindowsVersion is less than 5.2, then Windows = Windows XP

Else If WindowsVersion is greater than or equal to 5.2, then Windows = Windows Server 2003

End inner If statement

End outer If statement

---

#### **Note**

This procedure assumes that you know how to add actions to a script. For help, see [Adding an Action to a Script](#) (page 7).

---

#### **To create the sample script:**

1. Open your WiseScript product.
2. If a previous WiseScript opens, select File menu > New.  
If the New Installation File dialog appears, select Blank Script and click OK.
3. To get the operating system version number, add a Get System Information action, complete its dialog, and click OK:

- **Variable**

Enter OS

- **Retrieve**

Select **Windows Version**

The next section of the script performs logical tests on the OS variable and stores the results in a new variable, OSNAME.

4. Add an If Statement action, complete its dialog, and click OK:

- **If Variable**

Enter OS

- **operator drop-down list (unnamed)**

Select **Less Than**

- **The Value**

Enter 5.1

5. Add a Set Variable action, complete its dialog, and click OK:
  - **Variable**  
Enter OSNAME
  - **New Value**  
Enter Windows 2000
6. Add an ElseIf Statement action, complete its dialog, and click OK:
  - **If Variable**  
Enter OS
  - **operator drop-down list (unnamed)**  
Select **Greater than or Equal**
  - **The Value**  
Enter 5.1
7. Add an If Statement action, complete its dialog, and click OK:
  - **If Variable**  
Enter OS
  - **operator drop-down list (unnamed)**  
Select **Less Than**
  - **The Value**  
Enter 5.2
8. Add a Set Variable action, complete its dialog, and click OK:
  - **Variable**  
Enter OSNAME
  - **New Value**  
Enter Windows XP
9. Add an ElseIf Statement action, complete its dialog, and click OK:
  - **If Variable**  
Enter OS
  - **operator drop-down list (unnamed)**  
Select **Greater than or Equal**
  - **The Value**  
Enter 5.2
10. Add a Set Variable action, complete its dialog, and click OK:
  - **Variable**  
Enter OSNAME
  - **New Value**  
Enter Windows Server 2003

---

**Note**

To check for a version of Windows that is not mentioned here, add statements to the condition block for each additional version.

---

11. Close the condition block by adding an End Statement for each level of the block:
  - a. Add an End Statement action below the last line in the script. This closes the inner block.

- b. Add a second End Statement action. This closes the outer block.
12. To display the value of the OSNAME variable, add a Display Message action, complete its dialog, and click OK:
- **Message Title**  
Enter Current Windows Version
  - **Message Text**  
Enter This version of Windows is %OSNAME%.
  - **Message Icon**  
Select **Information**

Your completed script should look like this:

```
Get System Information into OS
If OS Less Than "5.1" then
  Set Variable OSNAME to Windows 2000
Elseif OS Greater Than or Equal "5.1" then
  If OS Less Than "5.2" then
    Set Variable OSNAME to Windows XP
  Elseif OS Greater Than or Equal "5.2" then
    Set Variable OSNAME to Windows Server 2003
  End
End
End
Display Message "Current Windows Version"
```

### To compile and verify the script:

1. In the WiseScript Editor, click Run in the lower right of the window.
2. In the Save As dialog, name the file Find\_Windows\_Ver.wse.  
The file is saved and compiled, and the compiled WiseScript runs on your computer.
3. Verify that the message displays the version of Windows that is on your computer.
4. Click OK to clear the message.

## Find and Report System Information

Scenario: You need to collect information about a particular computer.

When trying to diagnose a network or software problem, system administrators often need to get information from the managed computer, such as version numbers of specific files, the values of certain registry keys, what server the computer is logged onto, and so on. Remote control can provide some of this information, but it's a manual, time-consuming process, prone to mistakes. By using WiseScript, administrators can create re-usable scripts that can find this information and report it to the screen, write it to a text file, or even post it to an HTTP server.

More importantly, you can use the conditional logic built in to WiseScript to take action based on the information that is collected. This action can include deleting files, setting registry key values, executing programs, or calling functions within .DLLs for very specific and advanced tasks.

In this example, you will create a data-mining script that quickly finds information about a computer. In this tutorial environment, this information is displayed on the screen. In

a production environment, you would take appropriate action based on what the script discovers.

---

**Note**

This procedure assumes that you know how to add actions to a script. For help, see [Adding an Action to a Script](#) (page 7).

---

**To create the sample script:**

1. Open your WiseScript product.
2. If a previous WiseScript opens, select File menu > New.  
If the New Installation File dialog appears, select Blank Script and click OK.
3. To get the current time, add a Get System Information action, complete its dialog, and click OK:
  - **Variable**  
Enter TIME
  - **Retrieve**  
Select **Current Date/Time**
4. To get the operating system version, add a Get System Information action, complete its dialog, and click OK:
  - **Variable**  
Enter OS
  - **Retrieve**  
Select **Windows Version**

The next section of the script uses environment variables to collect network-specific information. For information on obtaining environment variables, see [Resources](#) (page 19).

5. Add a Get Environment Variable action, complete its dialog, and click OK:
  - **Env. Variable**  
Enter PROCESSOR\_IDENTIFIER
  - **Variable Name**  
Enter PROCESSOR
6. Add a Get Environment Variable action, complete its dialog, and click OK:
  - **Env. Variable**  
Enter USERNAME
  - **Variable Name**  
Enter USER
7. Add a Get Environment Variable action, complete its dialog, and click OK:
  - **Env. Variable**  
Enter USERDOMAIN
  - **Variable Name**  
Enter DOMAIN
8. Add a Get Environment Variable action, complete its dialog, and click OK:

- **Env. Variable**  
Enter LOGONSERVER
    - **Variable Name**  
Enter LOGON
- 9. To display the information that is collected, add a Display Message action, complete its dialog, and click OK:
  - **Message Title**  
Enter Current System Information
  - **Message Text**  
Press Ctrl+Enter after each of the following lines to enter carriage returns in the message box without closing the dialog.  
  
 The current time is: %TIME%  
 The current Windows version is: %OS%  
 The processor type is: %PROCESSOR%  
 The logged on user is: %USER%  
 The computer belongs to the %DOMAIN% domain  
 It is logged onto server: %LOGON%
  - **No Cancel**  
Select this checkbox to make this an informational message rather than one that asks the end user for a decision.

Your completed script should look like this:

```
Get System Information into TIME
Get System Information into OS
Get Environment Variable PROCESSOR_IDENTIFIER into Variable PROCESSOR
Get Environment Variable USERNAME into Variable USER
Get Environment Variable USERDOMAIN into Variable DOMAIN
Get Environment Variable LOGONSERVER into Variable LOGON
Display Message "Current System Information"
```

**To compile and verify the script:**

1. In the WiseScript Editor, click Run in the lower right of the window.
2. In the Save As dialog, name the file Report\_System\_Info.wse.  
The file is saved and compiled, and the compiled WiseScript runs on your computer.
3. Verify that the system information is displayed correctly.
4. Click OK to clear the message.

## Map a Network Drive

Scenario: To accommodate increased file storage requirements, you have added a new shared drive to your network. Now you need to map the new shared network drive on all your managed computers.

In this example, you will create a script that maps a shared network drive to a drive letter that you specify. This is accomplished by calling a Microsoft .DLL (mpr.dll), which should be in your System32 directory. When you add the action to call the .DLL, you specify parameters to pass to the .DLL during execution.

---

**Note**

This procedure assumes that you know how to add actions to a script. For help, see [Adding an Action to a Script](#) (page 7).

---

**To create the sample script:**

1. Open your WiseScript product.
2. If a previous WiseScript opens, select File menu > New.  
If the New Installation File dialog appears, select Blank Script and click OK.
3. Add a Set Variable action, complete its dialog, and click OK:
  - **Variable**  
Enter SHARE
  - **New Value**  
Enter a valid, shared drive in this format: `\\SERVER\SHARE`
4. Add a Set Variable action, complete its dialog, and click OK:
  - **Variable**  
Enter DRIVELETTER
  - **New Value**  
Enter the drive letter to map (example: J:)
5. Add a Call DLL Function action. This calls a Microsoft .DLL to perform the drive mapping.
  - a. Complete the upper portion of the Call DLL Function dialog:
    - ◆ **DLL Pathname**  
Enter %SYS32%\mpr.dll
    - ◆ **Function Name**  
Enter WNetAddConnectionA
    - ◆ **Call a function with variable parameter list**  
Select this option
  - b. On the Call DLL Function dialog, click Add. Complete the DLL Parameter Settings dialog that appears and click OK:
    - ◆ **Parameter Type**  
Select **string pointer**
    - ◆ **Passing Type**  
Select **Normal**
    - ◆ **Value Source**  
Select **Variable**
    - ◆ **Variable Name**  
Enter SHARE

The parameter is added to the list on the Call DLL Function dialog.
  - c. On the Call DLL Function dialog, click Add again. Complete the DLL Parameter Settings dialog that appears and click OK:
    - ◆ **Parameter Type**  
Select **string pointer**



- ◆ **Passing Type**  
Select **Normal**

- ◆ **Value Source**  
Select **Constant with NULL value**

This passes an empty string to the password for the connection and uses the current user's password. To pass a NULL value (example: When the password is NULL), set the parameter type to **dword**, set the value source to **Constant**, and enter 0 as the constant value.

The parameter is added to the list on the Call DLL Function dialog.

- d. On the Call DLL Function dialog, click Add again. Complete the DLL Parameter Settings dialog that appears and click OK:

- ◆ **Parameter Type**  
Select **string pointer**

- ◆ **Passing Type**  
Select **Normal**

- ◆ **Value Source**  
Select **Variable**

- ◆ **Variable Name**  
Enter DRIVELETTER

The parameter is added to the list on the Call DLL Function dialog.

- e. On the Call DLL Function dialog, set the following options:

- ◆ **Return Value Type**  
Select **dword**

- ◆ **Returned Variable**  
Enter RETURN

- f. Click OK on the Call DLL Functions dialog.

The final section of the script contains a condition block to display either of two messages, depending on the results of the mapping.

6. Add an If Statement action, complete its dialog, and click OK:

- **If Variable**  
Enter RETURN

- **operator drop-down list (unnamed)**  
Select **Equals**

- **The Value**  
Enter 0

7. Add a Display Message action, complete its dialog, and click OK:

- **Message Title**  
Enter Create Mapping

- **Message Text**  
Enter Successfully mapped %DRIVELETTER% to %SHARE%.

- **Message Icon**  
Select **Information**



# Resources

Typically, it is better to use computer-specific information in WiseScripts instead of hard-coding paths and so on. Following are several registry keys and environment variables that contain this information.

## Registry Keys

Common Folders, Current User:

HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders

Common Folders, Local Machine:

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders

## Environment Variables

You can obtain a complete list of available environment variables by opening a command prompt and typing SET. The following table lists some common variables.

Environment Variable	Represents
COMPUTERNAME	Machine name of the computer
HOMEPATH	Current user's home directory (Documents and Settings\User)
LOGONSERVER	The server that validated the current logon
OS	Current operating system name (example: Windows_NT)
USERDOMAIN	The domain name the computer is logged on to
USERDNSDOMAIN	The fully qualified version of USERDOMAIN
USERNAME	The current user's logon name

## For More Sample Scripts

### Sample Scripts Provided by Altiris

The Samples subdirectory in the WiseScript product's installation directory contains an assortment of WiseScripts (.WSE files) that perform complex actions and provide interactivity to the end user. Use these to further your knowledge of WiseScripts. These sample scripts do not provide step-by-step instructions for solving a particular problem. They provide examples that you can study and use as a basis for your own scripts.

For details on the sample scripts, see ScriptHelp.htm, which is also in the Samples subdirectory.

### Third-party Sample Scripts

The following sites contain repositories of both WiseScripts and compatible SMS Scripts:

MyITForum: [www.myitforum.com](http://www.myitforum.com)

Dragonsoft: [www.wisecscript.dragonsoft.ru.com](http://www.wisecscript.dragonsoft.ru.com) (contains more than 200 WiseScripts)

