Using Prnadmin.dll to Manage Printer Objects

Microsoft Corporation
Published: January 2003

Abstract

This white paper explains how IT administrators and programmers can create scripts by using the COM-based DLL PrnAdmin to control printers, drivers, and ports on local and remote computers. This white paper includes information about installing the DLL, and information about printer, driver, port, and form objects. Each section about objects includes reference information about the object interface and information about how to create scripts to manage the object. Following these sections are appendices that contain information about error handling and sample scripts. This advanced-level paper assumes readers are familiar with a scripting language, such as Microsoft® Visual Basic® or any other programming language that supports COM programming.

Contents

Introduction

Prnadmin.dll (PrnAdmin) is a COM-based DLL that provides large-scale, noninteractive control of printers, drivers, ports, and forms on local and remote computers. With PrnAdmin, you can write scripts to utilize the functionality in Printui.dll, and you can manage network printing resources such as TCP print ports.

You can create scripts that use PrnAdmin to accomplish the following tasks:

- Add and delete a local or remote printer.
- Add and delete printer connections.
- Add and delete a local or remote form.
- Add and delete a local or remote TCP port (LPR or RAW)
- Add and delete a local or remote printer driver.
- View a list of printers, ports, drivers, and forms on a local or remote computer.
- Control and configure a local or remote printer.

Intended for organizations that need to manage a large number of printers, drivers, ports and forms, PrnAdmin helps administrators and programmers manage local and remote resources running Microsoft® Windows NT® 4.0, Microsoft® Windows® 2000, or Windows Server 2003.

Notes

You cannot use the Windows Management Instrumentation (WMI) print classes on Windows NT 4.0.

There is limited support for WMI print classes in Windows 2000. These print classes are read-only.

The most comprehensive support for WMI print classes exists in Windows Server 2003 and Windows XP. These WMI print classes are read-enabled and write-enabled.

This paper contains the following sections:

Installing Prnadmin.dll

Prnadmin.dll is available on the *Microsoft Windows Server 2003 Resource Kit* companion CD and is installed when you install the Resource Kit Tools. After you install the resource kit, verify that the file exists, and then use Regsvr32 to register the DLL as a command component in the registry.

Note

To register prnadmin.dll you need to be an administrator or power user on the local machine. The “write/set” functionality in prnadmin requires print server or print queue administrative permissions.

To verify PrnAdmin is installed:

Go to the following directory:

%Drive%:/Program Files/Windows Server 2003 Resource Kit

Verify that the file Prnadmin.dll exists in the directory.

To register PrnAdmin:

Click **Start**, and then click **Command Prompt**.

Type the following at the command line:

Regsvr32 %Drive%:\Program Files\Windows Server 2003 Resource Kit\Prnadmin.dll

A message box similar to Figure 1 will be displayed.

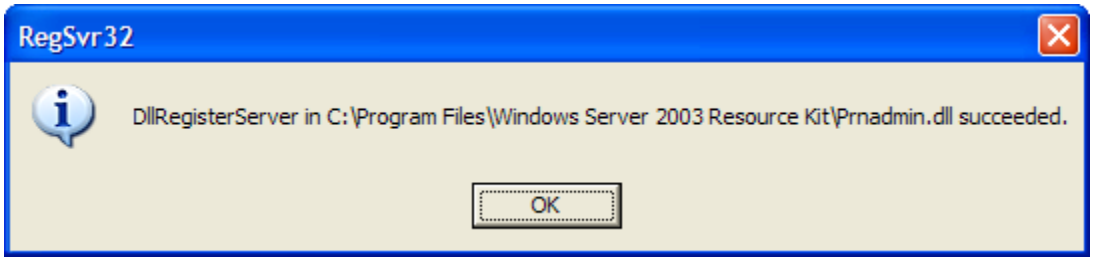

Figure 1: Successfully Registering Prnadmin.dll

Printer Objects

The Printer object is the primary method used to access the printer. It consists of an interface and functionality that you use in your scripts to manage the printer.

Using the Printer Object Interface

The Printer object has many properties and uses different types of property values. Some properties you can read and write, others you can read only or write only. Some values are strings or integers, some are boolean. Some properties only work on specific operating systems. The following tables identify how you can interact with the property, the property name, its type, and the function for which it is used. If an operating system is listed, it is the only operating system on which you can use the property. The tables include the following information:

- Table 1 lists the read and write Printer object properties, data types, and uses.
- Table 2 lists the read-only Printer object properties, data types, and uses.
- Table 3 lists the write-only Printer object properties, data types, and uses.

For more information about each property in the PRINTER_INFO_2 structure, see the Windows Server 2003 Platform SDK topic “PRINTER_INFO_2.”

Table 1. Read and Write Printer Object Properties

Property	Data Type	Use
Comment	String	Provides a brief description of the printer.
DataType	String	Specifies the data type used to record the print job.
DefaultPriority	Integer	Specifies the default priority value assigned to each print job.
DriverName	String	Specifies the name of the printer driver.
DriverPath	String	Specifies the path to the driver files
InfName	String	Specifies the path and name of the INF file
Location	String	Specifies the physical location of the printer.
PortName	String	Identifies one or more ports used to transmit data to the printer.
PrinterName	String	Specifies the name of the printer.
PrintProcessor	String	Specifies the name of the print processor used by the printer.
Priority	Integer	Specifies a priority value that the spooler uses to route print jobs.
SepFile	String	Specifies the name of the file used to create the separator page.
ServerName	String	Identifies the server that controls the printer. If this string is NULL, the printer is controlled locally.
ShareName	String	Identifies the share point for the printer.
StartTime	Integer	Specifies the earliest time at which the printer will print a job. This value is expressed as minutes elapsed since 12:00 A.M. Coordinated Universal Time (Greenwich Mean Time).
UntilTime	Integer	Specifies the latest time at which the printer will print a job. This value is expressed as minutes elapsed since 12:00 A.M. Coordinated Universal Time.

Table 2. Read-Only Printer Object Properties

Property	Data Type	Use
Attributes	Integer	Specifies the printer attributes. This property can have a combination of values.
AveragePPM	Integer	Specifies the average number of pages per minute that have been printed on the printer.
Jobs	Integer	Specifies the number of print jobs that have been queued for the printer.
Parameters	String	Specifies the default print-processor parameters.
Status	Integer	Specifies the printer status. This property can have a combination of values.

Table 3. Write-Only Printer Object Properties

Property	Data Type	Use
AttributeString	String	Specifies the printer attributes. This property can have a combination of values.
Default	Boolean	**Windows 95, Windows 98, or Windows Millennium Edition (Me):** Indicates the printer is the default printer in the system.
Direct	Boolean	The job is sent directly to the printer (it is not spooled).
DoCompleteFirst	Boolean	If set, and the printer is set for print-while-spooling, any jobs that have completed spooling are scheduled to be printed before jobs that have not completed spooling.
EnableBidi	Boolean	**Windows 95, Windows 98, or Windows Me:** Indicates whether bidirectional communications are enabled for the printer.
EnableDevq	Boolean	If set, **DevQueryPrint** is called. **DevQueryPrint** might fail if the document and printer setups do not match. Setting this flag causes mismatched documents to be held in the queue.
Hidden	Boolean	Specifies a reserved printer.
KeepPrintedJobs	Boolean	If set, jobs are kept after they are printed. If unset, jobs are deleted.
Local	Boolean	Specifies that the printer is a local printer.
Network	Boolean	Specifies that the printer is a network printer connection.
NewName	Boolean	Specifies the new name of the printer (when being renamed)
Published	Boolean	**Windows 2000 or Windows XP or Windows 2003 Server:** Indicates whether the printer is published in the directory service.
Queued	Boolean	If set, the printer spools and starts printing after the last page is spooled. If not set, and PRINTER_ATTRIBUTE_DIRECT is not set, the printer spools and prints while spooling.
RawOnly	Boolean	Indicates that only raw data type print jobs can be spooled.
Shared	Boolean	Specifies that the printer is a shared printer.
WorkOffline	Boolean	**Windows 95, Windows 98, or Windows Me:** Indicates whether the printer is currently connected.

Managing Printers

All operations on printers are done via the PrintMaster object. For most of them, the existence of a Printer object is required also. Creating these objects are included in the task instructions.

This section provides sample code for the following tasks:

Adding a Printer

The sample code in this section creates any required objects, adds a printer to a remote server, and configures some driver and port information.

```
dim oMaster
dim oPrinter

'The following code creates the required PrintMaster and Printer objects.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set oPrinter = CreateObject("Printer.Printer.1")

'The following code specifies the name of the computer where the printer will be added. To specify the local
'computer, either use empty quotes ("") for the computer name, or do not use the following line of code. If
'ServerName is not set, the local computer is used. Always precede the name of a remote computer
'with two backslashes (\\).
oPrinter.ServerName  = "\\server"

'The following code assigns a name to the printer. The string is required and cannot be empty.
oPrinter.PrinterName = "my printer"

'The following code specifies the printer driver to use. The string is required and cannot be empty.
oPrinter.DriverName  = "a driver x100"
```

```
'The following code specifies the printer port to use. The string is required and cannot be empty.
oPrinter.PortName    = "lpt1:"

'The following code specifies the location of the printer driver. This setting is optional, because by default
'the drivers are picked up from the driver cache directory.
oPrinter.DriverPath  = "c:\drivers"

'The following code specifies the location of the INF file. This setting is optional, because by default the INF
'file is picked up from the %windir%\inf\ntprint.inf directory.
oPrinter.InfFile     = "c:\winnt\inf\ntprint.inf"

'The following code adds the printer.
oMaster.PrinterAdd oPrinter

'The following code uses the Err object to determine whether the printer was added successfully.
if Err <> 0 then
        'An error occurred
end if
```

If you want to configure other printer settings, such as comments, create a Printer object and then call PrintMaster's method PrinterSet.

Deleting a Printer

The sample code in this section creates any required objects and then deletes a printer from a remote server.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code specifies the name of the printer on a remote computer to delete. To delete a local
'printer, use empty quotes ("") in place of ServerName.
oMaster.PrinterDel ServerName, PrinterName

'The following code uses the Err object to determine whether the printer was deleted successfully.
if Err <> 0 then
        'An error occurred
end if
```

Adding a Printer Connection

The sample code in this section creates any required objects and adds a connection on the local computer to a printer already installed on another computer.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code adds a connection on the local computer to a printer or a share name. The printer
'connection is a per-user resource. The PrinterName value must use the following form: \\server\my printer.
```

```
oMaster.PrinterConnectionAdd PrinterName

'The following code uses the Err object to determine whether the printer was deleted successfully.
if Err <> 0 then
        'An error occurred
end if
```

Deleting a Printer Connection
The sample code in this section creates any required objects and deletes a connection from the local computer to a printer installed on another computer.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code deletes a connection on the local computer to a printer. The ServerName property is
'not used because the printer connection is per user. The PrinterName must be the name of a printer and
'not a share. PrinterName must use the following form: \\server\myprinter.
oMaster.PrinterConnectionDel PrinterName

'The following code uses the Err object to determine whether the printer was deleted successfully.
if Err <> 0 then
        'An error occurred
end if
```

Getting the Configuration of a Printer
The sample code in this section creates any required objects, gets the configuration settings of a local and remote printer, and uses error handling to determine if the code was successful in getting the printer settings.

```
dim oMaster
dim oPrinter

'The following code creates the required PrintMaster and Printer objects.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set oPrinter = CreateObject("Printer.Printer.1")

'The following code gets the local printer's configuration settings.
oMaster.PrinterGet "","My Printer", oPrinter

'The following code gets a remote printer's configuration settings without specifying the name of the remote
'server where the printer is installed.
oMaster.PrinterGet "\\server", "My Printer", oPrinter

'The following code gets a remote printer's configuration settings and specifies the name of the remote
'server where the printer is installed.
oMaster.PrinterGet "", "\\server\My Printer", oPrinter

'The following code uses the Err object to determine whether the printer was deleted successfully.
if Err = 0 then

'The following code prints out the configuration if no errors occurred.
wscript.echo "PrinterName:  " & oPrinter.PrinterName
```

```
wscript.echo "ShareName:    " & oPrinter.ShareName
wscript.echo "PortName:     " & oPrinter.PortName
wscript.echo "DriverName    " & oPrinter.DriverName
wscript.echo "Comment:      " & oPrinter.Comment
wscript.echo "Location:     " & oPrinter.Location
wscript.echo "SepFile:      " & oPrinter.Sepfile
wscript.echo "PrintProc:    " & oPrinter.PrintProcessor
wscript.echo "Datatype:     " & oPrinter.Datatype
wscript.echo "Parameters:   " & oPrinter.Parameters
wscript.echo "Attributes:   " & CStr(oPrinter.Attributes)
wscript.echo "Priority:     " & CStr(oPrinter.Priority)
wscript.echo "DefaultPri:   " & CStr(oPrinter.DefaultPriority)
wscript.echo "StartTime:    " & CStr(oPrinter.StartTime)
wscript.echo "UntilTime:    " & CStr(oPrinter.UntilTime)
wscript.echo "Status:       " & CStr(oPrinter.Status)
wscript.echo "Jobcount:     " & CStr(oPrinter.Jobs)
wscript.echo "AveragePPM    " & CStr(oPrinter.AveragePPM)
end if
```

For more information about the configuration settings used above, see Using the Printer Object Interface, the Prncfg.vbs script in the %Drive%:\Program Files\Windows Server 2003 Resource Kit directory, and the Windows Server 2003 Platform SDK.

Configuring a Printer

The sample code in this section creates any required objects, gets the existing printer configuration, and then sets some properties. You can choose which settings you want to configure. For information about Printer object properties and how they can be set, see Using the Printer Object Interface.

```
dim oMaster
dim oPrinter

'The following code creates the required PrintMaster and Printer objects.

set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set oPrinter = CreateObject("Printer.Printer.1")

'The following code gets the existing printer settings. This step is optional; however, if you do not use it, you
'will not know which options have been left on the default settings. Because there are so many properties
'you can set, you might fail to enter a value, and the default settings might cause unwanted effects.

oMaster.PrinterGet "\\server", "MyPrinter", oPrinter

'The following code sets the ServerName and is used only for a remote printer if you did not use 'PrinterGet.
'If you did use PrinterGet, it fills in the ServerName property for you.

oPrinter.ServerName = "\\server"

'The following code sets some printer properties. Because there are so many settings, the following code
'only configures a few settings. You might or might not want to set any of the following properties.

oPrinter.PortName = "lpt1:"
oPrinter.ShareName = "my share"
oPrinter.Location = "my office"
oPrinter.Comment = "my good printer"
oPrinter.DataType = "RAW"
oPrinter.NewName = "New Printer Name"
oPrinter.SepFile = "c:\sep-file"
'The following code sets printer attributes. They can be set to true or false to enable or disable the setting.

oPrinter.Queued = true / false
oPrinter.Direct = true / false
oPrinter.Default = true / false
oPrinter.Shared = true / false
oPrinter.Hidden = true / false
oPrinter.EnableDevq = true / false
oPrinter.KeepPrintedJobs = true / false
```

```
oPrinter.DoCompleteFirst = true / false
oPrinter.WorkOffline = true / false
oPrinter.EnableBidi = true / false
oPrinter.RawOnly = true / false
oPrinter.Published = true / false
‘The following code saves the printer settings.
oMaster.PrinterSet oPrinter

‘The following code uses the Err object to determine whether the printer settings were updated successfully.
if Err <> 0 then
        ‘An error occurred
end if
```

Pausing a Printer

The sample code in this section creates any required objects and then pauses a local and remote printer.

```
dim oMaster

‘The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code pauses a local printer.
oMaster.PrinterPause "", "MyLocalPrinter"

‘The following code pauses a remote printer.
oMaster.PrinterPause "\\server", "MyPrinter"

‘The following code can also be used to pause a remote printer.
oMaster.PrinterPause "", "\\server\MyPrinter"

‘The following code uses the Err object to determine whether the printer was paused successfully.
if Err <> 0 then
        ‘An error occurred
end if
```

Resuming a Printer

The sample code in this section creates any required objects and then resumes printer services for a local and remote printer.

```
dim oMaster

‘The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code resumes a local printer.
oMaster.PrinterResume "", "MyLocalPrinter"

‘The following code resumes a remote printer.
oMaster.PrinterResume "\\server", "MyPrinter"

‘The following code can also be used to resume a remote printer.
oMaster.PrinterResume "", "\\server\MyPrinter"

‘The following code uses the Err object to determine whether the printer services were resumed
’successfully.
if Err <> 0 then
```

```
        'An error occurred
end if
```

Purging a Printer
The sample code in this section creates any required objects and then purges all print jobs from a local and remote printer.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code purges a local printer.
oMaster.PrinterPurge "", "MyLocalPrinter"

'The following code purges a remote printer.
oMaster.PrinterPurge "\\server", "MyPrinter"

'The following code can also be used to resume a remote printer.
oMaster.PrinterPurge "", "\\server\MyPrinter"

'The following code uses the Err object to determine whether the printer was purged successfully.
if Err <> 0 then
        'An error occurred
end if
```

Printing a Test Page
The sample code in this section creates any required objects and then prints a test page on a local and remote printer.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code prints a test page on a local printer.
oMaster.PrintTestPage "", "MyLocalPrinter"

'The following code prints a test page on a remote printer.
oMaster.PrintTestPage "\\server", "MyPrinter"

'The following code can also be used to print a test page on a remote printer.
oMaster.PrintTestPage "", "\\server\MyPrinter"

'The following code uses the Err object to determine whether the test page was printed successfully.
if Err <> 0 then
        'An error occurred
end if
```

Enumerating Printers
The sample code in this section creates any required objects and then lists all printers on a specific computer. Sample code is provided to include local printers in the list.

```
dim oMaster

'The following code creates the required PrintMaster object.
```

```
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code specifies the name of a computer. The printer list will be created based on printers
‘installed on the specified computer. The only way to list printers connected to the local computer is to set
‘the value of the ServerName property to an empty string.

for each oPrinter in oMaster.Printers("\\server")
        wscript.echo "PrinterName     : " & oPrinter.PrinterName
next

‘The following code uses the Err object to determine whether the printers were listed successfully.

if Err <> 0 then
        ‘An error occurred
end if
```

Note

If you identify your printer by using another property instead of PrinterName, you can use any of the Printer object properties on the oPrinter object in the sample code above.

Getting the Default Printer

The sample code in this section creates any required objects and then lists the default printer on the local computer only.

```
dim oMaster

‘The following code creates the required PrintMaster object.

set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code lists the default printer on the local computer.

wscript.echo oMaster.DefaultPrinter

‘The following code uses the Err object to determine whether the default printer was listed successfully.

if Err <> 0 then
        ‘An error occurred
end if
```

Setting the Default Printer

The sample code in this section creates any required objects and then sets the default printer on the local computer only.

```
dim oMaster

‘The following code creates the required PrintMaster object.

set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code sets the default printer on the local computer.

oMaster.DefaultPrinter = "my cool printer"

‘The following code uses the Err object to determine whether the default printer was updated successfully.

if Err <> 0 then
        ‘An error occurred
end if
```

Saving Printer Settings to a File (Persist Save)
The sample code in this section creates any required objects and then saves the printer settings to a file.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code sets some of the flags listed in Table 4.
Flags = kPrinterInfo2 + kPrinterSec

'The following code saves the settings to a binary file that contains all the data specified by flags.
oMaster.PrinterPersistSave PrinterName, FileName, Flags

'The following code uses the Err object to determine whether the settings were saved to a file successfully.
if Err <> 0 then
        'An error occurred
end if


'The following code displays the custom HRESULTs as returned by PrinterPersistSave:
Failed to write Printer data because writing failure 0x80040002
Failed to store Printer Info 2 because writing failure 0x80040005
Failed to store Printer Info 2 because GetPrinter failure 0x80040006
Failed to store Printer Info 7 because writing failure 0x80040009
Failed to store Printer Info 7 because GetPrinter failure 0x8004000a
Failed to store Printer Security Descriptor because writing failure 0x8004000d
Failed to store Printer Security Descriptor because GetPrinter failure 0x8004000e
Failed to store Printer Color Profiles because writing failure 0x80040011
Failed to store Printer Color Profiles because EnumcolorProfiles failure 0x80040012
Failed to store User DevMode because writing failure 0x80040015
Failed to store User DevMode because GetPrinter failure 0x80040016
Failed to store Printer DevMode because writing failure 0x80040019
Failed to store Printer DevMode because GetPrinter failure 0x8004001a
```

Table 4 lists the flags defined in Persist.vbs that you can use when saving printer settings to a file. You can use any combination of the flags listed in Table 4; however, it would not make sense to use the MinimumSettings or the AllSettings flags with any other flags.

Table 4. Flags for Saving Printer Settings

Flag	Value	Description
const kPrinterData	1	Printer Data
const kPrinterInfo2	2	PRINTER_INFO_2
const kPrinterInfo7	4	PRINTER_INFO_7
const kPrinterSec	8	Security descriptor
const kUserDevmode	16	User Devmode
const kPrinterDevmode	32	Printer Devmode
const kColorProf	64	Color Profile
const kMinimumSettings	35	kPrinterData + kPrinterInfo2 + kPrinterDevmode
const kAllSettings	127	kMinimumSettings + kPrinterInfo7 + kPrinterSec + kUserDevmode + kColorProf

Restoring Printer Settings from a File (Persist Restore)
The sample code in this section creates any required objects and then restores the printer settings from a file. There are a few different scenarios you can follow when restoring settings from a file. You can restore all saved settings, or you can, as in this sample, restore only a few of the saved settings. However, you can only restore information that was saved in the settings file. Attempting to restore a setting that was not saved to the file will result in an error.

```
dim oMaster

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code sets some of the flags listed in Table 5.
Flags = kPrinterInfo2 + kPrinterSec + kResolveName

'The following code saves the settings to a binary file that contains all the data specified by flags.
oMaster.PrinterPersistRestore PrinterName, FileName, Flags

'The following code uses the Err object to determine if the settings were restored to a file successfully.
if Err <> 0 then
        'An error occurred
end if


'The following code displays the custom HRESULTs as returned by PrinterPersistRestore:
Failed to restore Printer data because SetPrinterData failed 0x80040003
Failed to restore Printer data because reading failure 0x80040004
Failed to restore Printer Info 2 because reading failure 0x80040007
Failed to restore Printer Info 2 because SetPrinter failure 0x80040008
Failed to restore Printer Info 7 because reading failure 0x8004000b
Failed to restore Printer Info 7 because SetPrinter failure 0x8004000c
Failed to restore Printer Security Descriptor because reading failure 0x8004000f
Failed to restore Printer Security Descriptor because SetPrinter failure 0x80040010
Failed to restore Printer Color Profiles because reading failure 0x80040013
Failed to restore Printer Color Profiles because AddColorProfile failure 0x80040014
Failed to restore User DevMode because reading failure 0x80040017
Failed to restore User DevMode because SetPrinter failure 0x80040018
Failed to restore Printer DevMode because reading failure 0x8004001b
Failed to restore Printer DevMode because SetPrinter failure 0x8004001c
Failed because of unresolved printer name conflict 0x8004001d
Failed because of printer name conflict 0x8004001e
Restoring failure because failure at building backup info 0x8004001f
Restoring failure and Backup Failure, too; printer settings in undefined status 0x8004ffff
```

Note

The printer that is being restored from a file must exist and have the same driver as the one that was used to save the printer settings.

Table 5 lists the flags defined in Persist.vbs that you can use when restoring printer settings from a file. You can only use the following combination of the flags:

kForceName, kResolvePort

kResolveName, kResolvePort, kResolveShare, or kDontGenerateShare

Table 5. Flags for Restoring Printer Settings from a File

Flag	Value	Description
const kForceName	128	Forces the printer and its share name to be renamed. The new name will be set to the same name used in the settings file.

const kResolveName	256	Restore all saved settings except for printer name and share name.
const kResolvePort	512	Ignores the port name specified in the file. This helps in the following cases: • If you store the settings for a printer on one computer and then restore the settings for the printer on a different computer. • If the original printer had a custom port, it might not exist on the second computer. By specifying this flag, the port from the file will be ignored. • On a Terminal Server, at log off, the settings of the printer are stored in a file, and the printer and the port are deleted. At logon, the printer is recreated, but with a different port.
const kResolveShare	1024	Assigns the printer a share name that does not conflict with other share names on the computer.
const kDontGenerateShare	2048	Ignores the share name.

Getting a Printer's Registry Data

The sample code in this section creates any required objects, and then gets the configuration data for a printer and a print server from their keys in the registry.

Note

The code in this sample provides a wrapper around the spooler's GetPrinterDataEx API. Before using the following code, make sure you understand the values used in the registry keys. Using this code inappropriately can result in wrong data being written into the registry and unwanted behavior of your spooler and printers. For more information about the values for all of the printer and server keys, see the Prndata.vbs script in the %Drive%:\Program Files\Windows Server 2003 Resource Kit directory and the Windows Server 2003 Platform SDK topic "GetPrinterDataEx."

```
dim oMaster
dim PrinterData

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code gets printer data from the registry. The arguments it uses are a printer name, a key
'name, and a value name. For more information about the values for all of the printer and server keys,
'see the Prndata.vbs script.
PrinterData = oMaster.PrinterDataGet("MyPrinter", "MyKey", "MyValue")

'The following code gets the data for a print server from the registry. The arguments it uses are a server
'name, which must use double backslashes (\\); a key name, which must be empty quotes ("") for servers;
'and a value name.
PrinterData = oMaster.PrinterDataGet("\\server", "", "DefaultSpoolDirectory")

'The following code uses the Err object to determine whether the settings were retrieved from the registry
'successfully. If the code retrieved the data successfully, the return value will be a variant of the type
'retrieved. Use Table 6 to check the return types.
if Err = 0
        success
else
        'An error occurred
end if
```

Table 6. Successful Return Types for Getting Registry Data

Data Type	Description
long	If the data retrieved corresponds to a REG_DWORD in the registry
string	If the data retrieved corresponds to a REG_SZ in the registry
array of strings	If the data retrieved corresponds to a REG_MULTI_SZ in the registry
array of bytes	If the data corresponds to a REG_BINARY in the registry

Setting a Printer's Registry Data

The sample code in this section creates any required objects and then configures the data for a printer by using its keys in the registry.

Note

The code in this sample provides a wrapper around the spooler's SetPrinterDataEx API. Before using the following code, make sure you understand the values involved in setting the registry keys. Using this code inappropriately can result in wrong data being written into the registry and unwanted behavior of your spoolers and printers. For more information about the values for all of the printer and server keys, see the Prndata.vbs script in the %Drive%:\Program Files\Windows Server 2003 Resource Kit directory and the Windows Server 2003 Platform SDK topic "SetPrinterDataEx."

```
dim oMaster
dim PrinterData
dim Var

'The following code creates the required PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

'The following code sets printer data in the registry. The arguments it uses are a printer name, a key name,
'and a variant containing the data you want to set. For more information about the variant, see the section
'below.
oMaster.PrinterDataSet("MyPrinter", "MyKey", "MyValue", Var)

'The following code uses the Err object to determine whether the settings were set in the registry
'successfully..
if Err = 0
        success
else
        'An error occurred
end if
```

To set data in the registry, it is passed to the SetPrinterDataEx API through a variant. This variant must be one of these data types: long, string, array of strings, or array of bytes. Table 7 lists the data types that you can use and describes how they will set values in the registry. For more information about how to build the variant that must be passed as a parameter, see the Prndata.vbs script in the %Drive%:\Program Files\ Windows Server 2003 Resource Kit directory.

Table 7. Data Types Used to Set Registry Data

Data Type	Description

long	Sets a REG_DWORD in the registry.
string	Set a REG_SZ in the registry.
array of strings	Sets a REG_MULTI_SZ in the registry.
array of bytes	Sets a REG_BINARY in the registry.

Driver Objects

A printer driver is software that converts application data to control codes that a printer can understand. A printer driver then sends the control codes to the print manager. It is the print manager that sends the data to the printer by using a communication driver.

Using the Driver Object Interface

The Driver object has fewer properties than the Print object. The Driver object has properties you can read and write, and some you can only read. The following tables identify how you can interact with the property, the property name, its type, and the function for which it is used. This section includes the following:

- Table 8 lists the read and write Driver object properties, data types, and uses.
- Table 9 lists the read-only Driver object properties, data types, and uses.

For more information about each property in the DRIVER_INFO_3 structure, see the Windows Server 2003 Platform SDK topic “DRIVER_INFO_3.”

Table 8. Read and Write Driver Object Properties

Property	Data Type	Use
ModelName	String	Name of the printer model
ServerName	String	Name of the server (or "" for local machine)
DriverArchitecture	String	Specifies the type of processor.
DriverVersion	String	Provides a description of the driver.
InfFile	String	Path and filename of INF to be used to install the driver
Path	String	Specifies a file name or full path and file name for the file that contains the device driver (for example, "C:\Drivers").

Table 9. Read-Only Driver Object Properties

Property	Data Type	Use
Version	Long	Specifies the version number of the driver.
Environment	String	Specifies the environment for which the driver was written (for example, "Windows NT x86", "Windows IA64", or "Windows NT 4.0").
MonitorName	String	Specifies a language monitor (for example, "PJL monitor"). This member can be NULL and should be specified only for printers capable of bidirectional communication.

Managing Printer Drivers

All operations on printer drivers are done via the PrintMaster object. For most of them, the existence of a Driver object is required also. Creating these objects are included in the task instructions.

This section provides sample code to complete the following tasks:

Adding a Printer Driver

Deleting a Printer Driver

Enumerating Printer Drivers

Adding a Printer Driver
The sample code in this section creates any required objects, adds a printer driver, and then uses error handling to determine if the code was successful in adding a printer driver.

```
dim oMaster
dim oDriver

‘The following code creates the required PrintMaster and Driver objects.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set oDriver = CreateObject("Driver.Driver.1")

‘The following code sets the driver name. It is required and the string cannot be empty.
oDriver.ModelName = "drv x800"

‘The following code sets the driver processor. You must use either “Intel” or “Itanium.”
oDriver.DriverArchitecture = "Intel"

‘The following code specifies the operating system version for which the driver was written. If this setting is
‘not configured, the caller's environment is used. You can use one of the following sets of values:
‘"Windows 95, Windows 98, and Windows Millennium Edition"
‘"Windows NT 4.0 or 2000"
‘"Windows 2000, Windows XP and Windows Server 2003"
‘"Windows XP and Windows Server 2003"
odriver.DriverVersion      = "Windows NT 4.0 or 2000"

‘The following code sets the driver cache. This setting is optional.
oDriver.Path = "c:\files"

‘The following code sets the INF file. This setting is optional because if it is not set, the default directory,
‘%windir%\inf\ntprint.inf, is used.
oDriver.InfFile          = "c:\ntprint.inf"

‘The following code sets a server name to add a driver remotely. If you do not specify a ServerName, or if
‘you use empty double quotes (""), the default, which adds a driver to the local computer, is used.
oDriver.ServerName  = "\\server"

‘The following code adds the driver.
oMaster.DriverAdd oDriver

‘The following code uses the Err object to determine whether the driver was created successfully.
if Err <> 0 then
        ‘An error occurred
end if
```

Deleting a Printer Driver
The sample code in this section creates any required objects, deletes a printer driver, and then uses error handling to determine whether the code was successful in deleting the printer driver.

```
dim oMaster
dim oDriver

‘The following code creates the required PrintMaster and Driver objects.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
```

```
set oDriver = CreateObject("Driver.Driver.1")

‘The following code specifies the driver name. It is required and the string cannot be empty.
oDriver.ModelName = "drv x800"

‘The following code sets the driver processor. You must use either “Intel”, or “Itanium”.
oDriver.DriverArchitecture = "Intel"

‘The following code specifies the operating system version for which the driver was written. This setting is
‘required and the string cannot be empty. You must use one of the following sets of values:
‘"Windows 95 or 98"
‘"Windows NT 3.1"
‘"Windows NT 3.5 or 3.51"
‘"Windows NT 4.0"
‘"Windows NT 4.0 or 2000"
‘"Windows 2000"
odriver.DriverVersion      = "Windows NT 4.0 or 2000"

‘The following code sets a server name to delete a driver remotely. If you do not specify a ServerName, or
‘if you use empty double quotes (""), the default, which deletes a driver from the local computer, is used.
oDriver.ServerName  = "\\server"

‘The following code deletes the driver.
oMaster.DriverDel oDriver

‘The following code uses the Err object to determine whether the driver was created successfully.
if Err <> 0 then
        ‘An error occurred
end if
```

Enumerating Printer Drivers

The sample code in this section creates any required objects, lists all printer drivers on a remote computer, and then uses error handling to determine whether the code was successful in listing all of the printer drivers.

```
dim oMaster

‘The following code creates the required PrintMaster and Driver objects.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The following code specifies the name of a computer. The printer driver list will be created based on
‘printers installed on the specified computer. The only way to list printer drivers installed on the local
‘computer is to set the value of the ServerName property to an empty string.
for each oDriver in oMaster.Drivers(“\\server”)

‘The following code gets the driver name.
        wscript.echo "DriverName    : " & oDriver.ModelName

‘The following code gets the numerical driver version.
        wscript.echo "Version       : " & oDriver.Version

‘The following code gets a string description of the driver, for example "Windows 2000".
        wscript.echo "DriverVersion : " & CStr(oDriver.DriverVersion)

‘The following code gets the path where the files are.
```

```
        wscript.echo "DriverPath    : " & oDriver.Path
```

‘The following code gets the environment of the driver, for example “Windows NT x86”.

```
        wscript.echo "Environment   : " & oDriver.Environment
```

‘The following code gets the architecture of the driver, for example “Intel”.

```
        wscript.echo "DriverEnv     : " & oDriver.DriverArchitecture
```

‘The following code gets the monitor name, if any.

```
        wscript.echo "MonitorName   : " & oDriver.MonitorName
next
```

‘The following code uses the Err object to determine whether the drivers were listed successfully.

```
if Err <> 0 then
        ‘An error occurred
end if
```

Port Objects

The Port object can be used to manage local and remote printer ports. A printer port is a port through which a printer can be connected to a personal computer.

Using the Port Object Interface

Table 10. Read and Write Port Object Properties

Property	Data Type	Use
PortName	String	A string that identifies a supported printer port (for example, "LPT1:").
ServerName	String	A string identifying the server that controls the printer. If this string is NULL, the printer is controlled locally.
HostAddress	String	A string identifying the IP address or DNS name of printer.
PortNumber	Long	A number identifying the Port used by the protocol (e.g.TCP RAW; 9100, TCP LPR: 515).
QueueName	String	A string identifying the Queue name of the printer (for TCP LPR).
CommunityName	String	A string identifying the SNMP community name (“public”).
PortType	Long	The type of the port is passed to, and retrieved from, the objects as an integer. const kTcpRaw = 1 const kTcpLPr = 2 const kLocal = 3 const kLprMon = 5 const kHPdlc = 7 const kUnknown = 8
Snmp	BOOL	A true/false indicating device support of SNMP.
SNMPDeviceIndex	Long	A number identifying the SNMP index.
DoubleSpool	BOOL	A true/false indicating whether true byte counting is to be used or not.

Table 11. Read-Only Port Object Properties

Property	Data Type	Use
DeviceType	String	Applies to TCP and LPR MON ports. ex: IBM InfoPrint 20
Description	String	Standard TCP/IP Port, Local Port
MonitorName	String	Local Monitor, TCPMON.DLL

Managing Ports

All operations on printer ports are done via the Port object. This section provides sample code to complete the following tasks:

Adding Printer Ports

Deleting Printer Ports

Adding Printer Ports

The sample code in this section demonstrates how to add a port by using the Port object. The Port object can add Standard TCP RAW, Standard TCP LPR, or Standard Local ports.

```
dim oPort
dim oMaster
set oPort = CreateObject("Port.Port.1")
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘Indicate where to add the port. Double quotes ("" ) stand for the local computer, which is the default.
oPort.ServerName = "\\server"

‘The name of the port cannot be omitted.
oPort.PortName = "IP_1.2.3.4"

‘The type of the port can be 1 (TCP RAW), 2 (TCP LPR), or 3 (standard local).
oPort.PortType = 1

‘Mandatory for TCP ports. This is the address of the device to which the port connects.
oPort.HostAddress = "1.2.3.4"

‘For TCP RAW ports. Default is 9100.
oPort.PortNumber = 9102

‘Enable or disable SNMP.
oPort.SNMP = true / false

‘If SNMP is enabled, 1 is the default for index.
oPort.SNMPDeviceIndex = 2

‘If SNMP is enabled, "public" is the default community name.
oPort.CommunityName = "public"

'Applies to TCP LPR name, default is LPR
oPort.QueueName = "Queue"

‘Byte counting or double spool applies to TCP LPR ports, is disabled by default.
oPort.DoubleSpool = true / false

‘Try adding the port.
oMaster.PortAdd oPort
‘Test for the status.
If Err <> 0 then

        ‘An error occurred.
end if
```

Deleting Printer Ports
The sample code in this section demonstrates how to delete a port by using the Port object. The Port object can delete any type of port.

```
dim oMaster

‘Create the PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The first argument is the server name; can be "" for the local computer. The second argument is the name
‘of a port.
oMaster.PortDel "\\server", "c:\temp\localport.prn"

‘Use the Err object for the status of the operation.
if Err <> 0 then

        ‘An error occurred.

end if
```

Enumerating Printer Ports
The sample code in this section demonstrates how to enumerate a port by using the Port object. Enumerating ports requires a PrintMaster object. The Port object is mandatory only if the explicit option is set.

```
dim oMaster
dim oPort

‘Create the PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘The first argument in parenthesis is the server name. The parenthesis and the server name can be
‘missing, in which case the local computer will provide the collection.

for each oPort in oMaster.Ports("\\server")

‘Name of the port.
wscript.echo "PortName" & oPort.PortName

‘Monitor name if any.
wscript.echo "MonitorName" & oPort.MonitorName

‘Will return the port type string.
wscript.echo "Description" & oPort.Description
next

Use the Err object to check the status of the operation.
if Err <> 0 then
        ‘An error occurred
end if
```

Note:

If you list ports on a remote computer where you do not have administrative credentials, all ports other than local ports and TCP ports will have the PortType property set to the unknown port.

Getting the Configuration of a Printer Port

The sample code in this section demonstrates how to get the configuration of a port by using the Port object. The Port object can get the configuration of Standard TCP, HP DLC, and LPR MON port types.

```
dim oPort
dim oMaster

‘Create the Port object.
set oPort = CreateObject("Port.Port.1")

‘Create the PrintMaster object.
set oPort = CreateObject("PrintMaster.PrintMaster.1")

‘The first argument is the server name; can be "" for the local computer. The second argument is the name
‘of the port. The third argument is a Port object, which will receive the settings of the specified port.

oMaster.PortGet "\\server", "IP_1.2.3.4", oPort

‘Test for success.
if Err = 0 then
        ‘The name of the port.
        wscript.echo "PortName" & oPort.PortName

        ‘The type of the port. “Description” is a function in Portmgr.vbs that converts a number that
        ‘represents a port type to a string.
        wscript.echo "PortType" & Description(oPort.PortType)

        ‘The address of the device to which it connects.
        wscript.echo "HostAddress" & oPort.HostAddress

        ‘The name of the queue (applies to LPR ports).
        wscript.echo "QueueName" & oPort.QueueName

        ‘Applies to TCP RAW ports.
        wscript.echo "PortNumber" & CStr(oPort.PortNumber)

        ‘Check if SNMP is enabled.
        if oPort.SNMP then
                ‘The SNMP device index.
                wscript.echo "SNMP Index" & CStr(oPort.SNMPDeviceIndex)

                ‘The community name.
                wscript.echo "Community" & oPort.CommunityName
        end if
        if oPort.DoubleSpool then

                ‘Byte counting is enabled.

        else
```

```
                ‘Byte counting is disabled.
        end if
end if
```

Note:

Valid fields for each port type:

SNMPDeviceIndex and CommunityName are present only if SNMP is enabled.

- TCP RAW: PortName, PortType, HostAddress, PortNumber, SNMP
- TCP LPR: PortName, PortType, HostAddress, QueueName, DoubleSpool, SNMP
- HP DLC: PortName, PortType, HostAddress
- LPR MON: PortName, PortType, HostAddress, QueueName

Setting the Configuration of a Printer Port

The sample code in this section demonstrates how to set the configuration of a Standard TCP port by using the Port object. It is recommended that you get the configuration of the port, then set any of the properties listed below, and then update the port.

```
dim oPort
dim oMaster

‘Create the Port object.
set oPort = CreateObject("Port.Port.1")

‘Create the PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘Get the current configuration. The first argument is a server name; "" indicates local computer. The second
‘one is a port name; the third argument is a Port object, which will receive the settings of the specified port.
oMaster.PortGet "\\server", "IP_1.2.3.4", oPort

‘Set the new type: 1 (RAW) or 2 (LPR). An Err object may be used here to test the result of the get
'operation.
oPort.PortType = kTcpRaw

‘Set the address of the device to which it connects.
oPort.HostAddress = "2.3.4.5"

‘This applies to RAW ports only.
oPort.PortNumber = 9100

‘This applies to LPR ports only.
oPort.QueueName = "Queue"

‘Enable/disable SNMP.
oPort.SNMP = true

‘This applies if SNMP is enabled, SNMP device index.
oPort.SNMPDeviceIndex = 1
```

```
‘This applies if SNMP is enabled, SNMP community name.
oPort.CommunityName = "public"

Enable/disable byte counting (double spool).
oPort.DoubleSpool = true / false

‘Try updating the data.
oMaster.PortSet oPort

‘Error handling.
if Err <> 0 then

‘An error occurred
End if
```

Converting Printer Ports

The sample code in this section demonstrates how to convert an existing LPR MON port into a TCP port.

The only supported conversion is from LPR to TCP. The PortConversion method will query the device and Tcpmon.ini for the desired settings—such as protocol type (RAW or LPR), queue name, and port number—on that device. You can use the settings in the Port object passed as a parameter and add that corresponding port.

If the device is not responding, the Port object will be configured with default settings. You will know if the device did not respond from the fact that oPort.DeviceType will be empty. The only situation in which this read-only property is *not* empty is if the device responded and the device type could be identified.

```
dim oPort
dim oMaster

‘Create the Port object.
set oPort= CreateObject("Port.Port.1")

‘Create the PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘Set the IP address of the device.
oPort.HostAddress = "1.2.3.4"

‘The only supported flag is kLprToTcp.

oMaster.PortConversion oPort, kLprToTcp

‘Error handling.
if Err <> 0 then

‘An error occurred
End if
```

Forms Objects

This section provides more detail on using the object interface, methods for printer forms. In the Managing Forms section, sample code is provided for how to add, delete, and enumerate form objects.

Using the Forms Object Interface

Table 12 Read and Write Forms Object Properties

Property	Data Type	Use
ServerName	String	A string identifying the server that controls the printer. If this string is NULL, the printer is controlled locally.
Name	String	A string that specifies the name of the form.
Flags	Long	0 (user), 1 (builtin), or 2 (printer)

Methods

```
GetSize(Height as Variant, Width as Variant)
SetSize(Height as Variant, Width as Variant)
GetImageableArea(Top, Left, Bottom, Right as Integer)
SetImageableArea(Top, Left, Bottom, Right as Integer)
```

For more information about the Form object, see the Windows Server 2003 Platform SDK topic "FORM_INFO_1."

Managing Forms

The Form object works with dimensions of forms expressed in thousands of millimeters. The sample code below assumes that everything is in thousands of millimeters.

Adding Forms

Deleting Forms

Enumerating Forms

Adding Forms

The sample code in this section demonstrates how to add forms.

```
dim oMaster
dim oForm

'Create a PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
```

```
‘Create a Form object.
set oForm = CreateObject("Form.Form.1")

‘Set the form name.
oForm.Name = "MyFavoriteForm"

‘Set the server name where the form will be added.
oForm.ServerName = "\\server"

‘Set the size of the dimensions of the form.
oForm.SetSize iHeight, iWidth

‘Set the imageable area of the form. The coordinates are relative to the upper left corner of the form
oForm.SetImageableArea iTop, iLeft, iBottom, iRight

‘Try adding the form.
oMaster.FormAdd oForm

‘Test the result of the operation.
if err <> 0 then

‘An error occurred.
end if
```

Deleting Forms
The sample code in this section demonstrates how to delete forms.

```
dim oMaster
dim oForm

‘Create the PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘Create a Form object.
set oForm = CreateObject("Form.Form.1")

‘Set the name of the form to be deleted.
oForm.Name = "MyForm"

‘Set the server name where the form will be deleted from.
oForm.ServerName = "\\server"

‘Try deleting the form.
oMaster.FormDel oForm

‘Test the result.
if err <> 0 then

‘An error occurred.
end if
```

Enumerating Forms
The sample code in this section demonstrates how to enumerate forms.

```
dim oMaster
dim oForm

‘Create a PrintMaster object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")

‘Enumerate the forms on \\server. This string can be "" or missing, in which case the forms on the local
‘computer will be enumerated.
for each oForm in oMaster.Forms("\\Server")

        ‘Get the size of the form.
        oForm.GetSize iHeight, iWidth

        ‘Get the form's imageable area expressed as coordinate pairs.
        oForm.GetImageableArea iLeft, iTop, iRight, iBottom

        ‘The flags will contain a number identifying the type of the form, built-in, printer, and user.
        wscript.echo oForm.Flags, iHeight, iwidth, iLeft, iTop, iRight, iBottom
next
```

Summary

This paper provides additional information about printer, drivers, ports, and forms to help administrators and programmers manage a large number of local and remote printers. This paper is intended to be used in conjunction with the PrnAdmin DLL, which is available with the Windows Server 2003 Resource Kit Tools. By this point in the white paper, you should have an understanding of how to install the PrnAdmin DLL and register it. You should also have enough information from the property and attribute reference information and sample scripts to be able manage printers locally and remotely by using scripts.

Appendix A: Error Handling

The methods and properties of all interfaces return HRESULT codes.

The PrintMaster object implements the ISupportErrorInfo interface to provide rich error information.

Besides returning the HRESULT representing the error code of the action performed, all methods and properties of PrintMaster call FormatMessage and SetErrorInfo. This is to provide the scripting client with an error object that include a string description of the error.

If this were not the case, the scripting client would not be always able to set a description for the error code returned by a method of an interface. This means that for all interfaces other than IPrintMaster, Err.Description might not contain a string (that is, it might be empty).

Appendix B: Sample Scripts

This appendix briefly describes the scripts that are installed with Prnadmin.dll in the Resource Kit directory.

Clean.vbs
Use this script to delete all printing objects—forms, drivers, printers, and ports—except Built-in forms.

Clone.vbs
Use this script to clone Windows 2000 print servers. It generates four scripts and a batch file:

- *ComputerName*_drv_clone.vbs
- *ComputerName*_form_clone.vbs
- *ComputerName*_port_clone.vbs
- *ComputerName*_prn_clone.vbs
- *ComputerName*_install.bat

The batch file launches all four of the scripts. You can choose to clone only certain printing objects—ports, for example—by running the associated script without invoking the batch file. The drivers and ports must be installed before printers can be installed.

The script will not be able to clone:

- Out-of-box drivers (drivers not shipped with Windows 2000).
- Ports other than local ports and Standard TCP ports.
- Printers that use the drivers or ports mentioned above.

When the driver-cloning script is run, it might require user intervention. When you attempt to install an out-of-box driver, a dialog box is displayed that asks for a path to the driver. In most cases, you will choose **Cancel** to dismiss the dialog box and skip that driver.

Conall.vbs
Use this script to add printer connections to all printers on the specified server. The server cannot be the local computer.

Defprn.vbs
Use this script to get or set the default printer for the logged-on user. The script works only on the local computer.

Drvmgr.vbs
Use this script to add a driver. Adding a driver requires only a model name. The default architecture is the caller's architecture. The default version is "Windows 2000". For installing an out-of-box driver, you can specify the path to the .inf file for that driver and the path to the driver files.

To delete a driver, specify its model name, architecture, and version.

Forms.vbs
Use this script to add or enumerate forms by using either inches or centimeters as units. The script mimics the spooler's API in terms of passing parameters. This means that the UI for adding forms works slightly different than this script. The UI shows the coordinates of the imageable area of a form relative to the upper left corner and the lower right corner. When a adding a form by using the script, the coordinates of the imageable area need to be specified relative to the upper left corner of the form.

Persist.vbs
Use this script to save printer settings to a file and restore printer settings from that file.

Portconv.vbs
Use this script to:

- List device settings, for example **portconv.vbs -g -i "device name or IP address"** If the device does not respond to the query, a list of default properties will be displayed.
- Add a TCP port corresponding to a LPR MON port. For example, **portconv.vbs -a –p "1.2.3.4:Queue"** will make an attempt to query the device type and preferred settings. If the device does not respond to the query, a TCP LPR port will be added with default settings.
- Add, for all LPR MON ports from a server, corresponding TCP ports on a destination server. For example, **portconv.vbs -w -c \\source -d \\destination**

Note: The LPR ports will not be deleted. Ensure that printers will use the new TCP ports that you added, and then delete the old LPR MON ports.

Portmgr.vbs
Use this script to add, delete, enumerate, get, and set the configuration of a port. The following limitations apply:

- Only local ports or Standard TCP ports can be added.
- A TCP RAW port must have a device to connect to and a port number, usually 9100. SNMP can be enabled or disabled.
- A TCP LPR port must have a device to connect to and a queue. Double spool (LPR byte counting) can be enabled or disabled. SNMP can be enabled or disabled.
- The script can delete any type of port, including LPTx:, COMx:, FILE:, and NUL.
- The script can get the configuration of TCP, LPR MON, and HP DLC ports. This requires administrative credentials on the computer where the ports are.
- The script can set the configuration of TCP ports only.

Prncfg.vbs
Use this script to get or set the configuration of a printer. It includes a routine for converting the attribute number to a string.

Prnctrl.vbs
Use this script to control a printer. It can pause, resume, purge, or send a test page.

Prndata.vbs
Use this script to read or write to the printer's key in the registry. You can also use it to read or write print server data.

Prnmgr.vbs
Use this script to add, delete, or enumerate printers. When you add a printer and the printer driver is not present, the driver will be installed. If the printer driver is present, it will not be overwritten.

You can use this script to delete all local printers, or all printer connections, on the local computer.

Related Links

See the following resources for further information:

- Microsoft Platform Software Development Kit (SDK)
  http://www.microsoft.com/msdownload/platformsdk/sdkupdate/?MSCOMTB=ICP_Platform%20SDK%20home

For the latest information about Windows Server 2003, see the Windows Server 2003 Web site at http://www.microsoft.com/windowsserver2003.