Printer Administration Objects

1. Purpose

This program was initially meant to be a thin wrapper upon printui.dll. It had to expose the functionality of the previously mentioned library to scripting clients. Now, besides this, the tool provides more functionality, like adding, deleting and configuring TCP print ports.

This *Microsoft® Windows® 2000 Resource Kit* tool can do the following:

-1 Adding/deleting printers without UI (local or remote)
-2 Adding/deleting printer connections
-3 Adding/deleting forms (local or remote)
-4 Adding/deleting ports (standard local ports, TCP LPR/RAW ports) (local or remote)
-5 Adding/deleting drivers without UI (remote or local)
-6 Enumerating ports, drivers, printers, forms (remote or local)
-7 Printer control (pause, resume, purge) (local or remote)
-8 Printer configuration (share, rename, RAW only, and so on) (local or remote)

2. Requirements

2.1 Functional Requirements

The goal is to provide the user large scale non-interactive printer operability. The user will be able to manage printers on both the local computer and most importantly on remote computers.

2.2 End User Requirements.

PrnAdmin consists of sever COM objects residing in PrnAdmin.dll. The program must be installed using the following command: **regsvr32 [Path]PrnAdmin.dll**.

The user needs to know a scripting language (VBS, JS) or can make use of the objects provided by PrnAdmin in any programming language that supports COM Programming.

The tool runs only on Microsoft® Windows® 2000. However, all operations except adding and deleting ports can be done remotely to computers running Microsoft® Windows NT® version 4.0

2.3 Dependencies

The COM objects are directly dependent on Printui.dll. The code that handles ports is included in the tool.

3. Design

3.1 Introduction

The functionality of the tools is exposed through objects.

There are 7 types of objects: PrintMaster, Port, Printer, Driver, PortCollection, PrinterCollection and DriverCollection

The PrintMaster is the object whose methods must be called to perform most of the operations. It acts on the other type of objects.

The existence of the PrintMaster object is necessary for handling Collections of other objects.
Port, Printer and Driver objects could be auto-sufficient, that is, store some data in them and then call a method like Add or Delete. However this is not the way it works. The tool uses the PrintMaster object to centralize message handling. So the common scenario will be the following:

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

oPort.PortName = "paper.prn"
…
oPM.PortAdd oPort
…
For Each p in oMaster.Ports
…
Next
```

3.2 The Objects

3.2.1 The Printer Object

3.2.1.1 The Printer Interface

Read / Write Properties

```
PrinterName            as String
ServerName             as String
ShareName              as String
DriverPath             as String
PortName         as String
DriverName             as String
InfName          as String
Comment          as String
Location         as String
SepFile          as String
PrintProcessor   as String
DataType         as String
Priority         as Integer
DefaultPriority  as Integer
StartTime              as Integer
UntilTime              as Integer
```

Read-Only Properties

```
Parameters       as String
Attributes       as Integer
Status    as Integer
Jobs             as Integer
AveragePPM       as Integer
```

Set-Only properties

```
AttributeString as String           - used for setting attributes like RAWOnly
etc.
NewName   as String          - used for renaming a printer
Queued           as bool
Direct           as bool
Default   as bool
Shared           as bool
Hidden           as bool
Network   as bool
Local            as bool
EnableDevq       as bool
KeepPrintedJobs as bool
DocompleteFirst as bool
WorkOffline  as boo
EnableBidi       as bool
RawOnly      as bool
Published    as bool
```

For information about the attributes, please refer to the SDK documentation about PRINTER_INFO_2.

3.2.1.2 Handling printers

All operations on printers are done via the PrintMaster object. For most of them, the existence of a printer object is required also.

Adding a printer.

This operation requires a PrintMaster object and a Printer object. The following code explains the how to add a printer.

```
dim oPrinter
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set oPrinter = CreateObject("Printer.Printer.1")
The name of the server where to add the printer. Specify "" for the local
computer. This is the default if this property is not set. The name of the server
must always include \\.
oPrinter.ServerName  = \\server
This cannot be an empty string.
oPrinter.PrinterName = "my printer"
This cannot be an empty string.
oPrinter.DriverName  = "a driver x100"
This string cannot be empty.
oPrinter.PortName    = "lpt1:"
This is optional. By default, the drivers will be picked up from the driver cache
directory.
oPrinter.DriverPath  = "c:\drivers"
This is optional. The default is %windir%\inf\ntprint.inf
oPrinter.InfFile     = "c:\winnt\inf\ntprint.inf"
Adding the printer
oMaster.PrinterAdd oPrinter
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

If you want to configure other properties of the printer, like attributes or comment for example, you need to create a printer object and then call PrintMaster's method PrinterSet.

Deleting a Printer.

This operation requires a PrintMaster object. The following code explains the how to delete a printer.

```
dim oMaster
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
The name of the server where to delete the printer is the first argument. Specify
"" for the local computer. The name of the server must always include \\.
oMaster.PrinterDel ServerName, PrinterName
or
oMaster.PrinterDel "", PrinterName
Use the Err object provided by VBS for error handling
```

Adding a printer connection.

This operation requires a PrintMaster object. The following code explains the how to add a printer connection.

```
dim oMaster
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
Add the connection. The printername below must be of the form: \\server\my printer. It
can specify a printer or a share name. The connection will be added on the local
computer and is a per user resource.
oMaster.PrinterConnectionAdd PrinterName
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Deleting a Printer Connection.

This operation requires a PrintMaster object. The following code explains the how to delete a printer connection.

```
dim oMaster
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
The ServerName property will be ignored, since the printer connection is per
user.
Delete the connection. The printername below must be of the form: \\server\my printer.
It can specify a printer name, not a share name.
oMaster.PrinterConnectionDel PrinterName
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Getting the configuration of a printer

This operation requires a Printer object and a PrintMaster object.

```
dim oPrinter
dim oMaster
set oPrinter = CreateObject("Printer.Printer.1")
set oMaster  = CreateObject("PrintMaster.PrintMaster.1")
example of getting the settings of local printer
oMaster.PrinterGet "","My Printer", oPort
example of getting the settings for a remote printer
oMaster.PrinterGet \\server, "My Printer", oPort
another example for remote printer
oMaster.PrinterGet "", "\\server\My Printer", oPort
if the operation succeeded then print out the configuration.
if Err = 0 then
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
       Please see the prncfg.vbs script that comes with the tool to see the
       meaning of the bits of     Attributes
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
```

The status is a number. The bits of that number represent the status of a printer, like out of paper, paused etc.

The StartTime specifies the earliest time at which the printer will print a job. This value is

expressed as minutes elapsed since 12:00 AM GMT (Greenwich Mean Time); the UntilTime specifies the latest time at which the printer will print a job. These values are expressed as minutes elapsed since 12:00 AM GMT (Greenwich Mean Time).

Please refer to the SDK and to the script Prncfg.vbs for details.

Setting the configuration of a printer

This operation requires a Printer object and a PrintMaster object.

```
dim oPrinter
dim oMaster
create the printer object.
set oPrinter = CreateObject("Printer.Printer.1")
create the Master object.
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
The following operation is not required, but it is advisable to do it. This will
retrieve the information about a printer and set all the properties of the
printer object. If not called, when setting the configuration, all properties for
which the user didn't enter values will be written with defaults. This may very
likely cause unwanted effects.
oMastr.PrinterGet "\\server", "MyPrinter", oPrinter
If the user wants to set the configuration of a printer without previously
calling a PrinterGet, then they must set oPrinter.ServerName = \\server if the set
will occur for a remote printer
Otherwise, PrinterGet will fill in the Servername property.
The user may or may not set any of the following properties.
A new port name
oPrinter.PortName = "lpt:1"
new share name. this will be effective only if the shared attribute is set to
true.
oPrinter.ShareName = "my share"
location string. May be empty in which case the printer will have no location
information.
oPrinter.Location = "my office"
location string. May be empty in which case the printer will have no comment.
oPrinter.Comment = "my good printer"
can be RAW, EMF, TEXT. This property cannot be set to an empty string.
oPrinter.DataType = Data
a new name for the printer
oPrinter.NewName = NewName
separator file
oPrinter.SepFile = "c:\sep-file"
```

```
the following are printer attributes. Can be set to true or false in order to
enable or disable.
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
update settings
oMaster.PrinterSet oPrinter
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Pausing a Printer

This operation requires a PrintMaster object.

```
dim oMaster
creating the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
a remote printer can be specified in two ways:
oMaster.PrinterPause \\server, "MyPrinter"
or
oMaster.PrinterPause "", \\server\MyPrinter
pausing a local printer
oMaster.PrinterPause "", "MyLocalPrinter"
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Resuming a Printer

This operation requires a PrintMaster object.

```
dim oMaster
creating the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
a remote printer can be specified in two ways:
oMaster.PrinterResume \\server, "MyPrinter"
or
oMaster.PrinterResume "", \\server\MyPrinter
resuming a local printer
oMaster.PrinterResume "", "MyLocalPrinter"
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Purging a printer

```
dim oMaster
creating the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
a remote printer can be specified in two ways:
oMaster.PrinterPurge \\server, "MyPrinter"
or
oMaster.PrinterPurge "", \\server\MyPrinter
purging a local printer
oMaster.PrinterPurge "", "MyLocalPrinter"
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Printing a test page

This operation requires a PrintMaster object.

```
dim oMaster
creating the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
a remote printer can be specified in two ways:
oMaster.PrintTestPage \\server, "MyPrinter"
or
oMaster.PrintTestPage "", \\server\MyPrinter
sending a test page to a local printer
oMaster.PrintTestPage "", "MyLocalPrinter"
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Enumerating printers

This operation requires a PrintMaster object. The following code explains the how to enumerate printers.

```
dim oMaster
```

```
dim oPrinter
create a PrintMaster object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set a server name for listing remote printers. Printer connections will be listed
only if the ServerName property is set to the empty string. They won't be listed
even if the string represents the local computer.
The server name is an optional argument; this means a printer collection can be
enumerated using oMaster.Printers. It will get the printers on the local computer
for each oPrinter in oMaster.Printers("\\server")
      the user may use all the properties of the printer object. Methods like set
      can be applied on the oPrinter object
wscript.echo "ServerName      : " & oPrinter.PrinterName
end if
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Getting the default printer

This requires a PrintMaster object to be created. Applies only to the local computer.

```
dim oMaster
creating the PrintMaster object
set oPrint = CreateObject("PrintMaster.PrintMaster.1")
getting the default printer
wscript.echo oPrint.DefaultPrinter
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Setting the default printer

This requires a PrintMaster object to be created. Applies only to the local computer.

```
dim oMaster
creating the PrintMaster object
```

```
set oPrint = CreateObject("PrintMaster.PrintMaster.1")
setting the default printer
oPrint.DefaultPrinter = "my cool printer"
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Saving printer settings to a file (Persist Save)

This operation is done via the PrintMaster object.

```
dim oMaster
create the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set the flags; an explanation of the flags is below
Flags = kPrinterInfo2 + kPrinterSec
try saving the settings to a file
oMaster.PrinterPersistSave PrinterName, FileName, Flags
check the error status
if Err <> 0 then
   an error occurred
end if
```

There are several flags defined in persist.vbs:

Flags for saving settings:

```
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)
```

The user may choose any combination of the flags above. Obviously, the flag for all settings and the flag for minimal settings shouldn't be combined with any other flags.

A binary file will be created which contains all the data specified by flags.

Restoring printer settings from a file (Persist Restore)

This operation is done via the PrintMaster object.

```
dim oMaster
create the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
set the flags; an explanation of the flags is below
Flags = kPrinterInfo2 + kPrinterSec + kResolveName
try restoring the settings from a file
oMaster.PrinterPersistRestore PrinterName, FileName, Flags
check the error status
if Err <> 0 then
   an error occurred
end if
```

There are several flags defined in persist.vbs in addition to the ones for saving the settings, used for restoring the settings of a printer:

```
const kForceName        = 128
const kResolveName      = 256
const kReslovePort      = 512
const kResolveShare     = 1024
const kDontGenerateShare= 2048
```

The user can restore only partial information from the file. Ex: if a printer_info_2 and a security descriptor were saved, the user may choose to restore only the printer_info_2. Anyway, attempting to restore something that was not saved to the file will result in an error. The printer whose settings will be restored from a file must exist and have the same driver as the one that was saved.

There are several conflicting situations that occur when the printer name, share name or port name saved in the file differs from the one to be restored.

-9 use **kForceName** to force the printer and its share name to be renamed to the ones saved in the file

-10 use **kResolveName** to restore the settings except for printer name and share name

-11 use **kResolvePort** to skip the port name specified into file. This helps when, for example, the user stores settings for a printer on a computer and restores them on a 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.
Another example: on Terminal Server, at log off, the settings of the printer are stored into a file and the printer and the port are deleted. At log on, the printer is recreated, but with a different port. **kResolvePort** becomes mandatory in order to be able to restore printer settings.
In the case when print processor is changed between a storing and a restoring, by default print processor specified in file is ignored.

-12 use **kResolveShare** so that the printer will get a share name that is not conflicting with other share names on the computer

-13 use **kDontGenerateShare** to skip the share name

These flags can be combined together: (kForceName, kResolvePort), (kResolveName, kResolvePort, kResolveShare or kDontGenerateShare)

PrinterPersistSave and PrinterPersistRestore return custom HRESULTs as follows:

```
Failed to write Printer data because writing failure 0x80040002
Failed to restore Printer data because SetPrinterData failed 0x80040003
Failed to restore Printer data because because reading failure 0x80040004
Failed to store Printer Info 2 because writing failure 0x80040005
Failed to store Printer Info 2 because GetPrinter failure 0x80040006
Failed to restore Printer Info 2 because reading failure 0x80040007
Failed to restore Printer Info 2 because SetPrinter failure 0x80040008
Failed to store Printer Info 7 because writing failure 0x80040009
Failed to store Printer Info 7 because GetPrinter failure 0x8004000a
Failed to restore Printer Info 7 because reading failure 0x8004000b
Failed to restore Printer Info 7 because SetPrinter failure 0x8004000c
Failed to store Printer Security Descriptor because writing failure 0x8004000d
Failed to store Printer Security Descriptor because GetPrinter failure 0x8004000e
Failed to restore Printer Security Descriptor because reading failure 0x8004000f
Failed to restore Printer Security Descriptor because SetPrinter failure
0x80040010
Failed to store Printer Color Profiles because writing failure 0x80040011
Failed to store Printer Color Profiles because EnumcolorProfiles failure
0x80040012
Failed to restore Printer Color Profiles because reading failure 0x80040013
Failed to restore Printer Color Profiles because AddColorProfile failure
0x80040014
Failed to store User DevMode because writing failure 0x80040015
Failed to store User DevMode because GetPrinter failure 0x80040016
Failed to restore User DevMode because reading failure 0x80040017
Failed to restore User DevMode because SetPrinter failure 0x80040018
Failed to store Printer DevMode because writing failure 0x80040019
Failed to store Printer DevMode because GetPrinter failure 0x8004001a
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
```

-14 Getting and setting printer data

The tool provides a wrapper around the spooler's APIs: GetPrinterDataEx and SetPrinterDataEx.

Using GetPrinterdataEx, one can get the configuration data for a printer or a print server. The function retrieves the configuration data from the printer's or the servers key in the registry.

```
dim oMaster
dim PrinterData
create the master object
oMaster = CreateObject("PrintMaster.PrintMaster.1")
the arguments of the function are:
a printer name or a server name
a key name, can be any string chosen by the user for printers and must be
"" for servers
a value name; please refer to the prndata.vbs script for all the values
under each of the printer's keys and for servers
Example No 1
PrinterData = oMaster.PrinterDataGet("MyPrinter", "MyKey", "MyValue")
Example No 2
PrinterData = oMaster.PrinterDataGet("\\server", "",
"DefaultSpoolDirectory")
check the error code
if Err = 0
success
else
an error occurred
end if
```

This function will return a variant of long, string, array of strings or array of bytes type. The user must check the type of the variant and use it accordingly. The demo script provides code for doing this.

On success, the function will set the return value to a variant of the type:

-15 long, if the data retrieved corresponds to a REG_DWORD in registry

-16 string, if the data retrieved corresponds to a REG_SZ in registry
-17 array of strings, if the data retrieved corresponds to a REG_MULTI_SZ in registry
-18 array of bytes, if the data corresponds to a REG_BINARY in registry.

Using SetPrinterdataEx, one can set the configuration data for a printer or a print server. The function sets the configuration data under the printer's or the servers key in the registry.

```
dim oMaster
dim PrinterData
dim Var
create the master object
oMaster = CreateObject("PrintMaster.PrintMaster.1")
the arguments of the function are:
a printer name or a server name
a key name, can be any string for printers and must be "" for servers
a value name; please refer to the prndata.vbs script for all the values
under each of the printer's keys and for servers
a variant containing the data to be set
oMaster.PrinterDataSet("MyPrinter", "MyKey", "MyValue", Var)
check the error code
if Err = 0
success
else
an error occurred
end if
```

The data to be set is passed to the method through a variant. This variant must be of type: long, string, array of strings or array of bytes. A long will be used to set a RED_DWORD in the registry, a string will be user to set a REG_SZ in the registry, an array of strings will be used to set a REG_MULTI_SZ and an array of bytes to set REG_BINARY data.
The prndata.vbs script provides examples of how to build the variant that needs to be passed as parameter.

The user must understand the meaning of the values set. Using these wrapper functions inappropriately may result in wrong data being written into the registry and wrong behavior of the spooler/printers.

Please refer to the SDK for more information about SetPrinterDataEx and GetPrinterDataEx.

-19 3.2.2 The Driver Object

3.2.2.1 The Driver Interface

Read / Write Properties

```
ModelName              as String
ServerName             as String
DriverArchitecture     as String
DriverVersion   as String
InfFile         as String
Path                   as String
```

Read Only

```
Version   as Long
Environment     as String
MonitorName     as String
```

3.2.2.2 Handling drivers

Adding a printer driver

This operation requires a PrintMaster object and a Driver object.

```
dim oMaster
dim oDriver
create the PrintMaster object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
create the driver object
set oDriver = CreateObject("Driver.Driver.1")
this propert must be set, cannot be empty
oDriver.ModelName          = "drv x800"
this must be one of the following:
Intel, Alpha, MIPS, PowerPC
oDriver.DriverArchitecture = "one of the above"
this must be one of the following. If omitted, the caller's environment will be
used
Windows 95 or 98 | Windows NT 3.1 | Windows NT 3.5 or 3.51 | Windows NT 3.51 |
Windows NT 4.0 | Windows NT 4.0 or 2000 | Windows 2000
odriver.DriverVersion      = "one of the above"
this is optional. The default is the driver cache
oDriver.Path               = "c:\files"
this is optional. The default is %windir%\inf\ntprint.inf
oDriver.InfFile            = "c:\ntprint.inf"
use a server name to add a driver remotely. "" indicates the local computer,
which is the default
```

```
oDriver.ServerName          = \\server
try adding the driver
oMaster.DriverAdd oDriver
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Deleting a printer driver

This operation requires a PrintMaster object and a Driver object.

```
dim oMaster
dim oDriver
create the PrintMaster object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
create the driver object
set oDriver = CreateObject("Driver.Driver.1")
this property must be set, cannot be empty
oDriver.ModelName           = "drv x800"
this must be one of the following:
Intel, Alpha, MIPS, PowerPC
oDriver.DriverArchitecture = "one of the above"
this must be one of the following. It cannot be omitted
Windows 95 or 98 | Windows NT 3.1 | Windows NT 3.5 or 3.51 | Windows NT 3.51 |
Windows NT 4.0 | Windows NT 4.0 or 2000 | Windows 2000
odriver.DriverVersion       = "one of the above"
use a server name to delete a driver from a remote computer. "" indicates the
local computer, which is the default
oDriver.ServerName          = \\server
try deleting the driver
oMaster.DriverDel oDriver
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

Enumerating printer drivers

This operation requires a PrintMaster object

```
dim oMaster
dim oDriver
creating the PrintMaster object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
the property that enumerates drivers take an optional parameter for server name
if it is missing, the drivers on the local computer will be enumerated
for each oDriver in oMaster.Drivers("\\server")
   get the driver name
wscript.echo "DriverName    : " & oDriver.ModelName
get a number as driver version
wscript.echo "Version       : " & oDriver.Version
get a string description of the driver. Ex "Windows 2000"
          wscript.echo "DriverVersion : " & oDriver.DriverVersion
   get the path where the files are
          wscript.echo "DriverPath    : " & oDriver.Path
   environment of the driver ex: Windows NT x86
          wscript.echo "Environment   : " & oDriver.Environment
   architecture of the driver, ex Intel
          wscript.echo "DriverEnv     : " & oDriver.DriverArchitecture
   monitor name, if any
          wscript.echo "MonitorName   : " & oDriver.MonitorName
next
use the Err object to get the status
if Err <> 0 then
   an error occurred
end if
```

3.2.3 The Port Object

3.2.3.1 The Port Interface

Read / Wrtie Properties

```
PortName         as String
ServerName             as String
HostAddress            as String
PortNumber             as Long
QueueName              as String
CommunityName    as String
PortType         as Long
Snmp                   as BOOL
SNMPDeviceIndex  as Long
DoubleSpool            as BOOL
```

Read-Only properties

```
DeviceType             as String (Applies to TCP and LPR MON ports. ex: IBM
InfoPrint 20)
Description            as String (Standard TCP/IP Port, Local Port)
MonitorName            as String (Local Monitor, TCPMON.DLL)
```

3.2.3.2 Handling ports

This tool can add the following types of ports: Standard TCP, RAW and LPR, and Standard Local.
It can delete any type of port.
It can get the configuration of Standard TCP, HP DLC and LPR MON ports.
It can set the configuration of Standard TCP ports only.
The type of the port is passed to and retrieved from the objects as integers.
Here is the association of port type and numbers.

```
const kTcpRaw   = 1
const kTcpLPr   = 2
const kLocal    = 3
const kLprMon   = 5
const kHPdlc    = 7
const kUnknown  = 8
```

Adding ports

```
dim oPort
dim oMaster
create the port object
set oPort = CreateObject("Port.Port.1")
create the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
indicate where to add the port. "" stands 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, 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 the SNMP is enabled, 1 is 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 ports

Deleting ports doesn't need a port object.

```
dim oMaster
create the master object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
the first argument is the computer name; "" for the local computer
the second one is the name of a port
oMaster.PortDel "\\server", "c:\temp\localport.prn"
uses the Err object for the status of the operation
if Err <> o then
   an error occurred
end if
```

Enumerating ports.

This requires a PrintMaster object. The port object is mandatory only if option explicit is on.

```
dim oMaster
dim oPort
create the PrintMaster object
set oMaster = CreateObject("PrintMaster.PrintMaster.1")
the 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
can be "Standard Local" or "Standard TCP/IP" etc.
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 don't have administrator privileges, all ports different from local ports and TCP ports will have the PortType property set to the unknown port. Please refer to Portmgr.vbs to see how to get extended information about ports

Getting the configuration of a port

This applies only to Standard TCP ports, LPR MON and HP DLC ports.

```
dim oPort
dim oMaster
create the port object
set oPort = CreateObject("Port.Port.1")
create the master 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 is a port object which will get the settings of the port
oMaster.PortGet "\\server", "IP_1.2.3.4", oPort
if succeeded
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 which displays converts a number
      which 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:

TCP RAW: PortName, PortType, HostAddress, PortNumber, SNMP
TCP LPR: PortName, PortType, HostAddress, QueueName, DoubleSpool, SNMP
SNMPDeviceIndex and CommunityName are present only is SNMP is enabled
HP DLC: PortName, PortType, HostAddress,
LPR MON: PortName, PortType, HostAddress, QueueName

Setting the configuration of a port

Applies to Standard TCP ports only.
It is advisable to 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 master object
set oMaster = CreateObjec("PrintMaster.PrintMaster.1")
get the current configuration
the first argument is a server name; "" stands for the local computer.
the second one is a port name; the third argument is a port object which will
contain the settings of the current port
oMaster.PortGet "\\server", "IP_1.2.3.4", oPort
an err object may be used here to test the result of the get operation
set the new type: 1- RAW, 2 -LPR
oPort.PortType = 1
setting 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
use an Err object to test the result of the operation
```

Converting ports

The tool can be used to get a corresponding TCP port for an existing LPR MON 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, port number, on that device. The user can use the settings in the Port object passed as parameter and add that corresponding port.
If the device is not responding, the port object will be configured with default settings. The user will be able to know whether the device did not respond from the fact that oPort.DeviceType is empty. This read only property is not empty only 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 master 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
```

3.2.3 The Form Object

3.2.3.1 The Form Interface

Read / Wrtie Properties

```
ServerName              as String
Name            as String
Flags                   as Long
```

Methods

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

3.2.3.2 Handling forms

PrnAdmin works with dimensions of forms expressed in thousands of millimeters. The forms.vbs scripts has several routines that enable the user to work with either inches of centimeters. The code explained below assumes that everything is in thousands of millimeters. Please refer to the script for more detail on how to use inches and centimeters.

Adding forms

```
dim oMaster
dim oForm
create a master 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 top 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
success
else
failure
end if
```

Deleting a form

```
dim oMaster
dim oForm
create the master 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
```

Listing forms

```
dim oMaster
dim oForm
create a master 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, user
please refer to the script forms.vbs for more detail
wscript.echo oFrom.Flags, iHeight, iwidth, iLeft, iTop, iRight, iBottom
next
```

3.2.4 The PrintMaster Object

3.2.4.1 The PrintMaster Interface

Read / Write properties

```
DefaultPrinter  as String
```

Read only Properties

```
Printers        as PrinterCollection
Drivers         as DriverCollection
Ports                 as PortCollection
Forms           as FromCollection
```

They all may take an argument that is a server name. If this is missing, the collection of objects will be retrieved from the local computer.

Methods

```
PrinterAdd(oPrinter)
PrinterDel(server name, printer name)
PrinterGet(server name, printer name, oPrinter)
PrinterSet(oPrinter)
PrintTestPage
PrinterPause
PrinterResume
PrinterPurge

DriverAdd(oDriver)
DriverDel(oDriver)

PortAdd(oPort)
PortDel(server name, port name)
PortGet(server name, port name, oPort)
PortSet(oPort)

PrinterConnectionAdd(printer name)
PrinterConnectionDel(printer name)
PrinterPersistSave(printer name, file name, long - flag)
PrinterPersistSave(printer name, file name, long - flag)

FormAdd(oForm)
FromDel(oFrom)

PortConversion(oPort, long - flag)

PrinterDataGet("printer/server name", "key name", "value name")
PrinterDataSet("printer/server name", "key name", "value name", variant
DataToSet)
```

4 Error handling

The methods and properties of all interfaces return HRESULT codes.
The PrintMaster object implements the ISupportErrorInfo interface in order 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 wouldn't be always able to set a description for the error code returned by a method of an interface. This means for all interfaces other the IPrintMaster it is not guaranteed that Err.Description will contain a string (may be empty).

5 The Scripts

This chapter describes briefly the scripts.

Clean.vbs

This script offers an easy way of deleting (all) printing objects: forms, drivers, printers and ports.
Note: Built-in forms cannot be deleted.

Clone.vbs

How it works:

The script generates several scripts and a batch file.
COMPUTER_drv_clone.vbs
COMPUTER_form_clone.vbs
COMPUTER _port_clone.vbs
COMPUTER _prn_clone.vbs
COMPUTER _install.bat
The string COMPUTER will be the name of the computer to be cloned. The user may choose to clone only certain printing objects, for instance ports.
The batch file will launch all the 4 scripts listed above. The user may choose to run any of them, without invoking the batch file. Note that drivers and ports must be installed before the printers.

This script is designed to clone Windows 2000 print servers. The script won't be able to clone:
out of box drivers (drivers not shipped with Windows 2000)
ports other than local ports and Standard TCP ports
printers than use the drivers or ports mentioned above

When running the generated scripts, the driver cloning script may require the user's intervention. When attempting to install an out of box driver, a UI will show up asking for a path to the driver. In most cases the user will choose cancel to dismiss the UI and skip that driver.

Conall.vbs

This script adds printer connections to all printers on the specified server. This server cannot be the local computer.

Defprn.vbs

This script works only on the local computer . It can get / set the default printer for the logged on user.

Drvmgr.vbs

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

Deleting a driver requires model name, architecture and version to be specified.

Forms.vbs

The user can add / enumerate forms using either inches or centimeters as units. The tool mimics the spooler's API in terms of passing the parameters. This means that the UI for adding forms works slightly different than the tool.
The UI shows the coordinates of the imageable area of a form relative to the top left corner and , respectively, to the bottom right corner.
When a adding a form using the tool, the coordinates of the imageable area need to be relative to the top left corner of the form.

Persist.vbs

Using this script the settings of a printer can be saved to and restored from a file.

Portconv.vbs

This script can do 3 things:
-20 list the device settings, ex: portconv.vbs -g -i "device name or IP address"
If the device does not respond to the query, then a list of default properties will be displayed.
-21 add a TCP port corresponding to a LPR MON port, ex: portconv.vbs -a -p "1.2.3.4:Queue"

In this case, an attempt will be made to query the device about its type and preferred settings. If the device doesn't respond to the query, than a TCP LPR port will be added with default settings

-22 add for all LPR MON port from a server corresponding TCP ports on a destination server,
ex: portconv.vbs -w -c \\source -d \\destination
Note: the LPR ports will not be deleted. The user can take care so that printers will use the new TCP ports added and then delete the old LPR MON ports

Portmgr.vbs

Script for adding, deleting, enumerating ports, getting / setting the configuration of a port.
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 tool can delete any type of port, including LTPx:, COMx:, FILE, NUL.
The tool can get the configuration of TCP, LPR MON and HP DLC ports, but this requires administrator privileges on the computer where the ports are
The tool can set the configuration of TCP ports only.

Prncfg.vbs

Script for getting / setting the configuration of a printer.
The script includes a routine for converting the attribute number to a string.

Prnctrl.vbs

Script for controlling a printer: pause, resume, purge, send a test page.

Prndata.vbs

Script for:
writing/reading data under the printer's key in the registry
writing / reading print server data, ex: the spool directory

Prnmgr.vbs

Script to add, delete, enumerate printers. When adding a printer, if the driver is not present, it will be installed. Otherwise the driver will not be overwritten.
Using this script one can delete all local printer or all printer connections on the local computer.

Prndemo.js

Demo script for how to use the tool with JavaScript clients. The script offers mixed functionality as example.

Using the script one can add printers, list printers and get information about the configuration of a TCP device.

Feedback

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