

How To Create A CUPS Shared Printer In Linux For A Printer That Doesn't Have Adequate Windows Drivers

This tutorial is mostly aimed for people that have older printers, but just don't have adequate drivers for them for newer Windows operating systems. A perfect example is the HP LaserJet 1000 which doesn't have x64 drivers at all. But, this printer works perfectly under Linux x64 or x86. The point is to connect this printer to a Linux rig (any rig will do, even a Raspberry Pi, as long as you can connect the printer to it through USB or a printer port) and share the printer from that rig on the network and use that rig as a print server.

I won't get into installing the printer on the Linux install in great detail, every printer is specific in one way or another (requires this or that library, requires a special proprietary driver/UI, maybe requires compiling something because of license issues, etc.), so I can't get into all of those different examples. I'll do a general walk through for the steps that are more or less the same on every Linux OS. What I can do in detail is explain how to configure the printer so that any Windows OS can print to the Linux print server without needing any special driver at all.

❖ Configuring the Linux print server

There are a few prerequisites that need to be met in order to configure the Linux install as a print server.

1. **CUPS.** The Common UNIX Printing System (CUPS) will be the engine behind the Linux print server. Most Linux distros (distributions, different *flavors* of the same thing basically ☺) have it listed as the `cups` package. To install it in Ubuntu or any other Debian based distro, the terminal command would be `sudo apt-get install cups`. For other distros, install and configure commands may vary, depending on the package manager, service/init manager (systemd or another), etc. After installing CUPS, in systemd, the CUPS service (daemon) will automatically be configured and added to the list of services running on the Linux install. As with most Linux daemons (services), the name of the service is `cupsd`.

If your printer manufacturer doesn't offer Linux drivers or they are older and won't run or get compiled properly on a modern kernel, it's also preferable to install the `cups-filters` package (command: `sudo apt-get install cups-filters` in Ubuntu/Debian based distros). This is just a collection of PPD files that enable CUPS to *talk* to the printer. PPD is short for PostScript Printer Description. CUPS doesn't *speak* a lot of *languages*, but what it can do is speak the PostScript *language*, so with the help of some proprietary binaries or firmwares (binary blobs, sometimes, if the license allows it, included in the `cups-filters` package), it can *talk* using the PostScript *language* to the printer and send it commands to print. If the proprietary binaries and/or firmware are not included in the `cups-filters` package, try searching if your distro has a separate package that installs these binary blobs or try searching for a package from another distro that may contain them and then manually install (copy) them at the appropriate locations. Also, the `cups-filters` package may not contain the adequate PPD file for your printer. In this case, once again, you have to manually search online to see if a package from another distro contains the PPD file you need and/or any binary blobs that are required in order for CUPS to communicate with the printer.

In most cases (especially for ancient printers, Linux has great backwards compatibility), the `cups-filters` package is the only thing you'll need, if your manufacturer doesn't support Linux, so these are just some pointers what to do if for some reason your device is not in the *most common* list of *drivers* (PPD files).

2. **Samba.** Samba is basically a reimplementation of the SMB (Server Message Block) protocol (originally developed by IBM for OS/2, reimplemented by Microsoft in Windows NT 3.1 and later NT based operating systems), as well as other Microsoft specific or proprietary protocols (NetBIOS, WINS, SAM, LSA, NTLM, etc.). These protocols offer file and printer

sharing to clients (computers) in a computer network (such as a local area network). Basically, it was a reverse engineered networking layer, server authentication layer and other protocols developed by Microsoft and other companies. After Microsoft lost two antitrust lawsuits against the Samba team, they were ordered to disclose documents that fully describe the networking protocols developed by Microsoft. So, at the moment, the Samba development team is working with full documentation from Microsoft, meaning the project (Samba) is no longer a reverse engineered effort.

Since most of these old printers do have x86 (32-bit) drivers for Windows (designed for Windows XP, but most of them work even on Windows 7 and later versions of Windows if you manually install them), the simplest solution for the x86 clients would be to just share the printer in the LAN network as a CIFS/SMB printer. Even if the drivers for your printer don't work on anything else but on Windows XP, you'll still need them in order to install the printer on a Windows XP rig. Remember, the goal is to make the printer available and installable on any Windows version you might have in your home/LAN/MAN network.

Most Linux distros will have the Samba package simply listed as `samba` (`sudo apt-get install samba` for Ubuntu and Ubuntu/Debian based distros). On `systemd` based distros, this command will also automatically install the `smbd` and `nmbd` services (daemons) that are responsible for the networking protocols required by Windows. For distros that have other `init` (initialization) systems (not `systemd`), you'll have to refer to your distro's `init` system documentation in order to set these services to automatically start when the rig boots up.

3. **Zero-Configuration Networking (Zeroconf/mDNS/Avahi/Bonjour).** In order to view the Linux print server in the network and use it in DHCP (no static IP or additional settings required in our local network's router/DHCP server) we need to have these zeroconf services (daemons) installed and enabled. Avahi is a free software implementation of the mDNS (Multicast DNS) protocol. Apple's Bonjour software implements the mDNS protocol (as well as other network services).

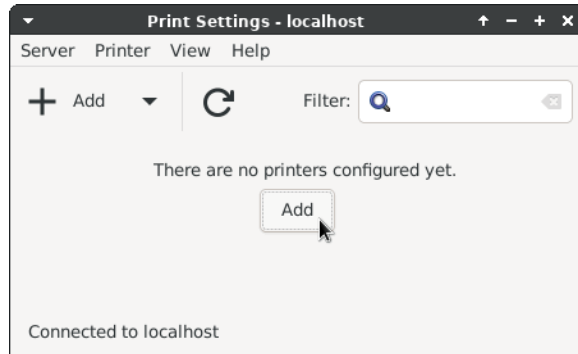
Some Linux distros have the `nss-mdns` package (GNU Name Service Switch - Multicast DNS) in their repositories, which is a plugin for the GNU NSS service, and also contains Avahi and the Avahi daemon (`avahi-daemon`). Depending on your Linux flavor, additional settings might be required (such as editing the `/etc/nsswitch.conf` or similar manual edits to configuration files). Look up your distro's manual pages or Wiki/forums in order to find out more about how to properly configure the Avahi daemon and NSS service to make the Linux print server broadcast itself in the network. For Ubuntu and Ubuntu/Debian based distros, I've found this set of packages usually resolves the problems you might be having when installing and configuring the rig to be browsable in the network: `sudo apt-get install avahi-daemon avahi-discover avahi-utils libnss-mdns mdns-scan`.

On most `systemd` based distros (Ubuntu, Suse, Red Hat, etc.), the package manager automatically configures the services (daemons) and starts them (Arch Linux is an exception). On other distros, you may have to do this manually (add them to the list of services using the system's process supervisor tools). For example, Void Linux uses RunIt as its `init/service` supervisor system. Once a package is installed, if it installs a service as well, the service will be visible as a directory in `/etc/sv`. The service is not automatically started or considered as a service at all, unless you make a symbolic link from that directory located in `/etc/sv` to `/var/service`. So, let's say we want the Avahi service (`avahi-daemon`) to automatically start with the operating system. All we have to do is make a symbolic link from `/etc/sv/avahi-daemon` to the `/var/service` directory (using the command `sudo ln -s /etc/sv/avahi-daemon /var/service`) and the service will start as soon as the symbolic link is created. You can check if the service is running with the command `sudo sv status avahi-daemon`. If it's running, `sv` (RunIt's service manager) will report something like the following: `run: avahi-daemon: (pid 770) 186392s`.

4. **Print Settings.** This is basically a GUI that allows users to manually add printers to their operating system. On most distros, this package is installed automatically and the desktop entry (shortcut) for it resides in Settings (**Settings → Print Settings**). More minimalistic distros, such as Arch or Void, may not have it preinstalled, so you need to manually install the `system-config-printer` package. Some distros may also require root or admin (users that are in the "wheel" group) privileges to install the printer, so it's advisable to invoke the

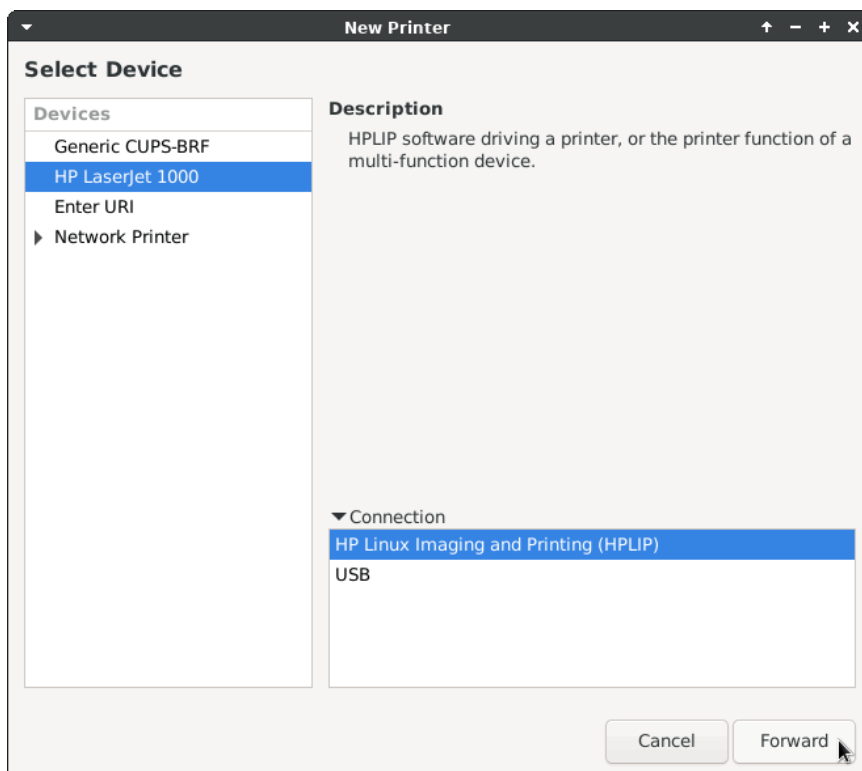
Print Settings GUI from the terminal as root (terminal command: `sudo system-config-printer`).

Now, once we have these prerequisites installed, we can continue to set up the printer. First, we need to install the printer in Linux. We can use either the **Print Settings** UI or we can use the CUPS web UI (<http://localhost:631>, <http://localhost.local:631> or http://<linux_rig_ip>:631). Using either one is fine. In this example, I'll be using the **Print Settings** UI. We invoke it as root by running `sudo system-config-printer`, and when the GUI shows up, we click the **Add** button.

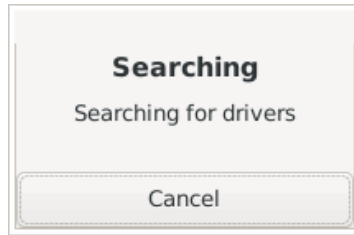


Next, a printer wizard will show up. In this particular example, I'll be adding the HP LaserJet 1000 printer (doesn't have x64 drivers for Windows at all, the latest drivers are x86 only and they do work on Windows Vista, 7, 8, 8.1 and 10, but only on the 32-bit versions), but the same principle applies to any other printer. If your printer has no USB port, only an LPT (printer) port, you might have to add it through the **Enter URI** field. If you feel lost, search online on how to install/add your printer on Linux. In this example, I'll be using the HPLIP (HP Linux Imaging and Printing libraries, which is available for most distros in the repository) as the driver for the printer. The **USB** setting listed on the right works as well (tested) and that uses the generic CUPS driver with the adequate CUPS PPD filter file, but sometimes, the printer doesn't print with this *driver* (filter), so I decided to switch to the proprietary HPLIP driver, which works great.

After we set up the URI (Uniform Resource Identifier, address; in this case, the OS automatically detected the URI, it just asked me for which driver to use on the list of compatible drivers, **HP Linux Imaging and Printing (HPLIP)** or **CUPS** - the **USB** selection), we click the **Forward** button.

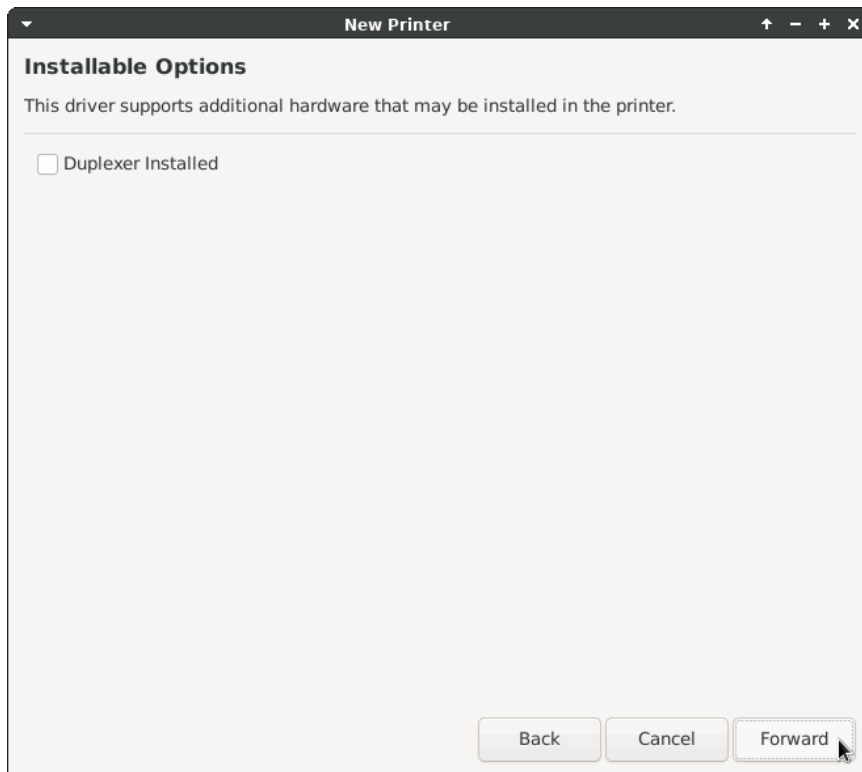


The next screen might show up (depending on how old the hardware on the Linux install is and whether it instantly finds the files it needs or not, LOL ☺). Just let it finish searching for the adequate drivers.



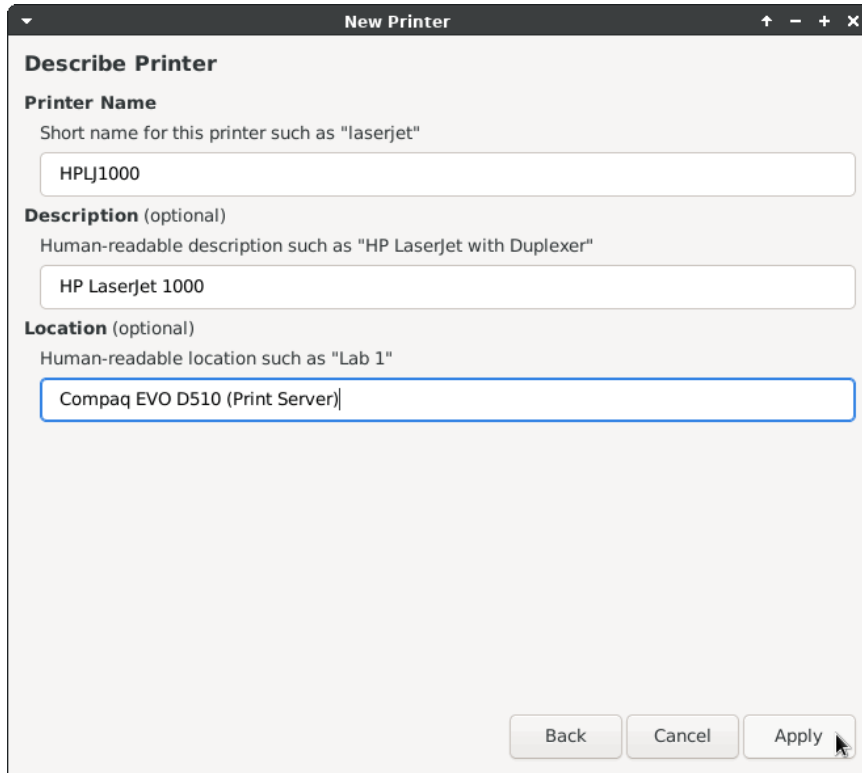
If the wizard doesn't find the adequate *drivers* (PPD filters file), it will ask you to point in which directory the PPD file for your printer is located. If you haven't downloaded one already try searching online to see if anyone from the Linux community has created a PPD file for your printer. Chances are, the first two or three results in the search will contain some forum post with the adequate PPD file as attachment in the post ☺. Place the PPD file in `/usr/share/ppd/<manufacturer>` (in my case `/usr/share/ppd/HP`). If your printer's manufacturer doesn't have a directory in `/usr/share`, create one. If the PPD file is compressed (zip, gz, xz, etc.), uncompress it first, then copy it to `/usr/share/ppd/<manufacturer>`. (terminal command: `sudo copy /path/to/ppd/file.ppd /usr/share/ppd/<manufacturer>`).

The next screen of the wizard might ask you for some printer specific options, as shown in the image below. The HP LaserJet 1000 doesn't have a duplexer, so I'll just leave the check box unticked and click the **Forward** button.



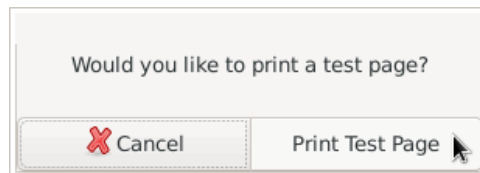
The next screen will ask you for some settings that are related to how the printer will be viewed in the network and some of them (such as the **Printer Name**) can't be changed afterwards through the **Print Settings** UI (you have to remove the printer and then add it again with a different printer name). The **Printer Name** field defines how your printer will be viewed in the network, as well as locally in the Linux install. I like short simple names for the printers, so I usually just use the first letters of the manufacturer and model (in this case, **HPLJ1000**). This makes things easier when you manually have to add the printer in Windows, since in most cases you have to type the IP address or the hostname manually. The **Description** field is optional, so I usually set this field to the real printer's make and model, in this case **HP LaserJet 1000**. The **Location** field is also optional, but I usually put in the hostname or something similar that

describes the PC/laptop which the printer is physically connected with. After we've enter the desired values, we hit the **Apply** button.



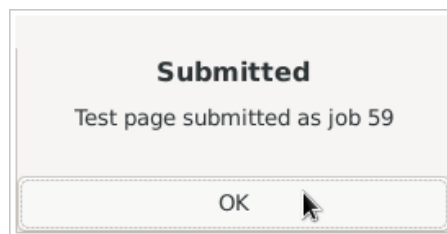
The screenshot shows a window titled "New Printer" with a "Describe Printer" section. It contains three input fields: "Printer Name" with the value "HPLJ1000", "Description (optional)" with the value "HP LaserJet 1000", and "Location (optional)" with the value "Compaq EVO D510 (Print Server)". At the bottom right, there are three buttons: "Back", "Cancel", and "Apply".

The next screen will ask us if we want to print a test page. It's preferable to do so to test out if the printer is working properly.



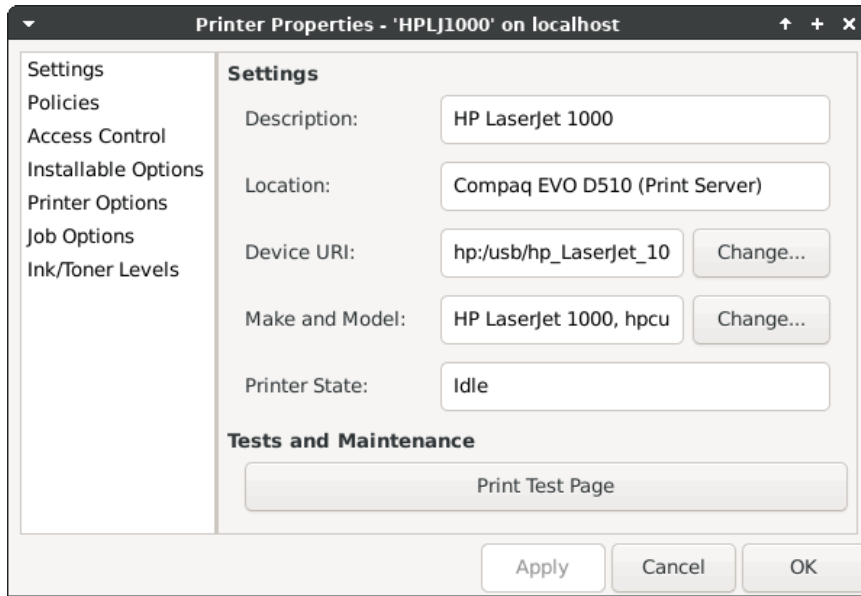
The dialog box contains the text "Would you like to print a test page?" and two buttons: "Cancel" (with a red X icon) and "Print Test Page".

After we hit the **Print Test Page** button, the following screen will show up. If everything's set up OK, your printer will print a test page.

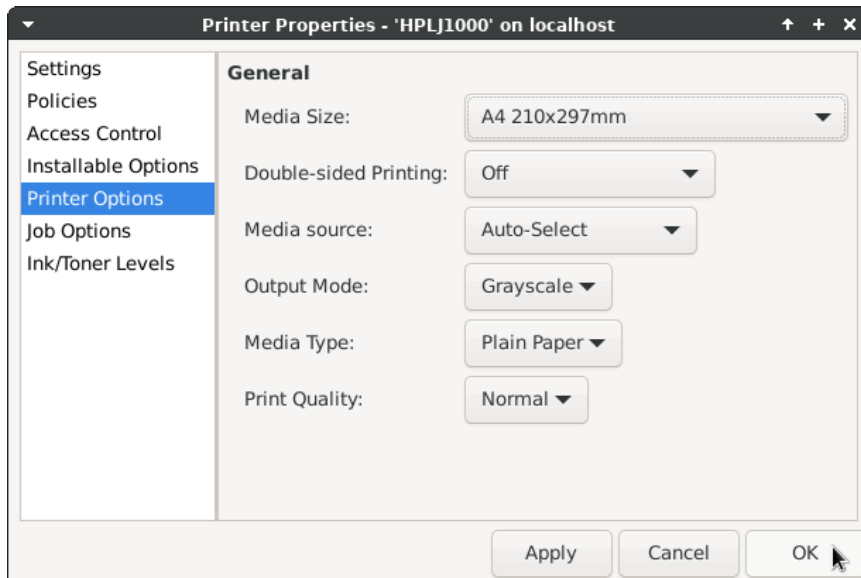


The dialog box displays the text "Submitted" in bold, followed by "Test page submitted as job 59", and an "OK" button at the bottom.

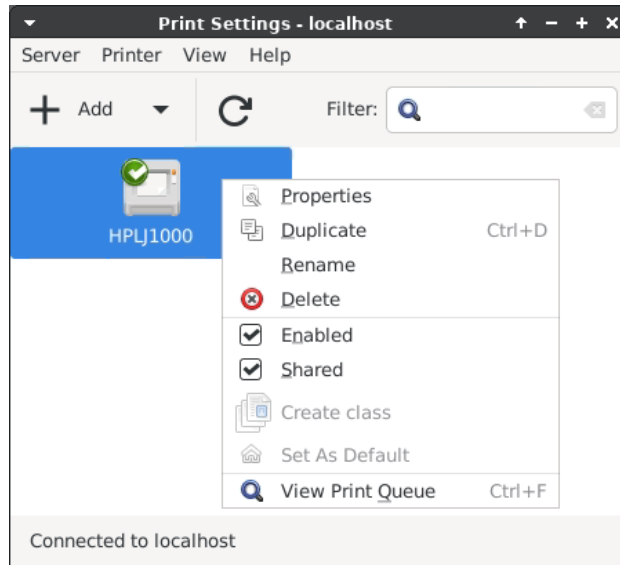
After you hit the **OK** button, the wizard will return you to the **Printer Properties** page. This page is basically the setup page for your printer. Here, you can choose various features, like which driver the printer uses, to which port/URI the printer is connected to, the size of the paper which the printer uses, etc.



The HP LaserJet 1000 is just your basic printer with a USB only interface, so it doesn't have a lot of options, not even a toner level indicator, so I'll just set the paper size in printer properties UI, hit the **Apply** button and then **OK**.

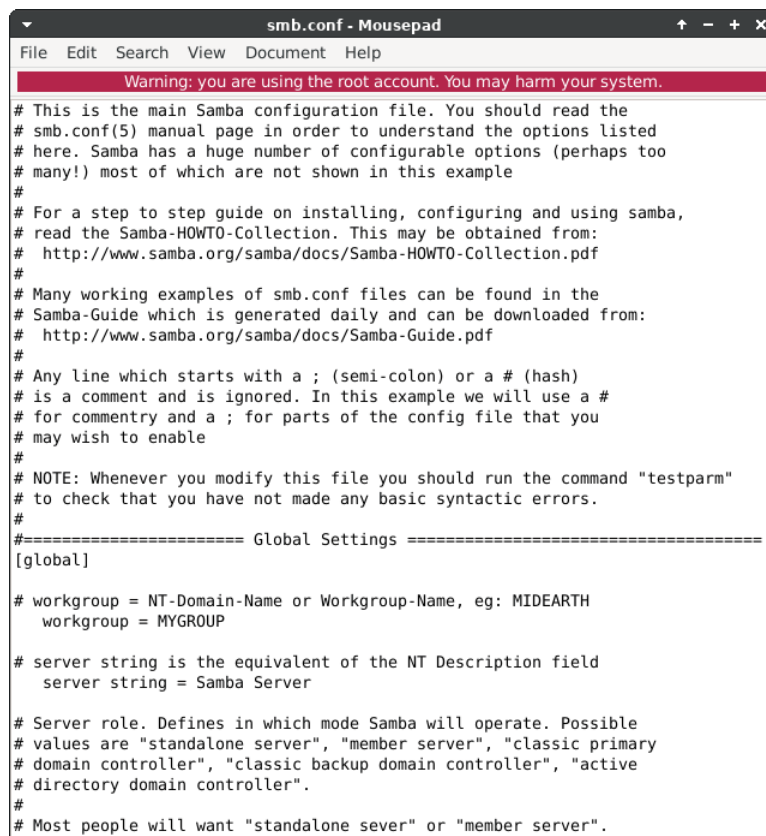


The wizard has finished adding the printer and we can start using it locally (in Linux). Make sure the printer is set up as **Enabled** and **Shared** in the Print Settings UI.



Next, we have to set up Samba and CUPS. Samba makes the printer browsable for the older rigs in our network (Windows XP and maybe Windows Vista/7, x86 only), as well as takes care of the printer handling (*tells* Linux to use CUPS for all printing jobs, instead of its own printing solution).

To configure Samba, we need to edit the `etc/samba/smb.conf` file (on some distros, this file may reside in `/etc` only, `/etc/smb.conf`). First, we'll make a backup of the original (unmodified) Samba settings file using the terminal command `sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.bak`. So, now we'll use our favorite text editor to edit this file. I usually use Mousepad or Nano, but you can use any editor preinstalled with your distro: `sudo <text_editor> /etc/samba/smb.conf`.



First, in the `[global]` section, edit the workgroup name to the one on your Windows installation. By default, Windows sets the workgroup name to "WORKGROUP", so I usually just set it to that.


```
smb.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# For a step to step guide on installing, configuring and using samba,
# read the Samba-HOWTO-Collection. This may be obtained from:
# http://www.samba.org/samba/docs/Samba-HOWTO-Collection.pdf
#
# Many working examples of smb.conf files can be found in the
# Samba-Guide which is generated daily and can be downloaded from:
# http://www.samba.org/samba/docs/Samba-Guide.pdf
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not made any basic syntactic errors.
#
#===== Global Settings =====
[global]

# workgroup = NT-Domain-Name or Workgroup-Name, eg: MIDEARTH
workgroup = WORKGROUP

# server string is the equivalent of the NT Description field
server string = Samba Server

# Server role. Defines in which mode Samba will operate. Possible
# values are "standalone server", "member server", "classic primary
# domain controller", "classic backup domain controller", "active
# directory domain controller".
#
# Most people will want "standalone sever" or "member server".
```

Next, in the `server string` line, write something memorable, something that describes your print server.

```
smb.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# For a step to step guide on installing, configuring and using samba,
# read the Samba-HOWTO-Collection. This may be obtained from:
# http://www.samba.org/samba/docs/Samba-HOWTO-Collection.pdf
#
# Many working examples of smb.conf files can be found in the
# Samba-Guide which is generated daily and can be downloaded from:
# http://www.samba.org/samba/docs/Samba-Guide.pdf
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not made any basic syntactic errors.
#
#===== Global Settings =====
[global]

# workgroup = NT-Domain-Name or Workgroup-Name, eg: MIDEARTH
workgroup = WORKGROUP

# server string is the equivalent of the NT Description field
server string = Compaq EVO D510 (Print Server)

# Server role. Defines in which mode Samba will operate. Possible
# values are "standalone server", "member server", "classic primary
# domain controller", "classic backup domain controller", "active
# directory domain controller".
#
# Most people will want "standalone sever" or "member server".
```

Next, right after the `server role = standalone server` line, add the following lines.


```

server min protocol = LANMAN1
client min protocol = LANMAN1
# lanman auth = yes
ntlm auth = yes
load printers = yes
printing = cups
printcap = cups
printcap name = cups

```

```

smb.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not made any basic syntactic errors.
#
#----- Global Settings -----
[global]
# workgroup = NT-Domain-Name or Workgroup-Name, eg: MIDEARTH
workgroup = WORKGROUP
# server string is the equivalent of the NT Description field
server string = Compaq EVO D510 (Print Server)
# Server role. Defines in which mode Samba will operate. Possible
# values are "standalone server", "member server", "classic primary
# domain controller", "classic backup domain controller", "active
# directory domain controller".
#
# Most people will want "standalone sever" or "member server".
# Running as "active directory domain controller" will require first
# running "samba-tool domain provision" to wipe databases and create a
# new domain.
server role = standalone server
server min protocol = LANMAN1
client min protocol = LANMAN1
# lanman auth = yes
ntlm auth = yes
load printers = yes
printing = cups
printcap = cups
printcap name = cups
# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the "loopback" interface. For more examples of the syntax see

```

I'll explain what each of these settings does.

- `server min protocol = LANMAN1`. This option defines the minimal version of the SMB protocol the server will use. After version 4.x, Samba defaults to SMB2, which is what Windows 7 uses. But, since we'd also like other older Windows versions to be able to access the shared printer, we'll set the `server min protocol` setting to `LANMAN1`, which is what Windows 2000 uses, but Windows XP supports as well.
- `client min protocol = LANMAN1`. This setting defines the minimum protocol version that a client can announce itself to the Samba server. As with the previous setting, we'll set this setting to a Windows 2000 client.
- `lanman auth = yes`. This setting may be deprecated on newer Samba versions (I'm not sure). The LANMAN1 protocol supports unauthenticated login to a print or a file share server, so this setting just requires that all clients using the LANMAN1 protocol authenticate themselves to the server in order to access the server's resources (file and print shares, active directory, etc.). If you're having problems with authentication from older clients (Windows 2000/XP), try and uncomment this line (remove the `#` symbol) restart the services and see if that solves the problem.
- `ntlm auth = yes`. This setting tells Samba to start communicating with a Windows client using the NTLMv1 (NT LAN Manager) protocol, but preferably, negotiate a switch to the NTLMv2 protocol. Since version 4.6.2, Samba starts communication with Windows clients with a minimal version of NTLMv2, since NTLMv1 has some security issues and is somewhat deprecated. But, some older Windows versions use it (namely Windows NT 3.1, but some not up to date Windows 2000 and XP rigs might use it as a default session startup protocol as well), so it's best to leave this setting to `yes`. Try disabling this setting (simply comment out this line, insert a `#` symbol at the beginning of the line) and see if you're having problems

communicating with the Samba printer share. If everything is OK, leave it disabled, if not, revert the setting back (delete the # symbol) and restart the `cupsd`, `nmbd` and `smbd` services.

- `load printers = yes`. This setting completely enables or disables the printer sharing option in Samba. By default, Samba has printer sharing enabled, but, just in case in some future versions of Samba, the development team decides to set the default for this setting to `no`, we'll manually set it to `yes`.
- `printing = cups`. This setting basically *tells* Samba not to handle any print jobs by itself, but instead forward all print jobs to CUPS. Convenient in most cases, since everything is handled from a single end point (the CUPS print server).
- `printcap = cups`. This setting *tells* Samba not to handle any of the printer's capabilities by itself (the `printcap` file), but instead let CUPS handle the printer's capabilities (duplexing, toner levels, toner density, toner save, etc.). The current version of CUPS doesn't need this directive to be defined in the Samba configuration file, but we'll set it to `cups` just in case.
- `printcap name = cups`. Once again, let CUPS do all the hard work ☺. This setting tells Samba to let CUPS handle the file name and location of the `printcap` file.

It's also advisable to completely disable the Home folder shares in Samba (the user's home directory, noted as "User's Files" or "My Documents" in Windows). We'll do this by commenting out the adequate lines. Look for the `[homes]` line in `smb.conf` and add the `;` or `#` symbols in front of all lines in the `[homes]` setting.

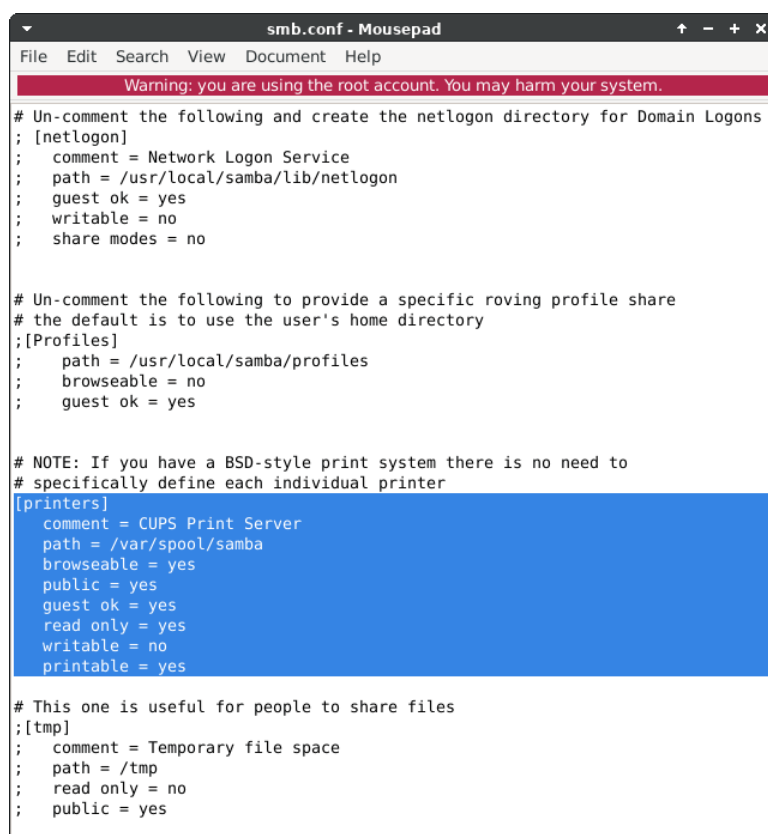
```
;  
;[homes]  
; comment = Home Directories  
; browseable = no  
; writable = yes
```

```
smb.conf - Mousepad  
File Edit Search View Document Help  
Warning: you are using the root account. You may harm your system.  
# at least one WINS Server on the network. The default is NO.  
; wins proxy = yes  
  
# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names  
# via DNS nslookups. The default is NO.  
dns proxy = no  
  
# These scripts are used on a domain controller or stand-alone  
# machine to add or delete corresponding unix accounts  
; add user script = /usr/sbin/useradd %u  
; add group script = /usr/sbin/groupadd %g  
; add machine script = /usr/sbin/adduser -n -g machines -c Machine -d /dev/null %u  
; delete user script = /usr/sbin/userdel %u  
; delete user from group script = /usr/sbin/deluser %u %g  
; delete group script = /usr/sbin/groupdel %g  
  
#===== Share Definitions =====  
;[homes]  
; comment = Home Directories  
; browseable = no  
; writable = yes  
  
# Un-comment the following and create the netlogon directory for Domain Logons  
; [netlogon]  
; comment = Network Logon Service  
; path = /usr/local/samba/lib/netlogon  
; guest ok = yes  
; writable = no  
; share modes = no  
  
# Un-comment the following to provide a specific roving profile share  
# the default is to use the user's home directory  
; [Profiles]  
; path = /usr/local/samba/profiles
```

Next, we get to the printer share part of the `smb.conf` file. Find the `[printers]` line in the `smb.conf` file and edit it in the following way.

```
[printers]  
comment = CUPS Print Server  
path = /var/spool/samba  
browseable = yes
```

```
public = yes
guest ok = yes
read only = yes
writable = no
printable = yes
```



```
smb.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.

# Un-comment the following and create the netlogon directory for Domain Logons
; [netlogon]
;   comment = Network Logon Service
;   path = /usr/local/samba/lib/netlogon
;   guest ok = yes
;   writable = no
;   share modes = no

# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
; [Profiles]
;   path = /usr/local/samba/profiles
;   browseable = no
;   guest ok = yes

# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
comment = CUPS Print Server
path = /var/spool/samba
browseable = yes
public = yes
guest ok = yes
read only = yes
writable = no
printable = yes

# This one is useful for people to share files
; [tmp]
;   comment = Temporary file space
;   path = /tmp
;   read only = no
;   public = yes
```

The [printers] section is a global section that defines the printer share for every printer on the print server. You can also define each of these settings on a per printer basis, but we won't get into that. Now, I'll explain what each of the options do.

- `comment = CUPS Print Server`. You can write anything you want in this section. This setting basically describes the print server's share and you can read this text if you hover over the shared printer's icon in the network.
- `path = /var/spool/samba`. This setting defines the directory which the print server will use a spool directory. The default setting is fine, there's no need to change this.
- `browseable = yes`. This setting defines whether the shared printer can be seen in the network or not. Otherwise, you'd have to manually define the address to the shared printer (`\\computer-hostname\shared-printer-name`).
- `public = yes`. This setting allows or disallows guests to print to that printer. Basically, anyone can print on the printer.
- `guest ok = yes`. A synonym for the setting above (`public = yes`). Some Samba versions have one or the other compiled in the binaries and libraries as a settings, some have both. We'll set both of them just in case.
- `read only = yes`. This setting basically limits users to write to the print server, but only via print spooling operations. Every other kind of write operation is disallowed.
- `writable = no`. A synonym for the setting above (`read only = yes`). Once again, some Samba versions may have binaries compiled with one setting or the other (or both), so just in case, it's best to define them both.

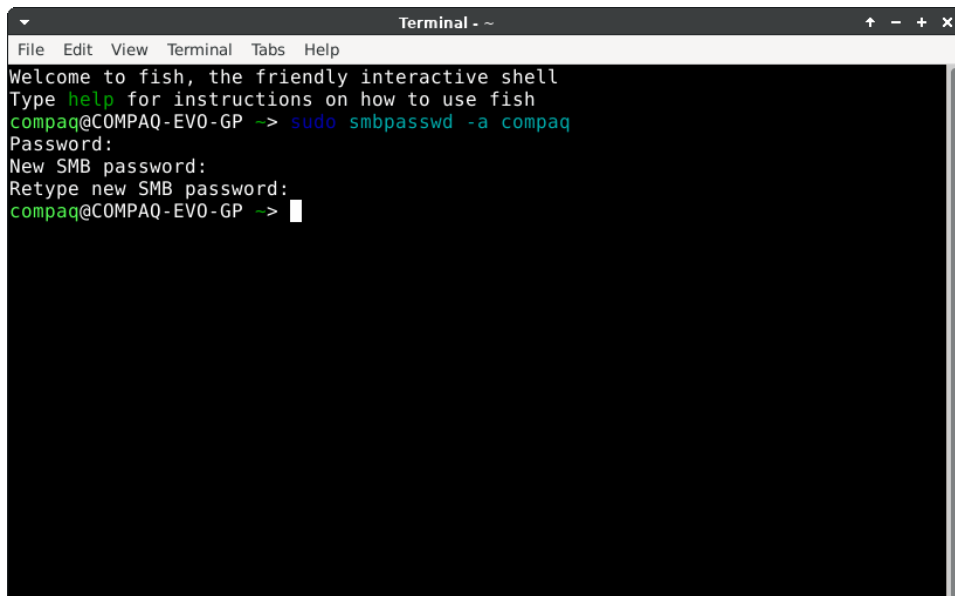
- `printable = yes`. This setting essentially declares the printer service in Samba as enabled, so this setting has to be declared as `yes` (enabled). If declared otherwise, the `smbd` service will refuse to load the `smb.conf` file. Also, Samba uses this setting to differentiate regular file shares from printer shares in the network.

It's also wise to note that Samba's `smbd` and `nmbd` services interpret white spaces in the `smb.conf` file verbatim. This means that `printable =yes` is not the same as `printable = yes`. The former setting will be ignored and the defaults will be loaded (whatever they may be for that particular setting). The latter will be parsed and loaded as a valid setting.

Even though we've allowed guests to print to the printer, it's wise to set an access username and a password for the Linux install. What this basically means is that, users that have the exact address to the printer share (`\\<computer-hostname>\<shared-printer-name>`) and add the printer manually in Windows, will be able to print even if they don't have the adequate credentials to access the PC through the network. But, if you browse the network or if you try to browse the PC through its network address (`\\<computer-hostname>` or `\\<computer_ip_address>`), you will be prompted to enter credentials in order to view the PC's shared resources (printers and/or file shares). It's good to have this failsafe, since anyone can print to the printer (guests are allowed), but if we do this, no one will be able to browse the PC through the network unless they have the adequate credentials (username and password).

And that is it essentially, hit `Ctrl + S` (Save) and save the file ☺.

Now, open a terminal window (**Ctrl + Alt + T** on most Linux distros, some might also have **Win + T** set as a shortcut for the terminal) and type in the following command: `sudo smbpasswd -a <username>`. The username has to be a valid username also defined in the Linux install (for example, the username you defined as your default logon username when you installed Linux). After you hit **Enter**, `smbpasswd` will ask you to define a password for that user (it might ask you for your root password first, LOL ☺). Enter it and repeat it.



```
Terminal - ~
File Edit View Terminal Tabs Help
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
compaq@COMPAQ-EVO-GP -> sudo smbpasswd -a compaq
Password:
New SMB password:
Retype new SMB password:
compaq@COMPAQ-EVO-GP -> |
```

Next, we need to enable the user. Type in the following in the terminal and hit **Enter** afterwards: `sudo smbpasswd -e <username>`.

```
Terminal - ~
File Edit View Terminal Tabs Help
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
compaq@COMPAQ-EVO-GP -> sudo smbpasswd -a compaq
Password:
New SMB password:
Retype new SMB password:
compaq@COMPAQ-EVO-GP -> sudo smbpasswd -e compaq
Enabled user compaq.
compaq@COMPAQ-EVO-GP -> █
```

Now, we have to restart the `smbd` and `nmbd` services in order for the declared directives in the `smb.conf` file to take effect. In Ubuntu and other systemd based distros, the command to restart a service is as follows: `sudo systemctl restart <name_of_service>`. So, in our case, the commands will be `sudo systemctl restart smbd` followed by `sudo systemctl restart nmbd`. (I don't actually know if `systemctl` can restart multiple services in one command, LOL ☺). But, in my example, I'm not using a systemd based distro, I'm using Void Linux, which uses RunIt as a service and `init` (system initialization) manager, so the commands differ. In my case, I'll be using the following command to restart the `smbd` and `nmbd` services: `sudo sv restart smbd nmbd` (`sv` can restart multiple services in one command).

```
Terminal - ~
File Edit View Terminal Tabs Help
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
compaq@COMPAQ-EVO-GP -> sudo smbpasswd -a compaq
Password:
New SMB password:
Retype new SMB password:
compaq@COMPAQ-EVO-GP -> sudo smbpasswd -e compaq
Enabled user compaq.
compaq@COMPAQ-EVO-GP -> sudo sv restart smbd nmbd
Password:
ok: run: smbd: (pid 31422) 0s
ok: run: nmbd: (pid 31426) 1s
compaq@COMPAQ-EVO-GP -> █
```

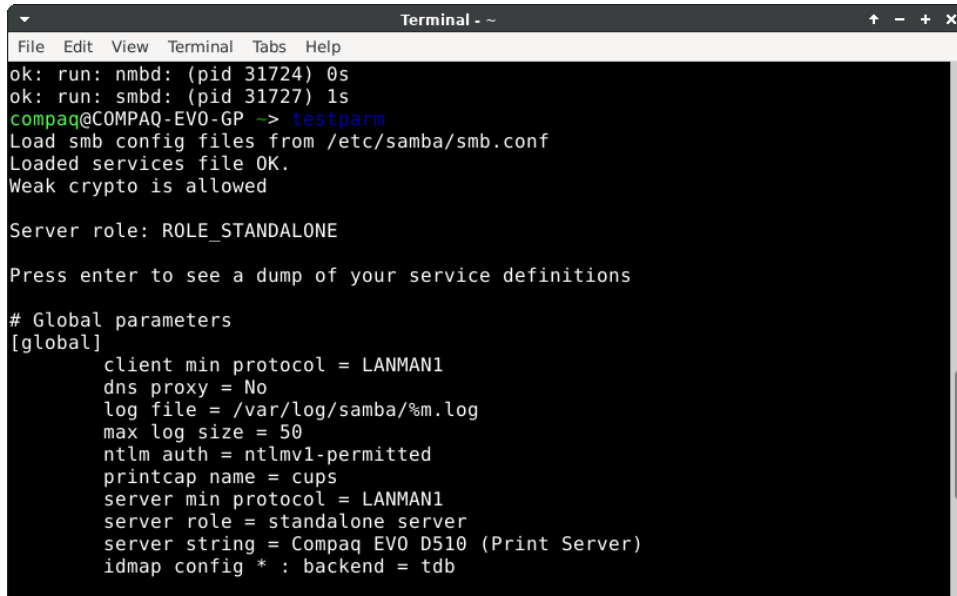
Alternatively, if you don't want to bother manually restarting the services, you just restart the print server and be done with it, LOL ☺.

After restarting the services, test the parameters of the Samba configuration. We can do this with the `testparm` tool that Samba has included. The output of the command should be similar to this.

```
# Global parameters
[global]
  client min protocol = LANMAN1
  dns proxy = No
  log file = /var/log/samba/%m.log
  max log size = 50
  ntlm auth = ntlmv1-permitted
```

```
printcap name = cups
server min protocol = LANMAN1
server role = standalone server
server string = Compaq EVO D510 (Print Server)
idmap config * : backend = tdb
```

```
[printers]
browseable = No
comment = CUPS Print Server
guest ok = Yes
path = /var/spool/samba
printable = Yes
```



```
Terminal - ~
File Edit View Terminal Tabs Help
ok: run: nmbd: (pid 31724) 0s
ok: run: smbd: (pid 31727) 1s
compaq@COMPAQ-EVO-GP -> testparm
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed

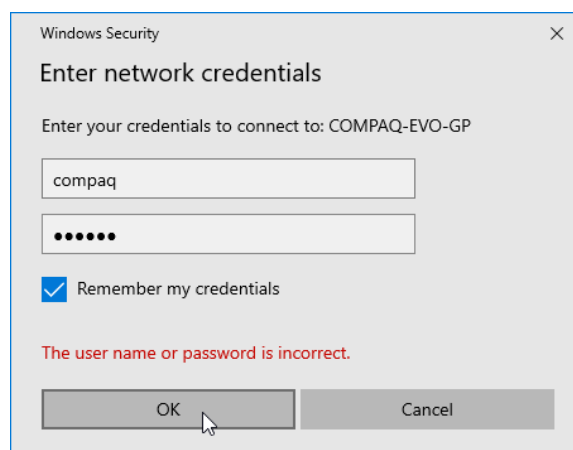
Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

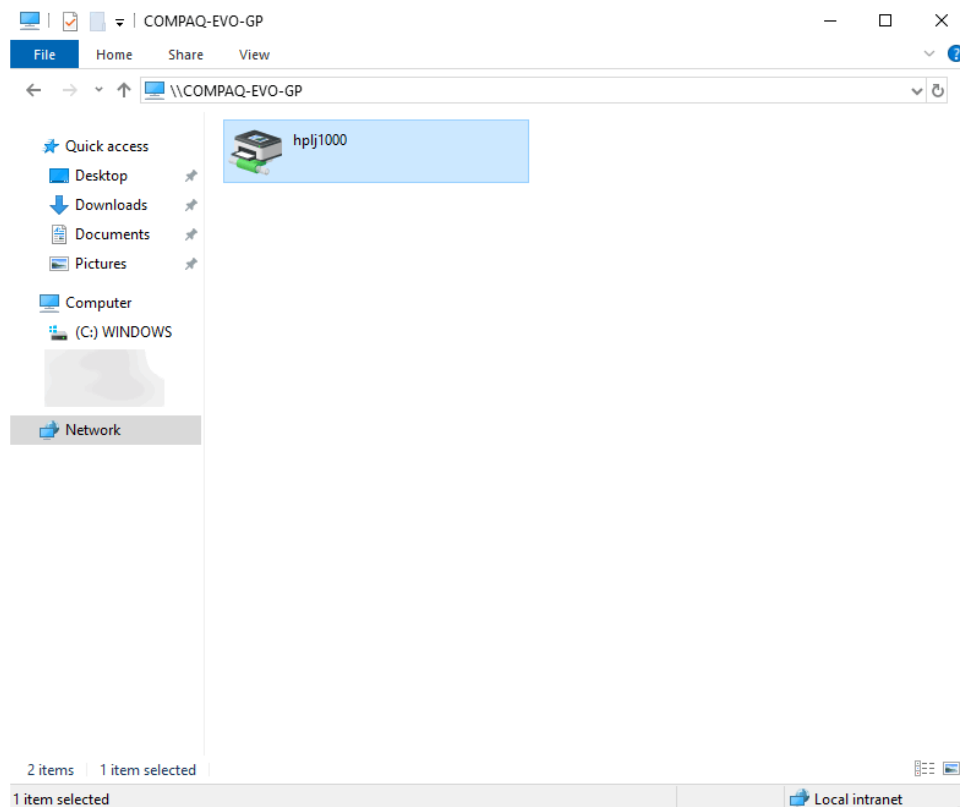
# Global parameters
[global]
    client min protocol = LANMAN1
    dns proxy = No
    log file = /var/log/samba/%m.log
    max log size = 50
    ntlm auth = ntlmv1-permitted
    printcap name = cups
    server min protocol = LANMAN1
    server role = standalone server
    server string = Compaq EVO D510 (Print Server)
    idmap config * : backend = tdb
```

Don't worry about Samba reporting that the printers won't be browseable in the `[printers]` section (`browseable = No`), you will be able to browse them, no worries on that part. I really have no idea why the `testparm` reports that. It reports that on every single print server configuration I've done, and I'm still able to browse the printer in the network, LOL ☺.

Now, we can try and browse the Linux install from the network in Windows. Once you type in the computer's hostname or IP address in the address bar (`\\<computer-hostname>` or `\\<computer_ip_address>`) in Windows Explorer (named File Explorer in Windows 10 and later), a prompt asking you for a username and a password should be presented to you.



Enter the credentials you previously defined with the `smbpasswd` tool (the username and the password). Don't forget to tick the **Remember my credentials** tick box ☺. After you hit **OK**, you should be able to see the shared printer in Explorer.



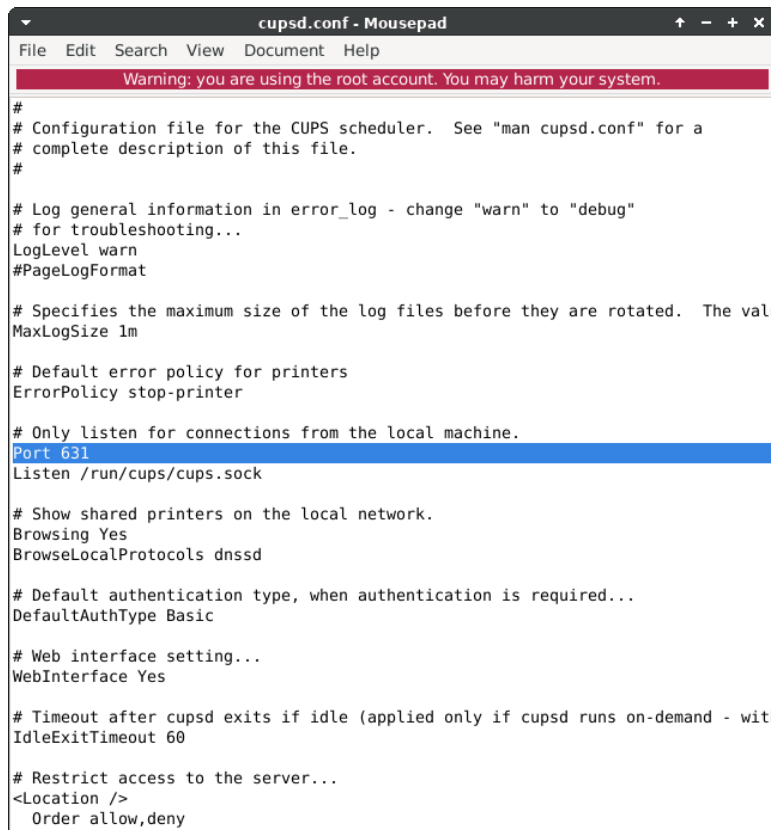
If establishing the connection takes a while, try and disable the `ntlm auth = yes` option in `smb.conf` (insert a `#` symbol at the front) and restart the `smbd` and `nmbd` services. This should resolve the problem.

If you'd like to be able to browse the Linux install through the network with the hostname (not just the IP address), you have to make sure that the Avahi daemon (`avahi-daemon`) is up and running. That's the service responsible for mDNS/ZeroConf/Bonjour. On systemd based distros, the command for checking whether the service is running or not would be `sudo systemctl status avahi-daemon`. If the command replies with a PID somewhere in the response, this means that the service is running 😊. If not, it's probably not running and you'd have to dig into the issue deeper (another service blocking it, maybe a missing library, etc.). If the Avahi daemon is running, but you still can't browse the print server with the print server's hostname, see if a firewall is installed on your Linux print server (Uncomplicated Firewall in Ubuntu – `ufw`). Try to unblock the Avahi service with the firewall's tools (in Ubuntu, type in `sudo ufw app list` in the terminal to get the Avahi daemon's name in the application list, then type `sudo ufw allow "<name_of_avahi_daemon_as_listed_in_the_previous_command>"`).

If you have older Windows x86 installs in your network (Windows 2000/XP/Vista/7), try and connect to the printer. If your manufacturer offers x86 drivers for your printer, try and manually load them (point the printer installation wizard in Windows to look for drivers in a directory or on a CD, as is the case in Windows 2000/XP). If they install, try and print a test page, it should work 😊.

Good, now that we have Samba configured, we can continue and configure CUPS. Configuring CUPS is pretty much the same as configuring Samba, there's one configuration file responsible for all of the options in CUPS. The configuration file is named `cupsd.conf` and is located in `/etc/cups` (on some Linux distros, it may be located in `/etc`). So, we'll just use our favorite text editor to edit this file (`sudo <your_favorite_text_editor> /etc/cups/cupsd.conf`). In my case, that's Mousepad, so the command would be `sudo mousepad /etc/cups/cupsd.conf`.

Find the `# Only listen for connections from the local machine.` line and below it, it'll read `Listen localhost:631`. Change this line to `Port 631`, as shown in the image below.



```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
#
# Configuration file for the CUPS scheduler. See "man cupsd.conf" for a
# complete description of this file.
#
# Log general information in error_log - change "warn" to "debug"
# for troubleshooting...
LogLevel warn
#PageLogFormat

# Specifies the maximum size of the log files before they are rotated. The value
MaxLogSize 1m

# Default error policy for printers
ErrorPolicy stop-printer

# Only listen for connections from the local machine.
Port 631
Listen /run/cups/cups.sock

# Show shared printers on the local network.
Browsing Yes
BrowseLocalProtocols dnssd

# Default authentication type, when authentication is required...
DefaultAuthType Basic

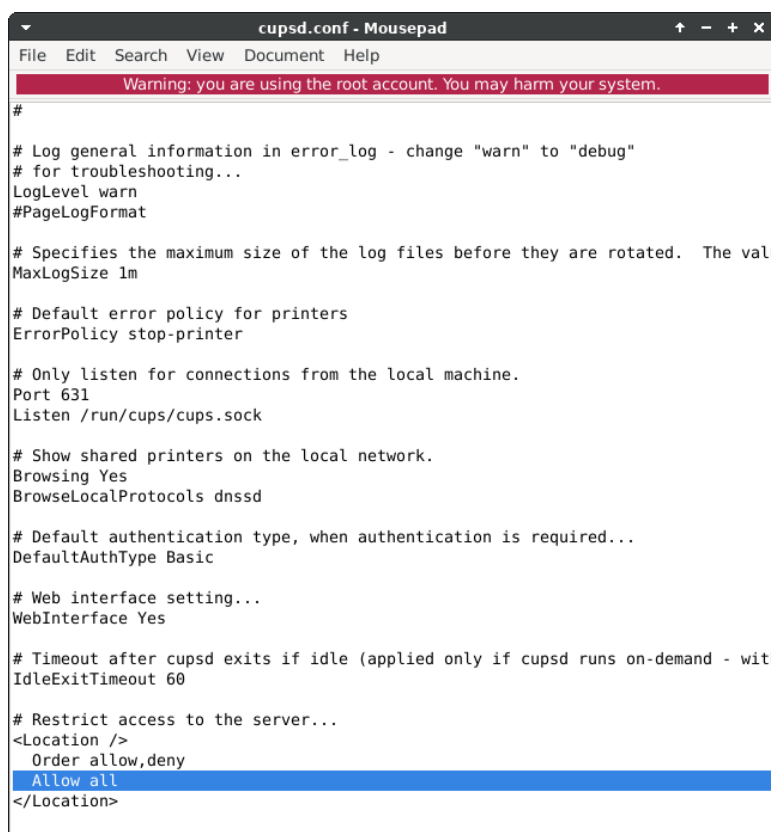
# Web interface setting...
WebInterface Yes

# Timeout after cupsd exits if idle (applied only if cupsd runs on-demand - with
IdleExitTimeout 60

# Restrict access to the server...
<Location />
  Order allow,deny
```

This setting *tells* CUPS to listen for incoming print jobs on HTTP port 631 or IPP (Internet Printing Protocol) port 631, but not just on our local machine (the Linux print server, localhost), but from any machine in our local network (or the internet for that matter). This is a prerequisite and must be allowed, since we'll be using the HTTP/IPP *driverless* method (sort of... it uses a generic driver) in Windows 10 (and above) to print to this print server (i.e. the printer we don't have drivers for).

Next, find the `# Restrict access to the server...` line and add the `Allow all` in a new line after the `Order allow,deny` line, as in the picture below.



```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.
#
# Log general information in error_log - change "warn" to "debug"
# for troubleshooting...
LogLevel warn
#PageLogFormat

# Specifies the maximum size of the log files before they are rotated. The value
MaxLogSize 1m

# Default error policy for printers
ErrorPolicy stop-printer

# Only listen for connections from the local machine.
Port 631
Listen /run/cups/cups.sock

# Show shared printers on the local network.
Browsing Yes
BrowseLocalProtocols dnssd

# Default authentication type, when authentication is required...
DefaultAuthType Basic

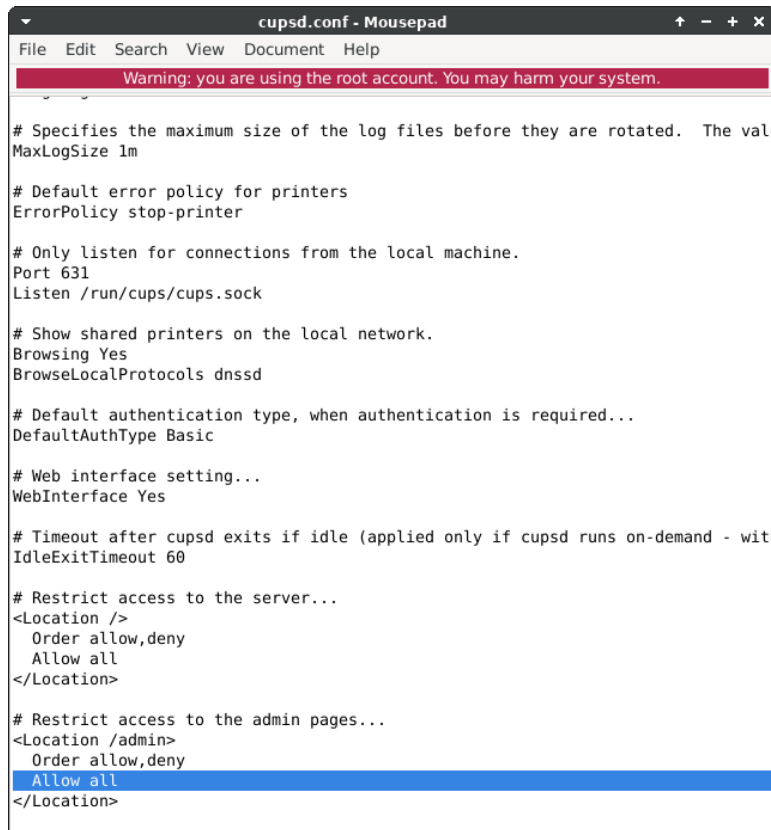
# Web interface setting...
WebInterface Yes

# Timeout after cupsd exits if idle (applied only if cupsd runs on-demand - with
IdleExitTimeout 60

# Restrict access to the server...
<Location />
  Order allow,deny
  Allow all
</Location>
```

This setting basically *tells* CUPS to just allow everyone access to the print server. Since we added the printer as root, anyone trying to modify any of the printer's settings through CUPS has to have the root password, so there's no need to alarm yourself regarding security ☺. All anyone can do is view the printer's settings through CUPS and that's basically it.

Next we have to find the `# Restrict access to the admin pages...` line and after the `Order allow,deny` line, add a new line and write `Allow all` in it, as shown in the image below.



```
cupsd.conf - Mousepad
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.

# Specifies the maximum size of the log files before they are rotated. The value
MaxLogSize 1m

# Default error policy for printers
ErrorPolicy stop-printer

# Only listen for connections from the local machine.
Port 631
Listen /run/cups/cups.sock

# Show shared printers on the local network.
Browsing Yes
BrowseLocalProtocols dnssd

# Default authentication type, when authentication is required...
DefaultAuthType Basic

# Web interface setting...
WebInterface Yes

# Timeout after cupsd exits if idle (applied only if cupsd runs on-demand - with
IdleExitTimeout 60

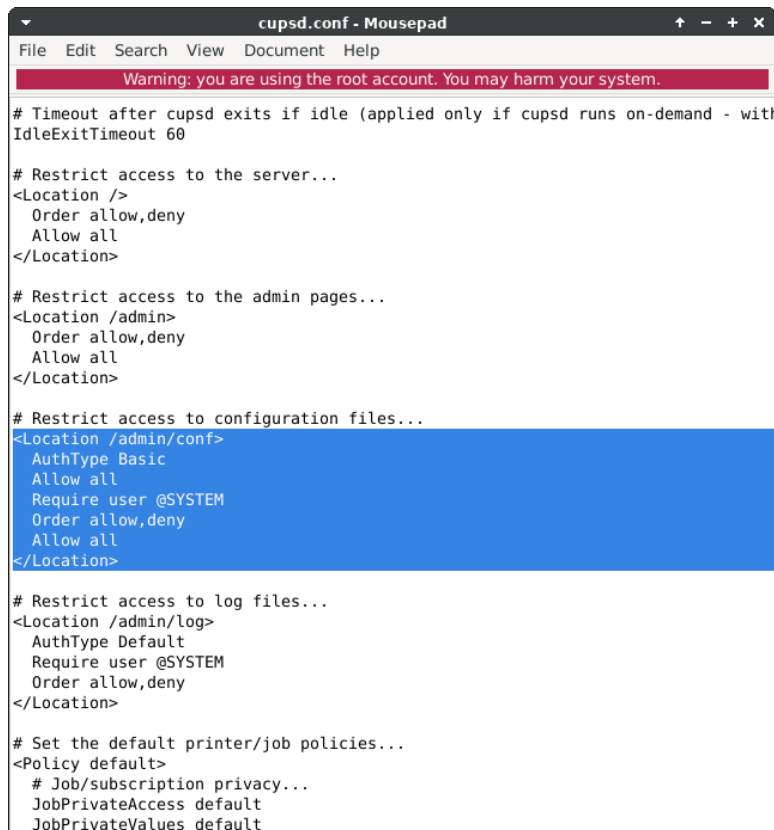
# Restrict access to the server...
<Location />
  Order allow,deny
  Allow all
</Location>

# Restrict access to the admin pages...
<Location /admin>
  Order allow,deny
  Allow all
</Location>
```

As the setting says, this basically allows everyone to access the admin pages (the web UI's admin pages). Once again, the printer is set up from the root account, so no worries regarding security, everyone can just read (view) the admin pages, nothing more ☺.

Next, find the `# Restrict access to configuration files...` line and change whatever is between `<Location /admin/conf>` and `</Location>` to the following.

```
<Location /admin/conf>
  AuthType Basic
  Allow all
  Require user @SYSTEM
  Order allow,deny
  Allow all
</Location>
```



```
File Edit Search View Document Help
Warning: you are using the root account. You may harm your system.

# Timeout after cupsd exits if idle (applied only if cupsd runs on-demand - with
IdleExitTimeout 60

# Restrict access to the server...
<Location />
  Order allow,deny
  Allow all
</Location>

# Restrict access to the admin pages...
<Location /admin>
  Order allow,deny
  Allow all
</Location>

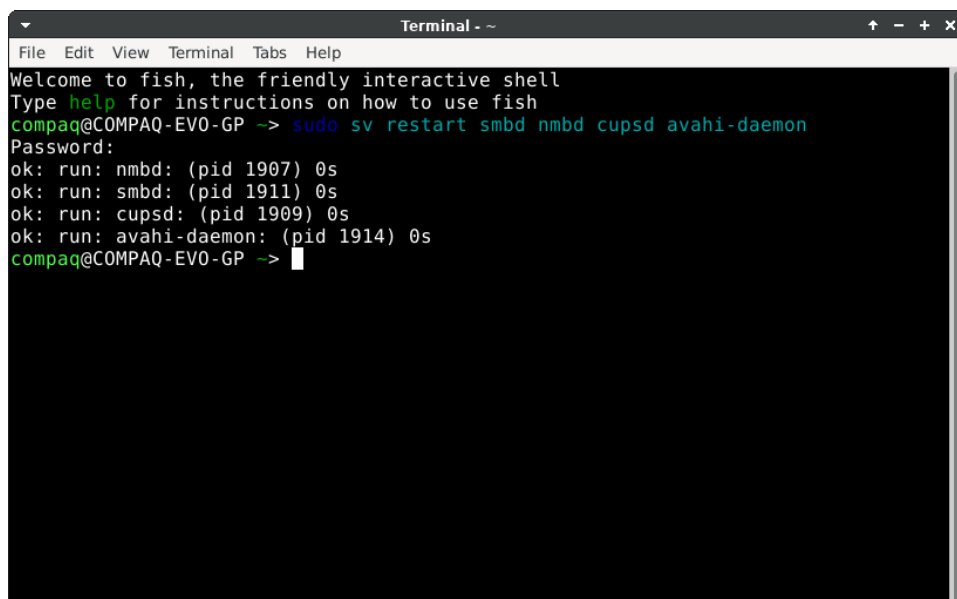
# Restrict access to configuration files...
<Location /admin/conf>
  AuthType Basic
  Allow all
  Require user @SYSTEM
  Order allow,deny
  Allow all
</Location>

# Restrict access to log files...
<Location /admin/log>
  AuthType Default
  Require user @SYSTEM
  Order allow,deny
</Location>

# Set the default printer/job policies...
<Policy default>
  # Job/subscription privacy...
  JobPrivateAccess default
  JobPrivateValues default
```

This also strips down a bit on security, but once again, nothing to worry about. Essentially, the user that would like to modify any of the printer's settings has to have the admin's password or the root password (which are the same on most popular Linux distro's, even though they can differ and you can change them after the distro's installer finishes installing the OS).

And that is basically it. All we have to do now is just restart the CUPS service (`sudo systemctl restart cupsd` on systemd based distros) and that's that 😊. Since all three services (smbd, nmbd and cupsd), as well as the avahi-daemon, are supposed to work together regarding the printing services, it's wise to restart all of them at once (one after another, the order is not really important). In my case, I'll use `sudo sv restart smbd nmbd cupsd avahi-daemon`.



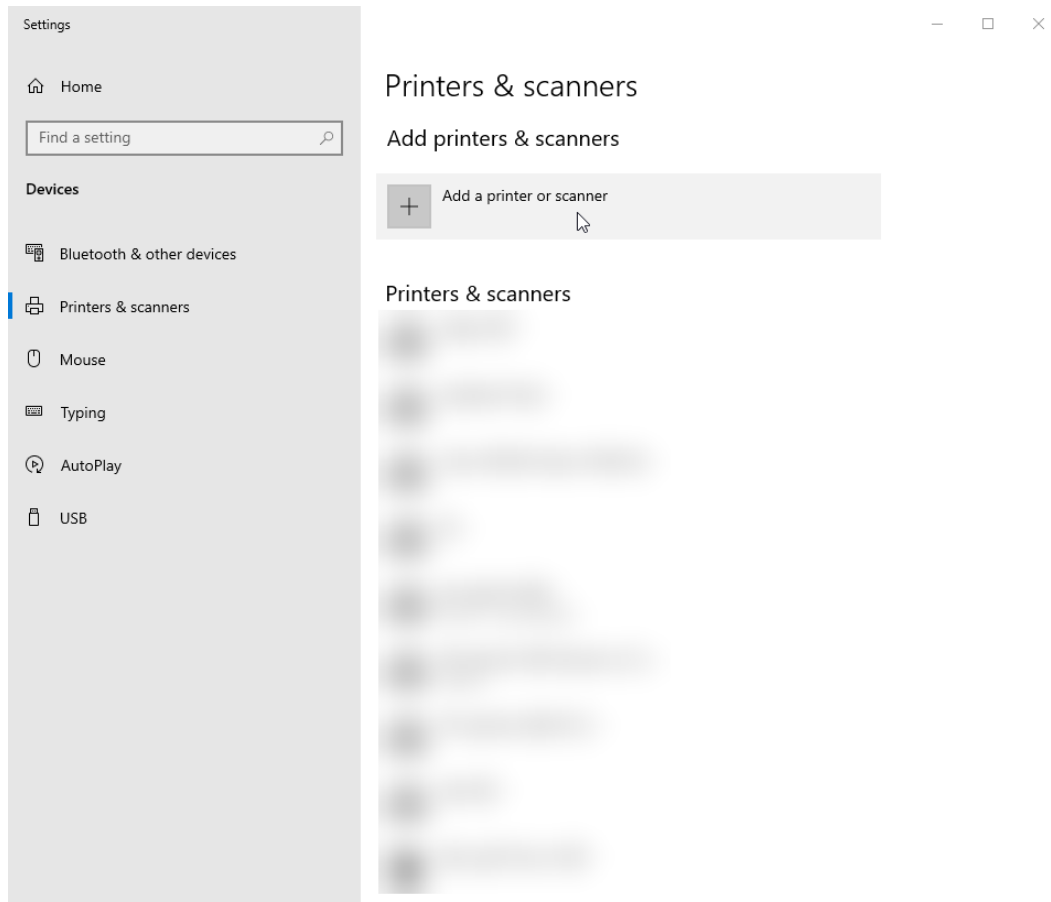
```
Terminal - ~
File Edit View Terminal Tabs Help
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
compaq@COMPAQ-EVO-GP -> sudo sv restart smbd nmbd cupsd avahi-daemon
Password:
ok: run: nmbd: (pid 1907) 0s
ok: run: smbd: (pid 1911) 0s
ok: run: cupsd: (pid 1909) 0s
ok: run: avahi-daemon: (pid 1914) 0s
compaq@COMPAQ-EVO-GP -> █
```

Once again, if you can't be bothered with commands or figuring out if your distro runs systemd or another service/init manager, just restart the print server 😊.

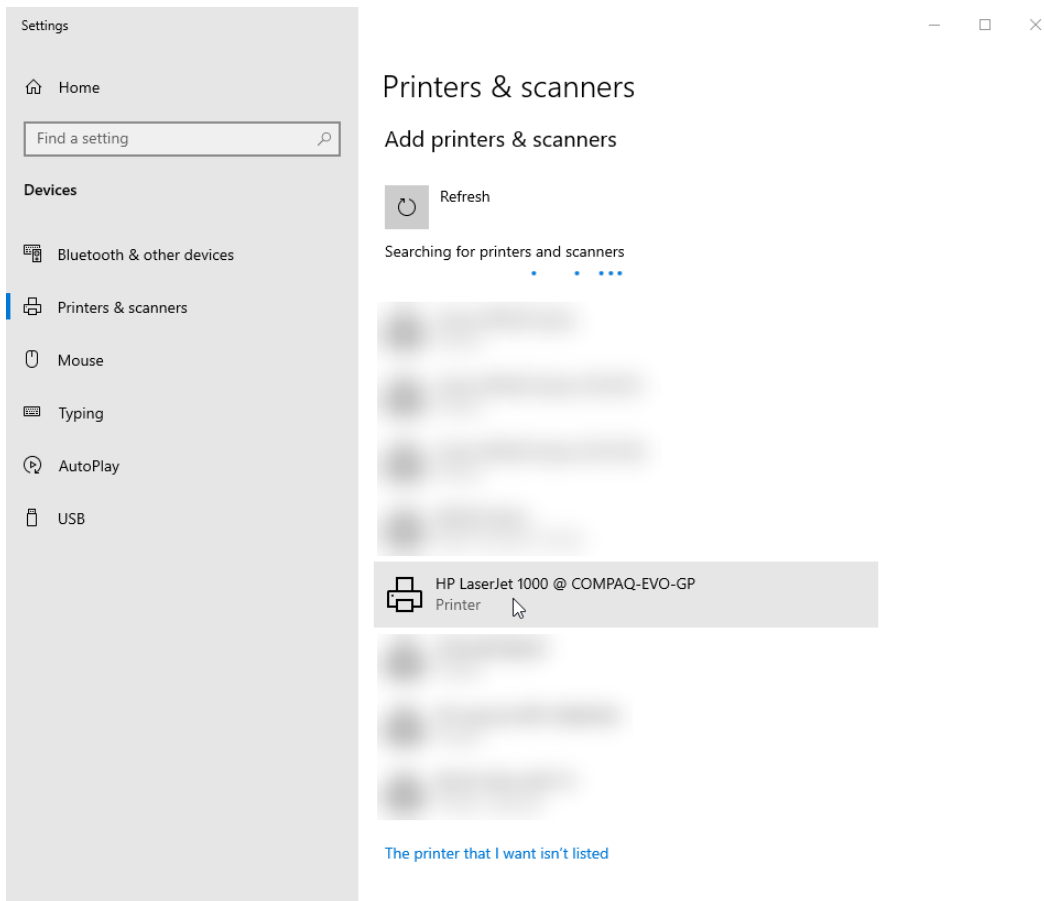
The printer is now configured and we can move on to adding it on our local network's Windows installs 😊.

❖ Adding the printer on Windows 10/11 (x86 & x64)

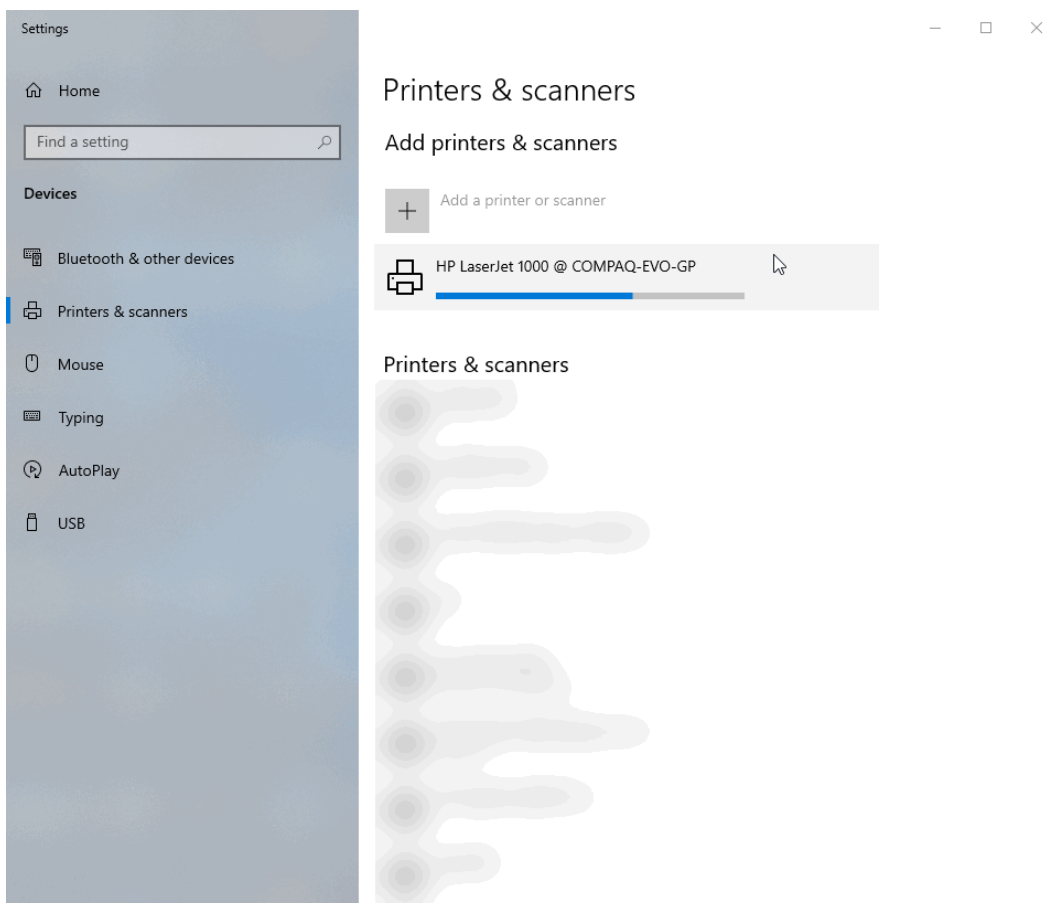
Things are super easy in Windows 10 and 11. Hit Start (the Windows logo key) and start typing "printers" and the **Printers & Scanners** section (from the new Settings section) will pop up. Click on the **Add a printer or scanner** button.

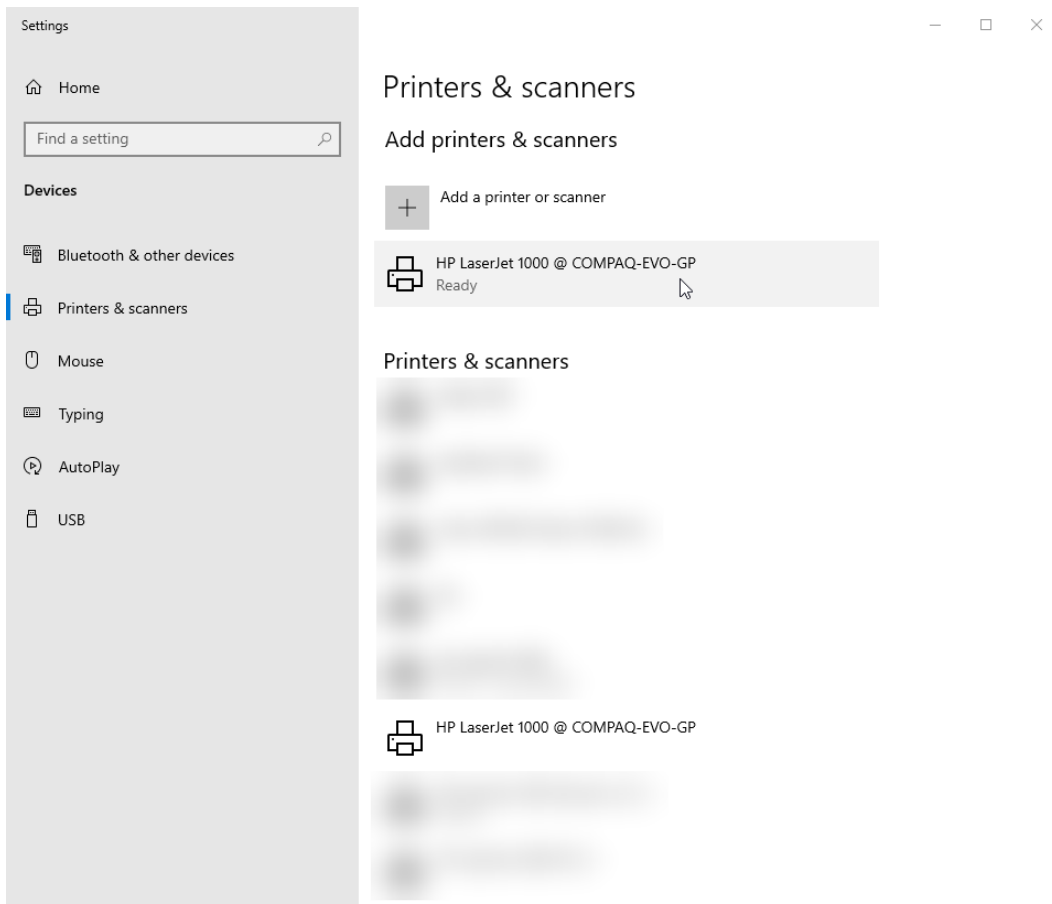


After clicking on the button, Windows will start searching the local area network for available printers that advertise themselves through the IPP or WSD (Web Services for Devices) protocols. If you've configured everything correctly, the printer should show up as **<Human Readable Printer Name> @ <COMPUTER-HOSTNAME>**.



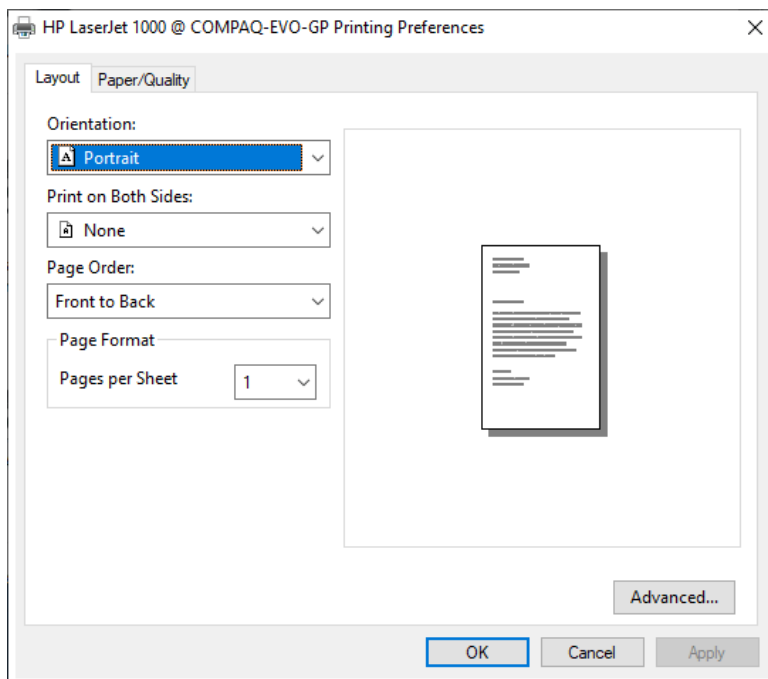
Click on the printer and then click on **Add device**, Windows should continue installing it. It will use the generic IPP Class driver, but this driver works just fine with CUPS, no problems there ☺.





And that is basically it. Try and print a test page or a document to this printer, it will print 😊.

For the x86 (32-bit) versions of Windows 10, you can also try and add this printer through Samba (manually through the network) if you've got working x86 drivers for your printer and they work in Windows 10. That is actually the preferred method, since the method described above uses a generic IPP driver from Microsoft and it does work, but can't print in color (or so I've read, I don't own any color printers) and the settings of the driver are somewhat limited. Go to **Control Panel → Devices and Printers** and look for the printer added through the **Printers & Scanners** Settings option (the *new* Control Panel in Windows 8/8.1 and above, mostly usable and working in Windows 10 and above). Right click on the printer → **Printer properties**, now click on **Preferences**.



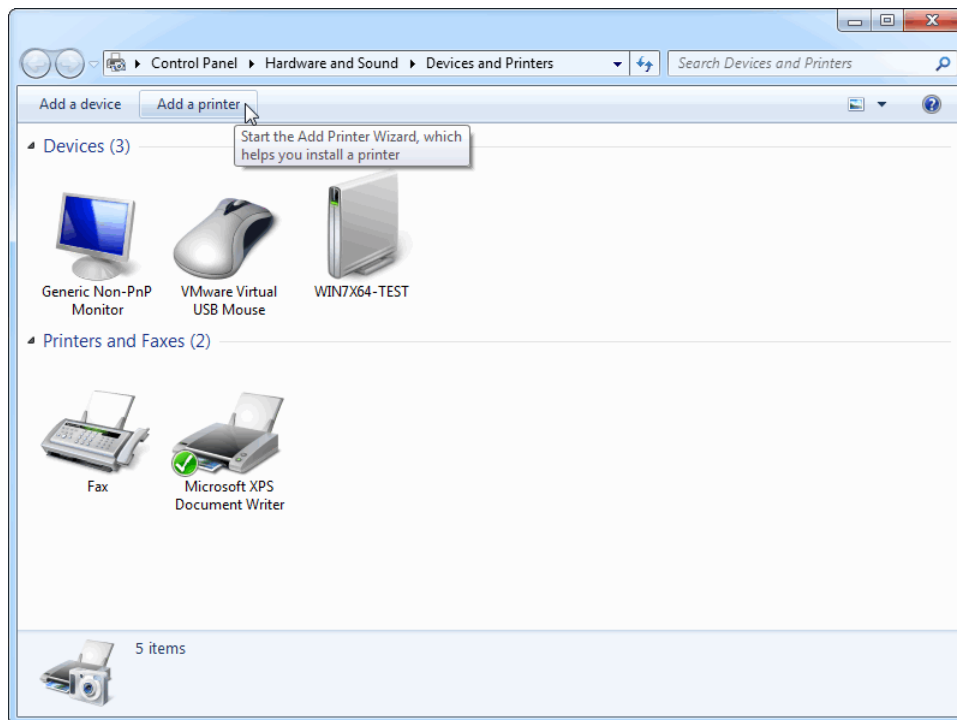
As you can see, the driver doesn't offer a lot of options. Even the **Advanced** button doesn't offer much, except choosing paper size and scaling. If your printer has no other options except these (lower end laser models) and can't print in color, using this driver is just fine. But if you'd like to print in color and the Windows install is 32-bit, it's preferable that you install the printer through the network (SMB) and load the 32-bit drivers for the printer.

❖ Adding the printer on Windows Vista/7 (x86 & x64)

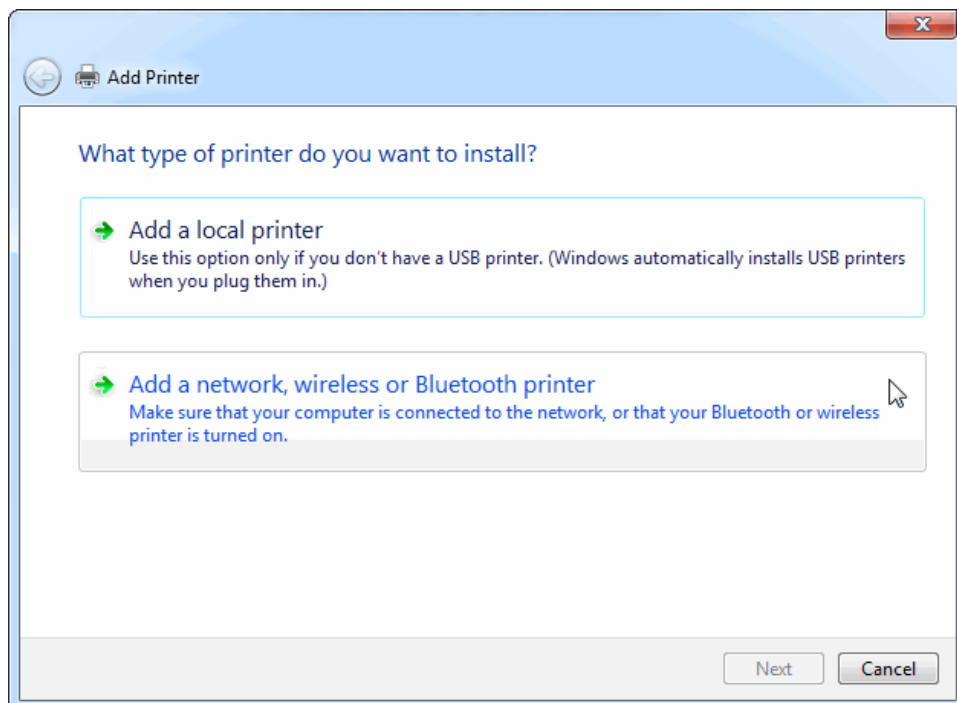
Things get more and more complicated as we move down in Windows versions, LOL ☺. But, the good news is that in Windows 7 and lower, we'll be using a PS (PostScript) driver which can print in color.

In this case, we'll use the PS *language* (standard) that CUPS understands. CUPS by default is equipped with IPP/HTTP (and the secure versions as well, IPPS/HTTPS) as a data transfer protocol and PS (PostScript) as a *language* that is used to *talk* to the printer. There are plugins that expand the capabilities of CUPS (such as the PCL5 and PCL6 plugins), but since we (I) have no idea what kind of printer we'll be using and what kind of protocol (language) it uses for printing, it's best to stick to the open source and standardized protocols and for which there are generic (or compatible) drivers in most Windows versions.

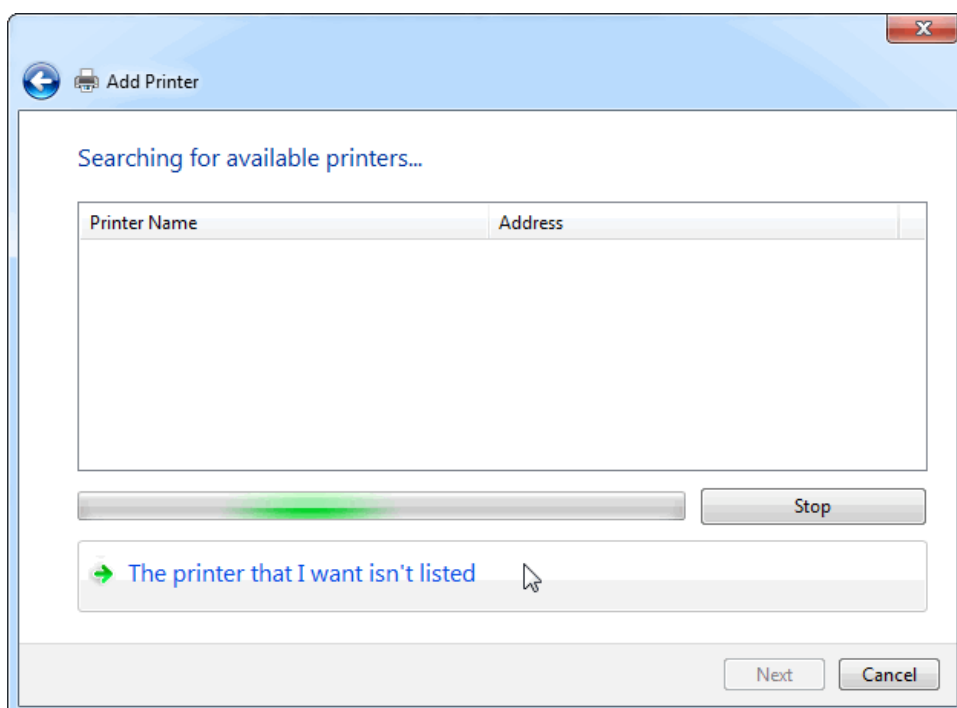
Let's start. Go to **Control Panel → Devices and Printers** and click on the **Add printer** button.



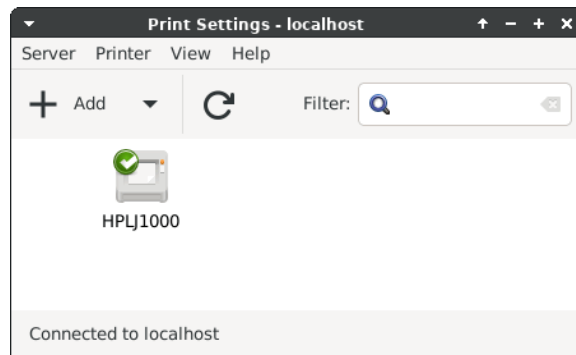
Afterwards, select **Add a network, wireless or Bluetooth printer**.



Windows will try to search for printers that have *advertised* themselves as network printers through mDNS/ZeroConf. We don't actually want that, it'll connect through Samba to the printer. Even if our printer is listed on the list, we want to print through HTTP (IPP), so just hit on **The printer that I want isn't listed** button.



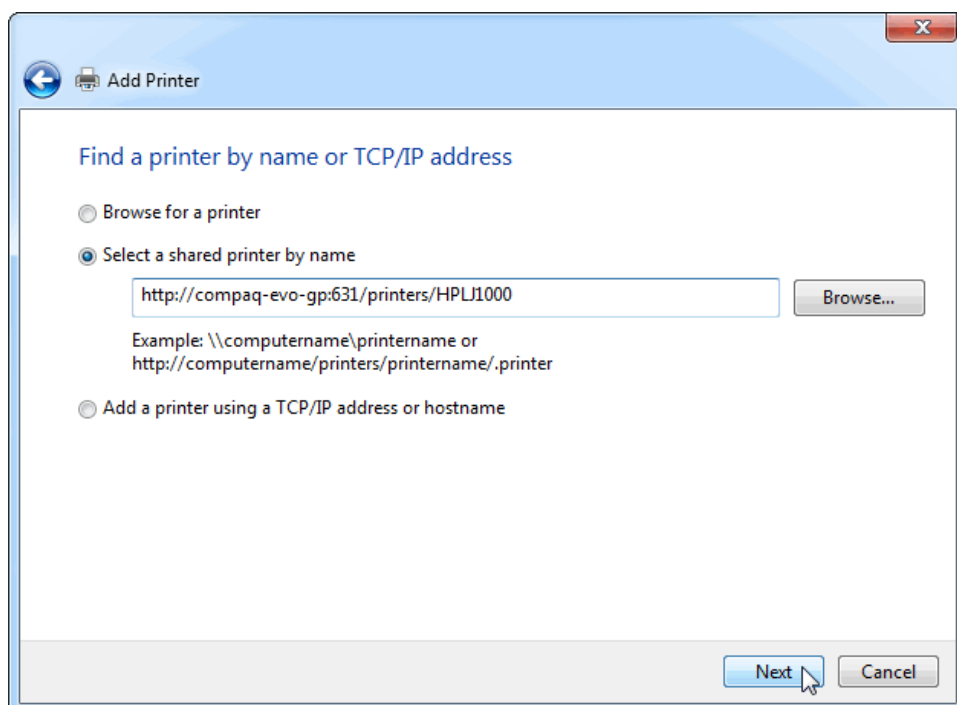
Next, click on **Select a shared printer by name**. Mark down the name of the printer as listed in the Print Settings in Linux.



In this case, the printer's name is **HPLJ1000**, which is also the share name of the printer. Also, keep note of the caption of the letters (all capitals). Caption doesn't matter in Samba shares, since Windows doesn't distinguish between capital and small letters (in Windows, the directories **Share** and **share** have the same name, which is not true for the rest of the operating systems out there, as well as all POSIX compatible Oses, like Linux), but it does when HTTP or IPP is used as a data transfer protocol. By default, HTTP distinguishes between capital and small letters in an URL (this can be mitigated, but by default, it does).

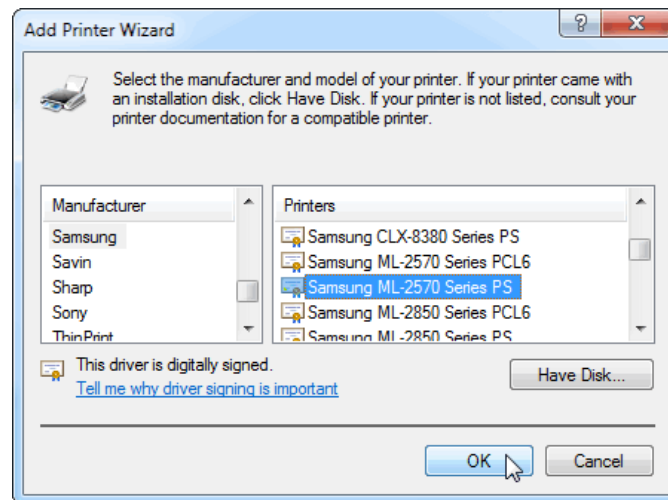
Moving on. As noted above, select the radio button **Select a shared printer by name** and enter the HTTP address of the printer. The HTTP address should have the following syntax: `http://<print_server_hostname>:631/printers/<printer_share_name>`. If you don't like using hostnames, use `http://<print_server_ip>:631/printers/<printer_share_name>`. Just remember that in this case you have to set up the print server with a static IP address. Otherwise, the next time the DHCP server assigns an address to the print server, the address might not be the same as the one you entered in the address field (the current IP address of the print server).

So, in this case, we'll be entering `http://compaq-evo-gp:631/printers/HPLJ1000`. The `631` part is the port on which the print server is listening. This can be changed, but the default one is just fine.



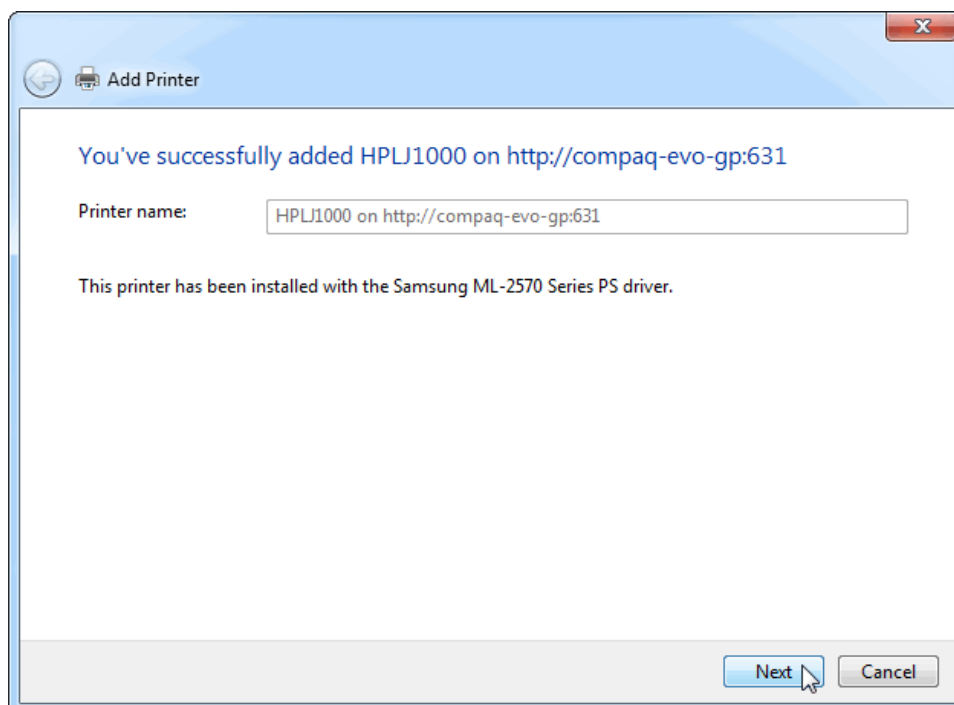
Don't pay attention to what the **Add Printer** wizard says about the print server's printer share address (`http://computername/printers/printername/.printer`), the `/.printer` part at the end of the address doesn't work (or, at least, has never worked for me, LOL ☺). Maybe it works on printers shared on Windows servers, who knows... I wouldn't know since I've never configured a Windows based print server, LOL ☺.

After we click **Next**, Windows will prompt us for a driver. What we're looking for is a generic PS (PostScript) driver. The **Microsoft PS Class Driver** should do the job fine, but in some cases, I've had problems with it (like printing zoomed out, extra space at the edges of the page). Through trial and error, I found out that the Samsung ML-XXXX series PS drivers work best with most printers in this type of setup. So, we'll choose a Samsung PS driver. In this case, It'll be the **Samsung ML-2570 Series PS** driver.

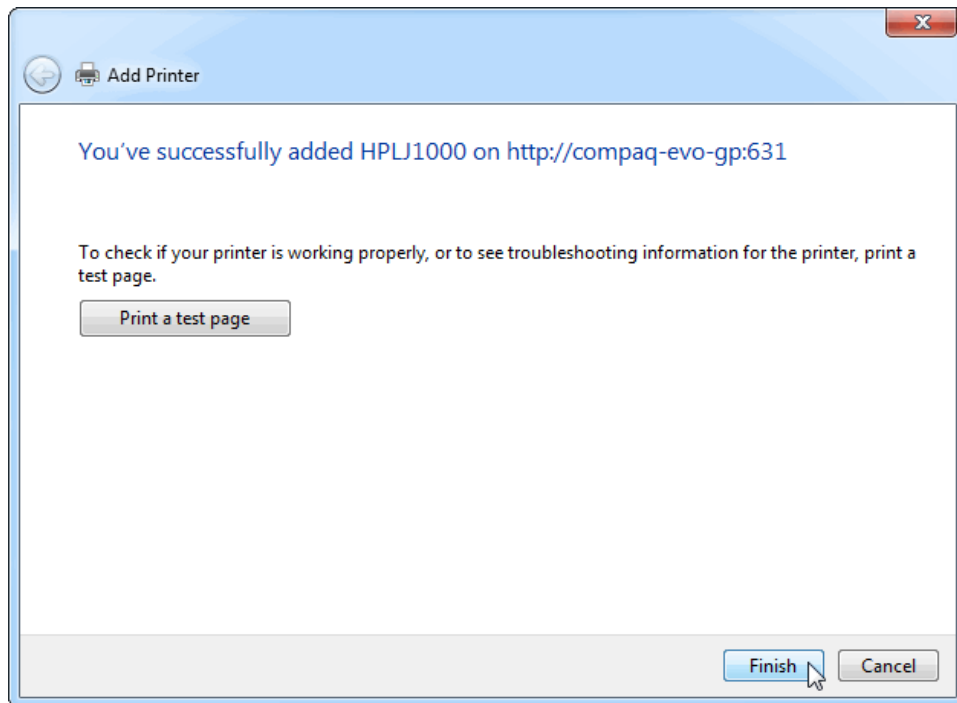


If this driver doesn't work or doesn't work properly with your printer, choose another PS driver from the list. Trial and error is part of this *hacky* setup ☺.

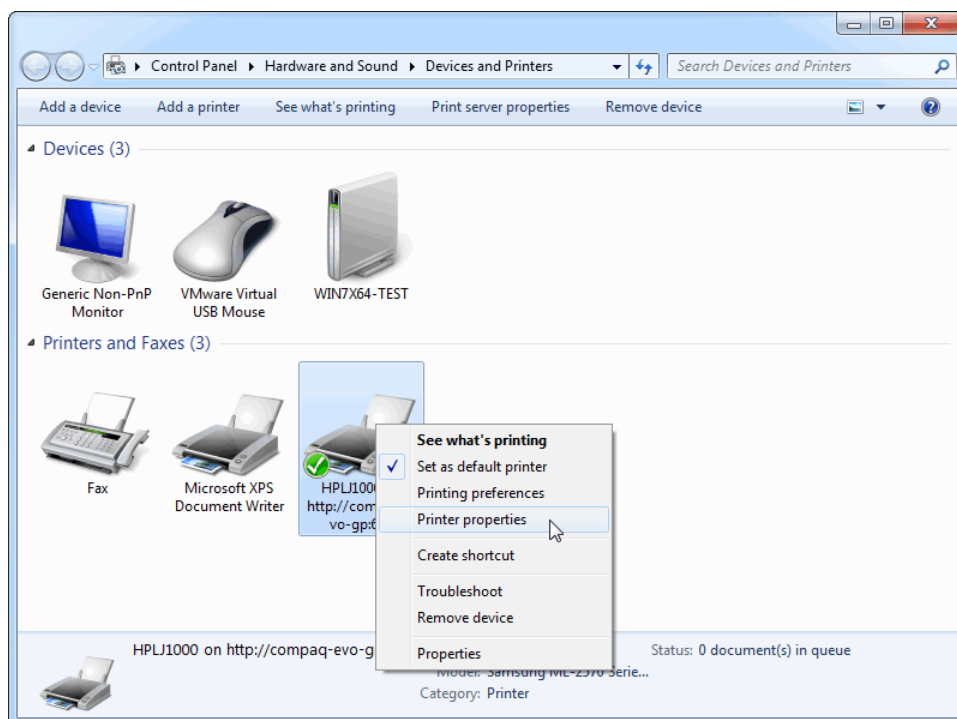
After we click on OK, Windows will install the printer driver and we can click the **Next** button.



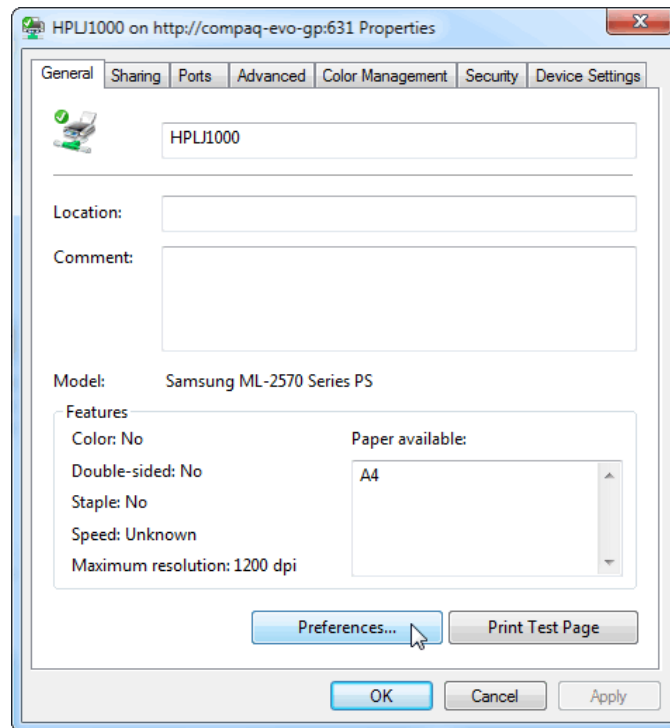
Afterwards, click on **Print a test page**. If a test page comes out of the printer, that means that the driver is working. We can tinker with the settings later on, but since the driver is working, we can click the **Finish** button.



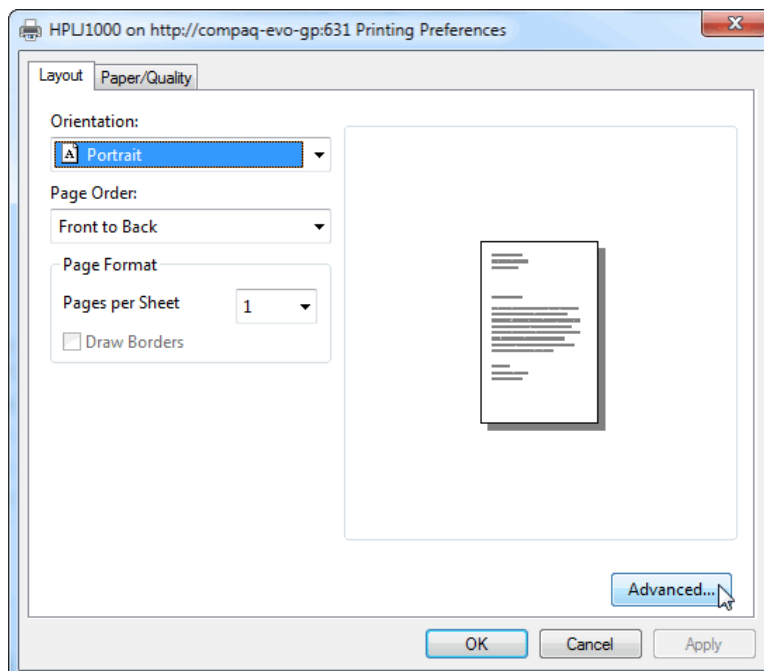
Now, since we've (most probably, LOL 😊) added the printer, let's look at some of the options this printer driver has. Go to **Control Panel** → **Devices and Printers**, right click on the printer and choose **Printer Properties**.



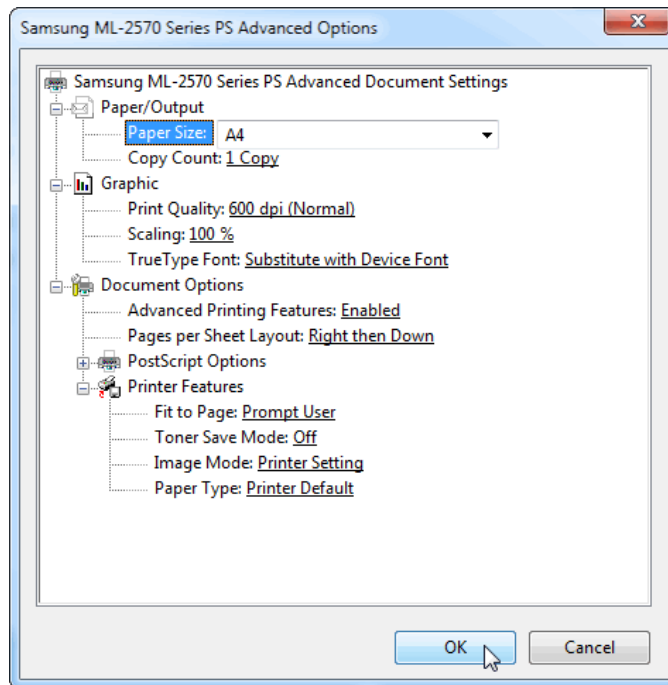
In the Printer Properties tabbed window, select the **General** tab and then click on **Preferences**. The driver's settings are *hidden* behind this button.



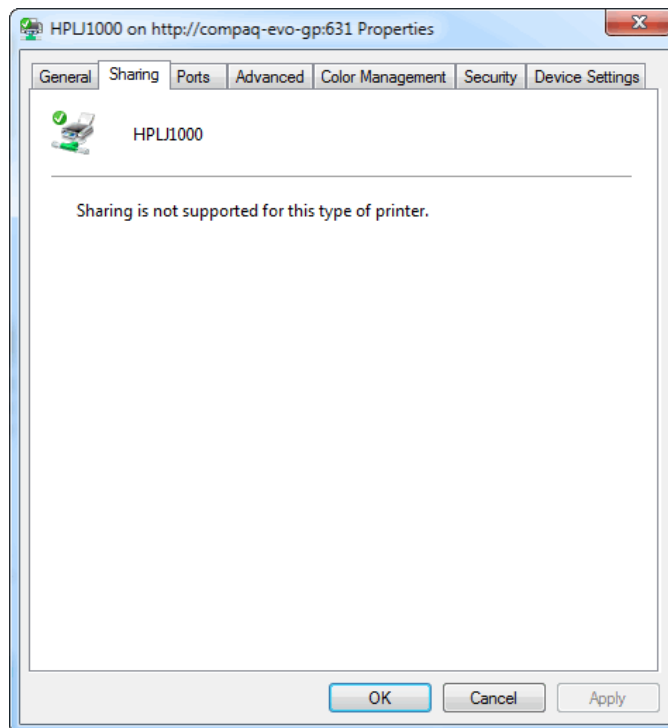
As we can see from the settings window, there aren't a lot of options available. This is a generic driver from the driver store that Windows will use if no other driver is supplied, but it should suffice for most people's printing needs. In the **Layout** tab, click on the **Advanced** button.



Here, we can set up some options that most low end (mid/low budget) USB/LPT/Ethernet printers have. Don't expect every option to work, since we're using a driver that's not even intended to be used with the printer you've shared on the print server. From what I've gathered, the paper size option works 100% of the time, and that's about it ☺. Every other option... depends, but most of them don't work. The scaling and toner save options might work, if you choose a driver from the same manufacturer, but a different model. So, I'll just set up the paper size option here and be done with the advanced settings.

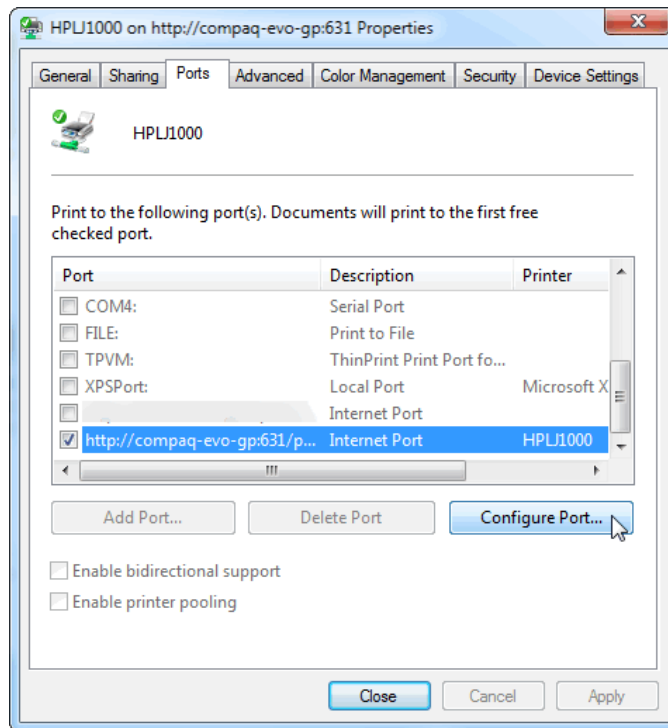


Afterwards, just click the **OK** button and the **OK** button on the preferences window to get back to the properties window. After you get back to the properties window, click on the **Sharing** tab.

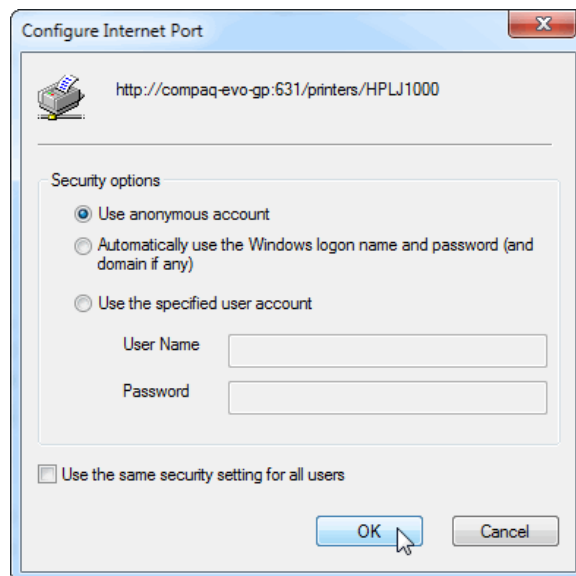


As expected, no sharing options at all. If we used SMB (Samba) or IPP to connect to the printer, we would've had sharing options, but, since we're transferring data to CUPS via HTTP, we don't get any sharing options. This isn't such a big deal, the print server will be online 24/7 and we can add the printer to another machine through the same HTTP address we used to add the printer earlier.

Let's go over to the **Ports** tab.

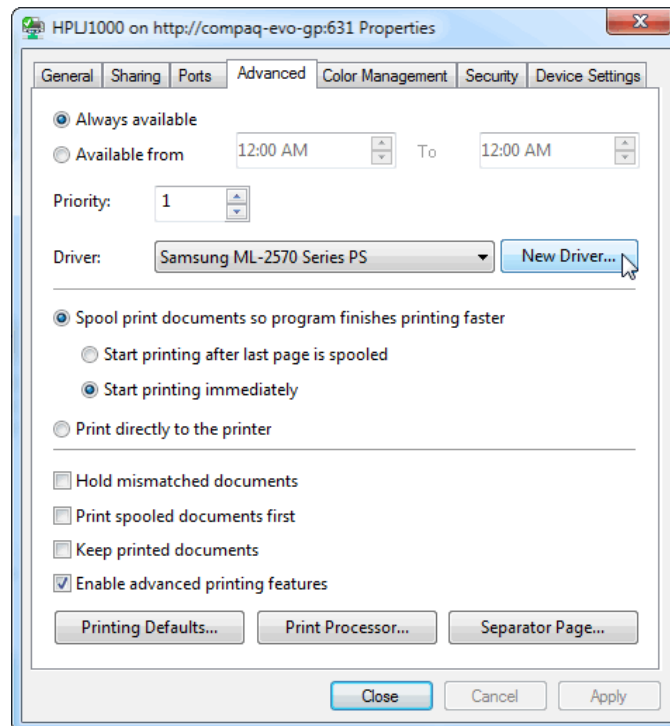


Note the **Description** field, it reads **Internet Port**. As expected, Windows uses HTTP to communicate with the printer. Now, click on the **Configure Port** button.



Note that we're not actually authenticating to the print server in any way to print. If we didn't do the `Allow all` additions to the CUPS configuration file, we most probably would have to authenticate ourselves in order to print (most probably with the local user account on the print server or any other user account that has print privileges on the print server).

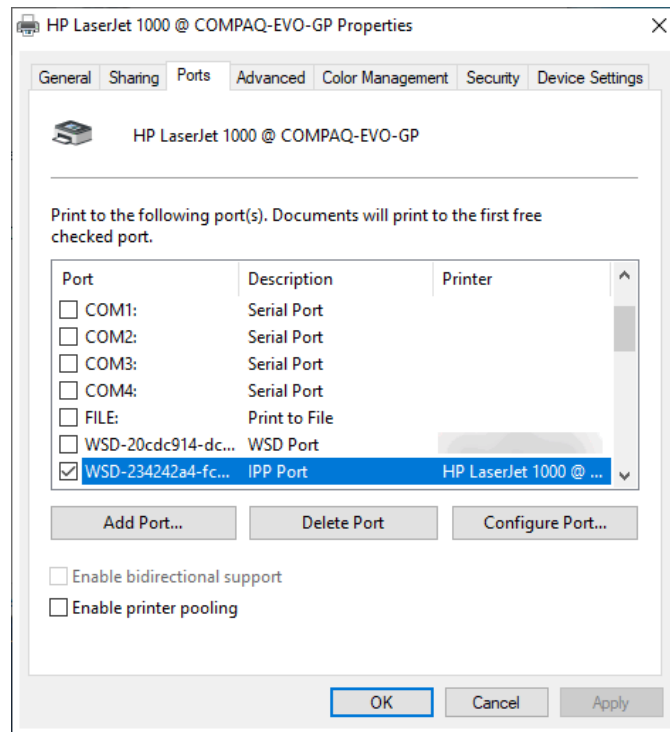
Click the **OK** button to go back to the Properties tabbed window and go to the **Advanced** tab.



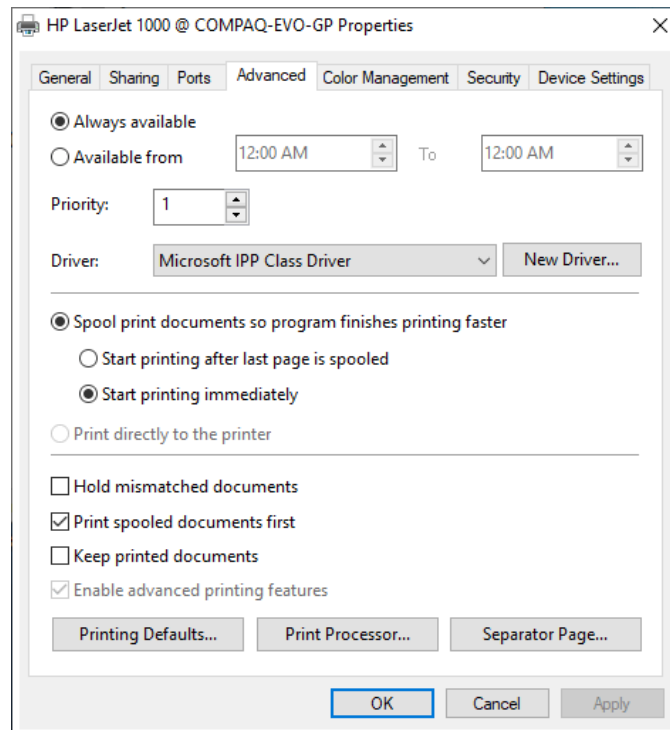
This tab holds real value regarding testing drivers and how/if they work. Note the **New Driver...** button and the **Driver:** drop down menu. See, we don't have to actually uninstall the printer in order to use a different driver for the printer. We just have to choose one from the drop down list (the already installed printer drivers in Windows) or choose one manually from the store or a driver .inf file (if we want to manually install a new driver, a driver that's not included in the Windows driver store) through the **New Driver...** button. Don't mess with the settings in the **Printing Defaults...**, **Print Processor...** and **Separator Page...** buttons, the defaults are fine, you might mess something up with the configuration. If you need to change the driver, just use the **Driver:** drop down menu and/or **New Driver...** buttons.

We can click the **Close** button now.

What I haven't mentioned is that we can use the same method as presented in Windows Vista/7, to connect to the printer in Windows 10 or above. We don't have to use IPP in Windows 10/11 to print to this printer. But, we can't use the IPP class driver from Microsoft if we choose to add the printer as a HTTP shared printer. The IPP class driver works only through an IPP port. If we go to **Control Panel → Devices and Printers** in Windows 10, right click on our printer (in my case **HP LaserJet 1000 @ COMPAQ-EVO-GP**), go to **Properties** and then choose the **Ports** tab, here's what we'll find.



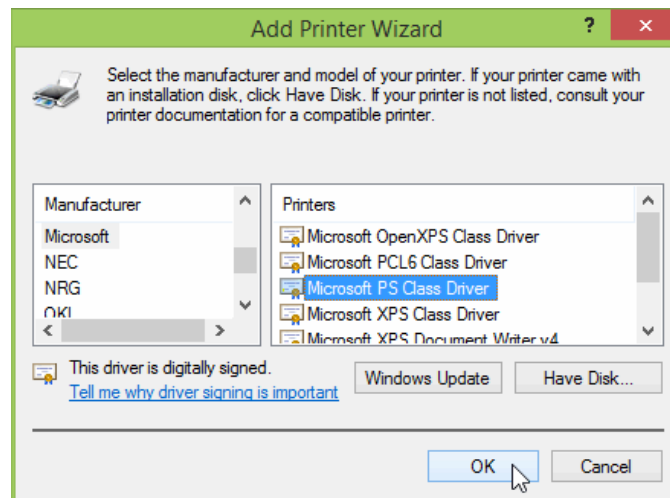
Note the **Description** field, it reads **IPP Port**. Now, let's shift to the **Advanced** tab and see which driver the printer is using.



As expected, it's using the **Microsoft IPP Class Driver**. This driver works only with the IPP protocol, so you can't use it if you're using HTTP as a communication protocol (between the print server and the client). So, if you change the driver here with a driver that's not an IPP class driver, the printer won't work or it will print raw data, since CUPS expects IPP data to be transferred through the IPP protocol, not PS or PCL data. The same thing will happen if we use the IPP driver, but add the printer as a HTTP printer. You can't mix and match these two types of communication protocols and the drivers they use (well, at least not in Windows ☺). If you choose to use the HTTP protocol, you have to use a PS driver (CUPS understands PS through HTTP), if you use the IPP protocol, you have to use the Microsoft IPP Class Driver or another IPP driver, preferably supplied by the manufacturer (since that driver will most likely have more options than the generic Microsoft IPP driver).

❖ Adding the printer on Windows 8/8.1 (x86 & x64)

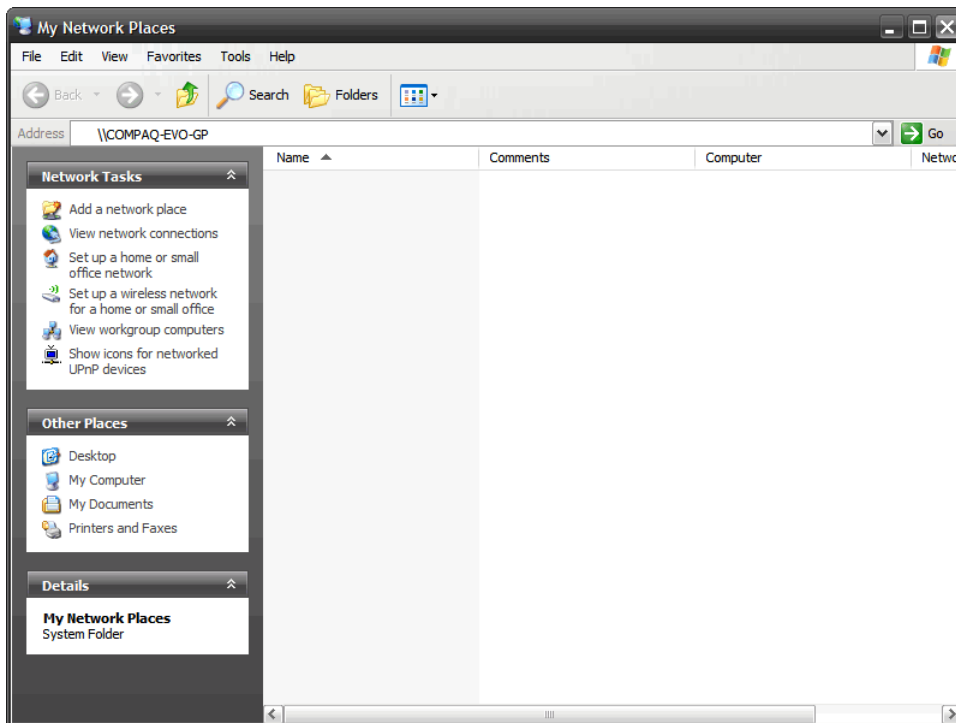
Adding the printer in Windows 8/8.1 is more or less the same as in Windows Vista/7, we use the HTTP protocol to add the printer. What may differ is the number of PS drivers available in Windows 8/8.1, for example the **Microsoft PS Class Driver** (which might not be available on Windows 7, depending on how many drivers were added or removed from the installation disc/USB drive). You can try and use this driver, it should work just fine ☺. You can also try the **Windows Update** button, that might add new printer drivers in the selection (though, to be honest, it has never added any new drivers for me, LOL ☺).



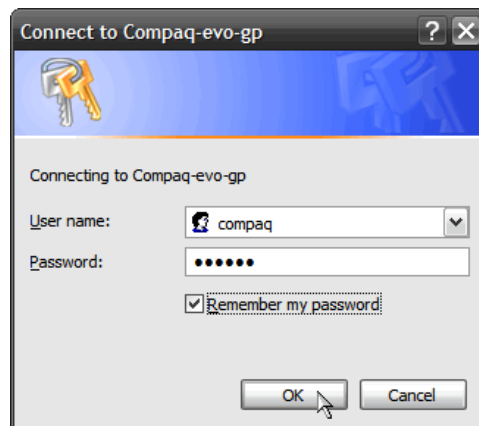
❖ Adding the printer on Windows 2000/XP (x86 & x64)

Once again, we can choose between two methods. Since Windows 2000 is x86 only, we can use the x86 drivers available for our printer. If the printer is so old, the last drivers available from the manufacturer are for Windows 95/98, you can use the HTTP data transfer protocol in both Windows XP (don't know if HTTP for printers is supported on Windows 2000). If you're using Windows XP x64, you can only use the HTTP transfer method (since, most probably, we don't have x64 at all for our printer). In any case, there is at least one method available to connect these old OSes to the print server ☺.

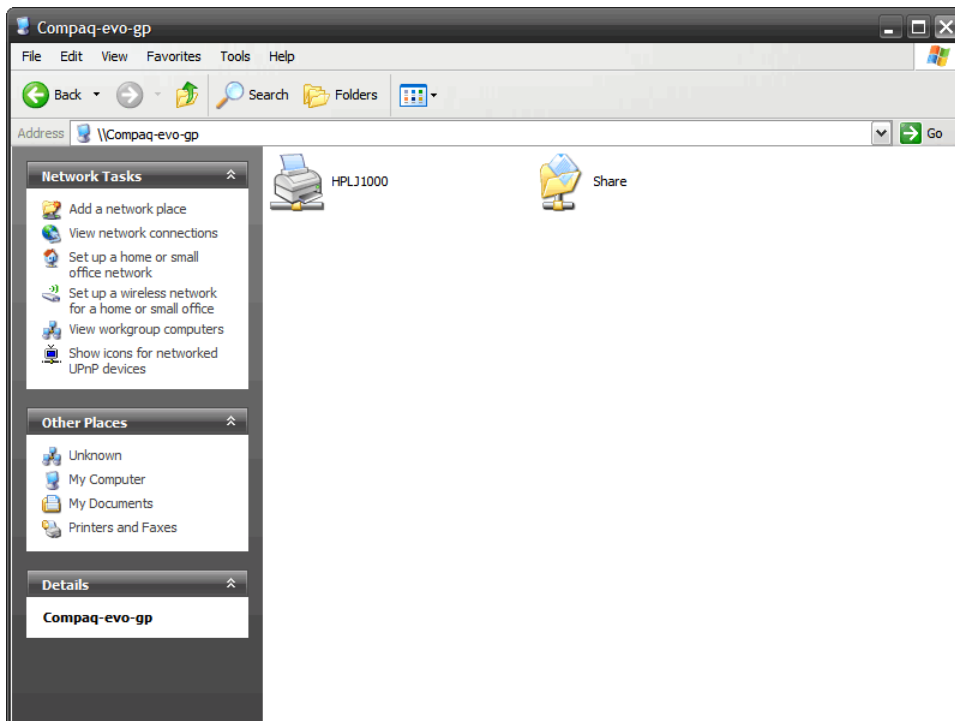
Let's get started. First, we'll use the *classic* method, where we manually install the appropriate drivers for the printer (preferably, downloaded from the manufacturer's site). On the desktop, double click on **My Network Places**. If this icon doesn't exist on your desktop, that's fine, just double click on **My Computer** or **My Documents**. If none of these icons exist on your desktop, hit **Win + E**, which will open Windows Explorer. In the address bar field, type in `\\<print_server_hostname>` and hit **Enter**. You can also use your print server's IP address instead of the hostname.



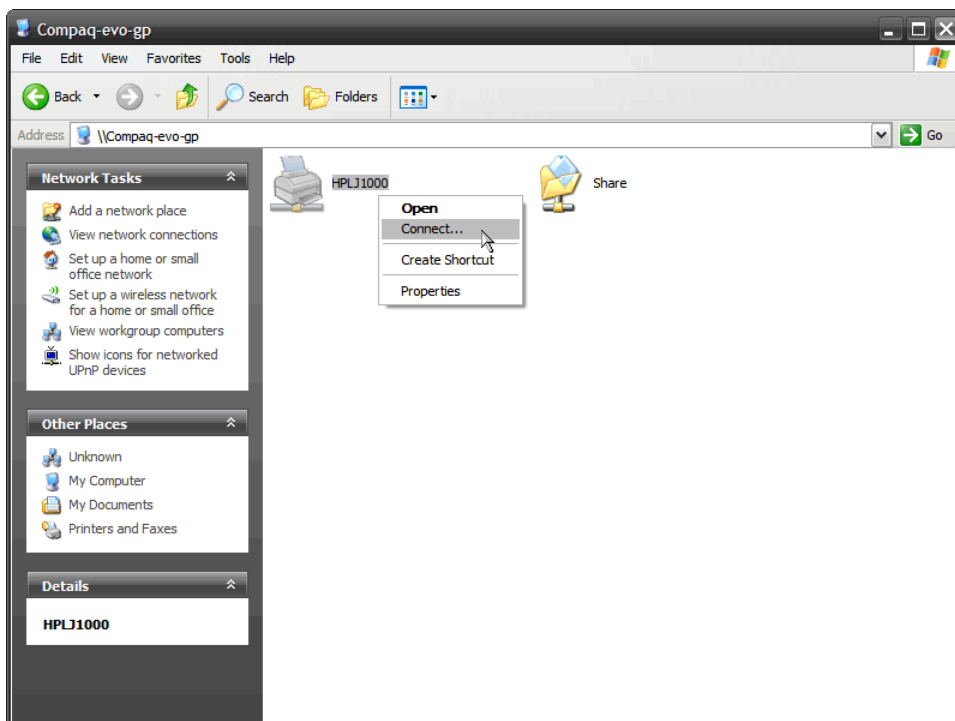
Next, Windows will most probably ask you for credentials. Remember the credentials we set in Samba? Well, now we have to use those. Enter the username and password you set in Samba to access the print server's shares through the network. In this example, I'll just enter the credentials I set up in Samba on my print server. Don't forget to tick the **Remember my password** tick box.



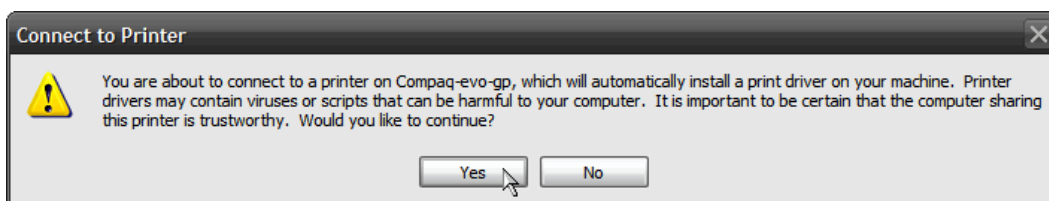
If the credentials are correct, Windows Explorer should show us the shared resources on the print server.



And, as we can see, there's the shared printer on our print server ☺. Right click on the printer and select **Connect....**



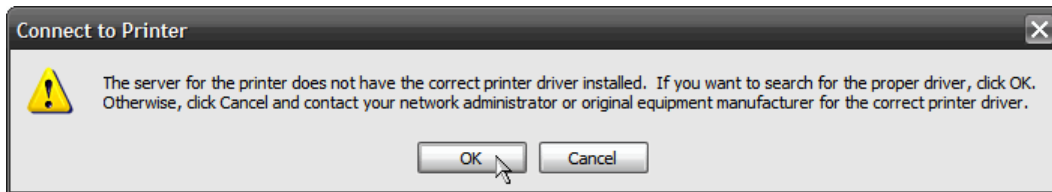
You might get this warning after you click on **Connect....**



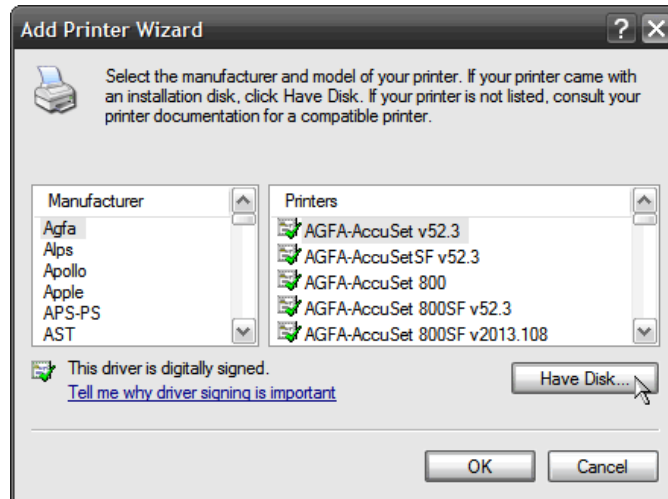
Just click on the **Yes** button. It doesn't matter anyway since we didn't configure our print server to actually share the printer drivers with the clients connected to it. Of course, this can be done, but since most operating systems nowadays are 64-bit (even smart device OSes), there's not

much point sharing the x86 (32-bit) drivers through SMB. We can manually install the drivers for the few (if any) x86 Windows OSes in our local network ☺.

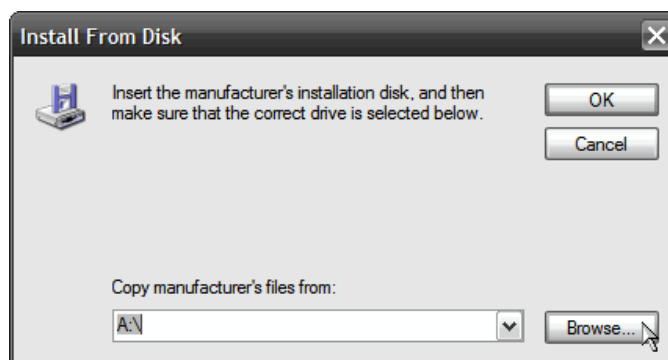
After we hit the **Yes** button, this window will appear. Yes, we do want to search for the appropriate drivers, so we click on the **OK** button.



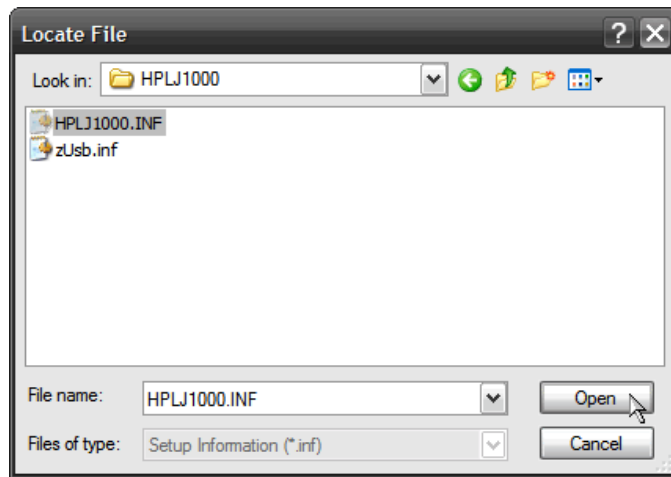
Next, the driver selection window will pop up. Click on the **Have Disk...** button.



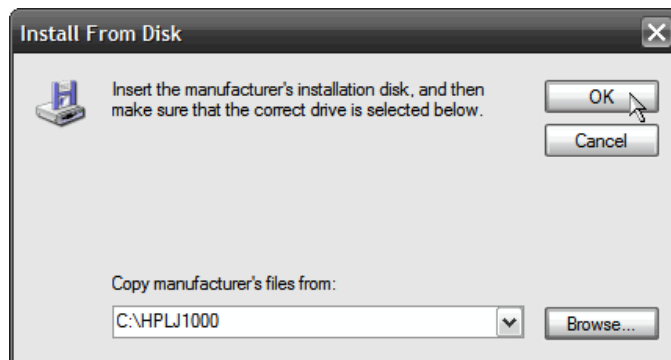
Since we're connecting to the printer through SMB (Samba) and not using CPUS as an intermediate layer (we're not using HTTP or IPP, we're using SMB as the communication protocol between the print server and the client), we need to load the appropriate drivers for the printer. In this case, HP LaserJet 1000 does have x86 drivers available for Windows 2000 and XP. I've already downloaded them, extracted the content from the .exe archive (you can do this using WinRAR or 7-Zip, just right click on the .exe file and select **Extract to <name_of_archive>** or **7-Zip → Extract to "<name_of_archive>"**) and put them in **C:\HPLJ1000**. So, after we hit the **Have Disk...** button, we need to point Windows to the .inf file (the driver installation file that holds all relevant data regarding the printer, as well as info on where to copy the driver files, etc.). When the **Install From Disk** window appears, hit the **Browse** button.



Next, we point the **Locate File** window to where our drivers are located, i.e. in this case, in **C:\HPLJ1000**. Then, we hit the **Open** button.



Once again, hit the **OK** button.



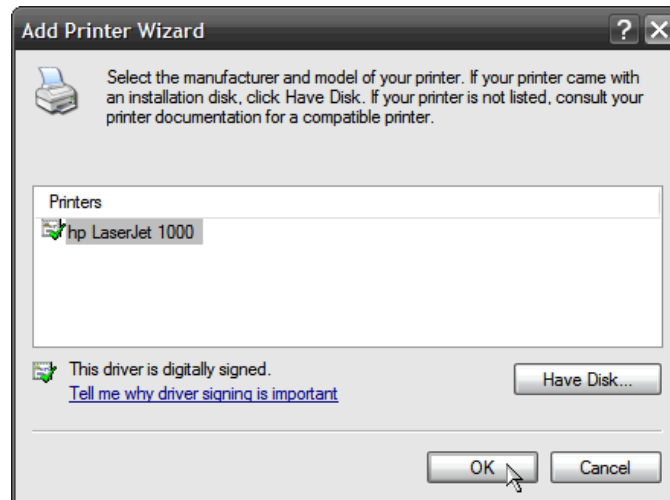
The **Add Printer Wizard** will then ask us to choose from a list of drivers available in the .inf file. In most cases, manufacturers make drivers for a series of models, not just one model and they bundle them all up in one package. This is mostly due to the fact that most of these printers differ in minor things, like having or not having Wi-Fi or maybe the mechanical parts, size/shape of the printer design, etc. These minor differences are the reason why the drivers for a whole series are bundled. It makes no sense to make different drivers for the **XXY** model, when the **XXX** model is basically the same, the only difference is, let's say, connector placement. This doesn't just go for printer drivers, this is more or less the same for any driver out there: graphics cards, USB hubs, cameras, Wi-Fi cards, etc.

In this particular case, the drivers for the HP LaserJet 1000 are one of a kind and this printer is one of a kind. Exactly one of the reasons why there are no x64 drivers available. See, back in the day, HP did a minor *boo boo* by subcontracting Zenographics to use their proprietary ZjStream host based data transfer protocol. Basically, what this means is that HP doesn't have access to the design plans for their proprietary ZJS decoder/controller chips, or the source code for the drivers. Not only that, but HP were told that they couldn't sell Zenographic's proprietary decoder chip with the ZJS protocol in their printer, but that they had to make it available separately as an add-in module, LOL ☺. So, HP had to sell the printer, but not with a native USB port on the back of the printer, oh no, a separate *connector* was sold in bundle with the printer, that actually had the decoder chip in it. The connector on the printer side looks like a regular printer port (LPT) connector and you'd be thinking "so what if I lose this USB to LPT cable, no biggie, I can always use a regular printer port cable ☺". Well, that's not the case, since this chip not only decodes the ZJS data, but also controls all of the mechanical parts in the printer, so basically, without that *connector* (*converter cable* as referred to by HP), this printer doesn't have a *brain*, so even if you connect a regular printer cable to it, it won't work. You have to use this proprietary *converter cable* which also holds the *brain* of the printer. So, long story short, HP doesn't have access to the printer driver's source code or the design plans for the ZJS decoder/controller chip and they're at the mercy of Zenographics whether they'd like to release x64 (64-bit) drivers for this particular printer or not. Of course, in pursuit for profit, Zenographics never release an x64 driver, thinking that the lack of drivers for x64 OSes will drive (force) HP or other major printer manufacturer to subcontract them for the electronics and

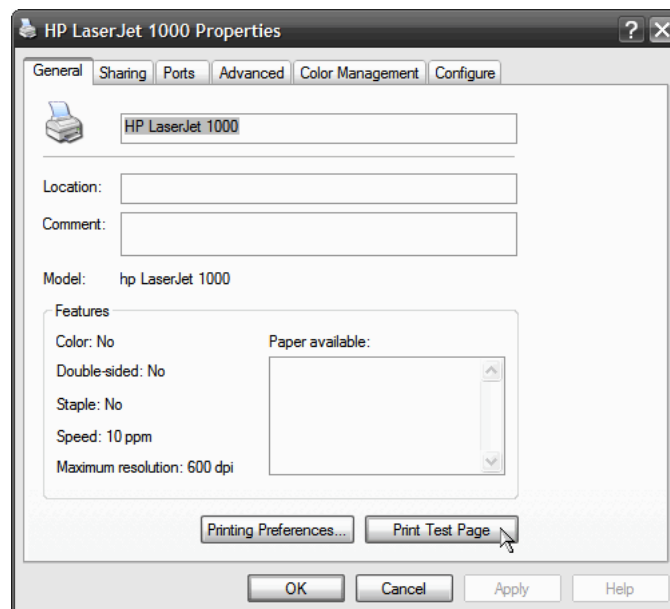
drivers part of future printer models, for which they will release x64 Windows drivers. This, of course, never happens and Zenographics ceases to exist in the mid 2000's.

In any case, this is why x64 (64-bit) drivers are available for similar looking models, like the HP LaserJet 1100, 1200 or 1300, but not for the LaserJet 1000. The other models use PCL as the communication *language* with the printer.

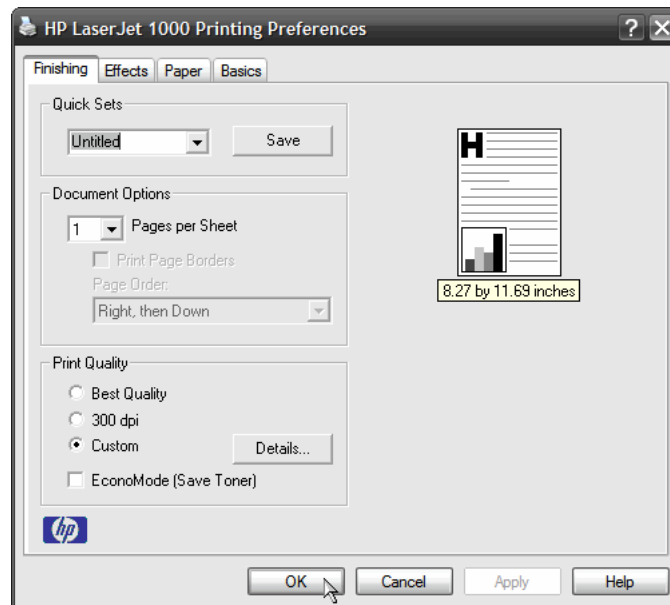
Getting back on track, that is why in this particular case we have only one driver to choose from the list, the HP LaserJet 1000. We select it and click on the **OK** button.



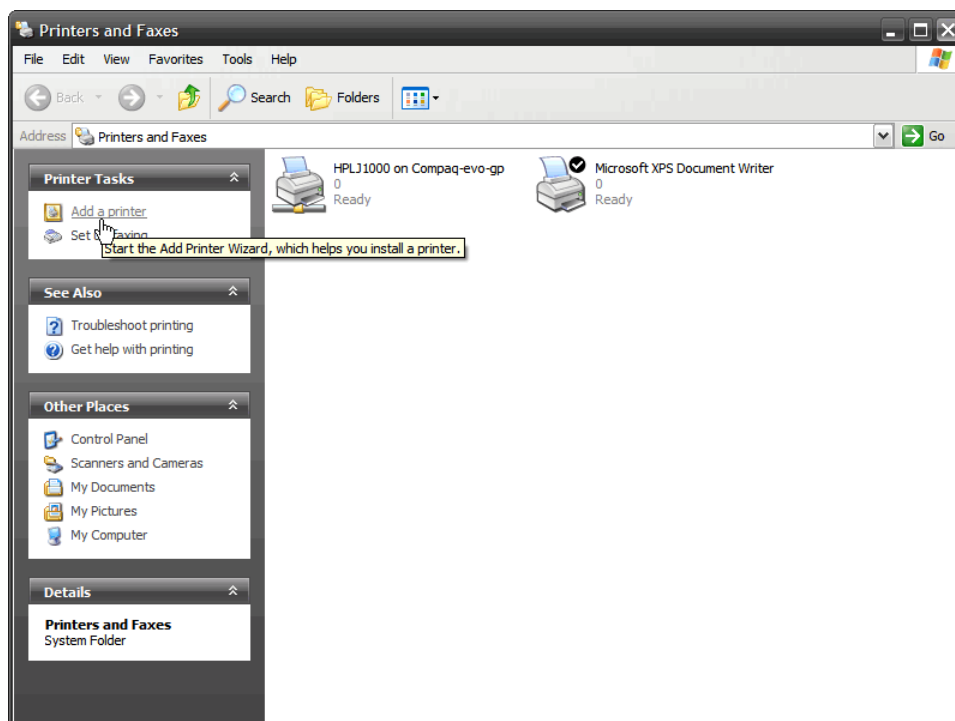
And, we're done ☺. Just print out a test page, to make sure everything is working. **Start → Settings → Control Panel → Printers and Faxes**, right click on your printer's icon, select **Properties**, then click on the **Print Test Page** button.



If we click on the **Printing Preferences...** button, we'll see all of the features of a manufacturer designated driver (logos, advanced options, etc.).



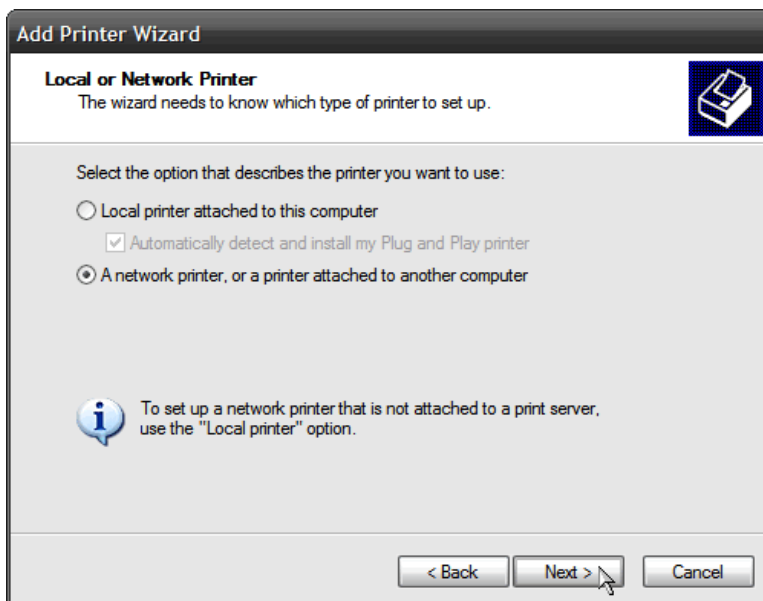
If for some reason, the SMB share method doesn't work for you or works slowly (hit print, then wait 30+ seconds for the printer to start printing, LOL ☺), we can add the printer using the HTTP protocol (yes, it's supported in Windows XP, not sure about Windows 2000 though). The method is almost the same as the one described in Windows 7, except for a few minor differences. Go to **Control Panel → Printers and Faxes** and click on the **Add a printer** link.



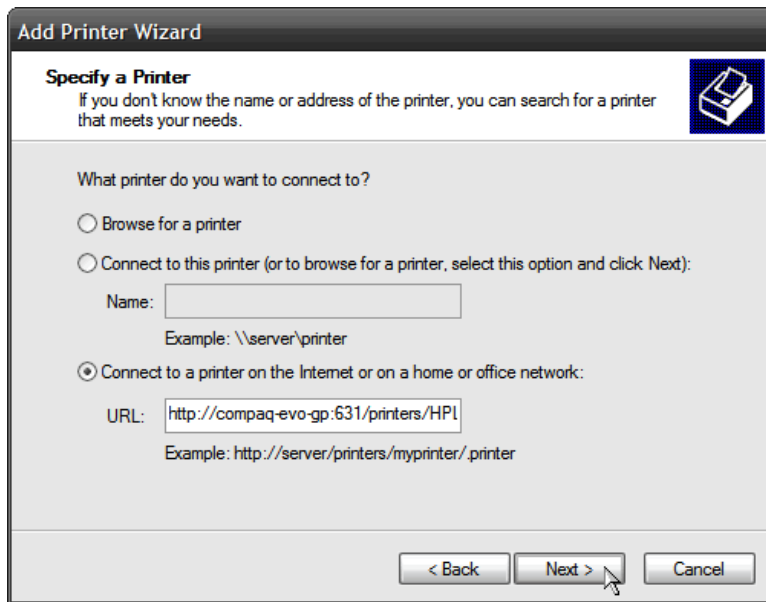
The **Add Printer Wizard** will pop up. Click the **Next** button.



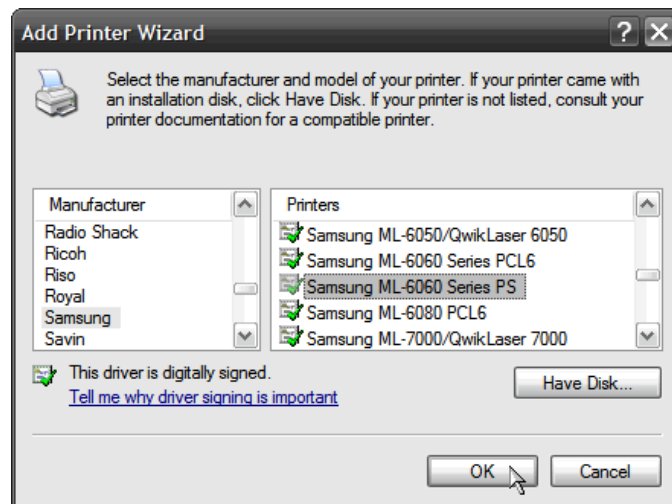
Select the **Add a network printer, or a printer attached to another computer** radio button. Afterwards, click on **Next**.



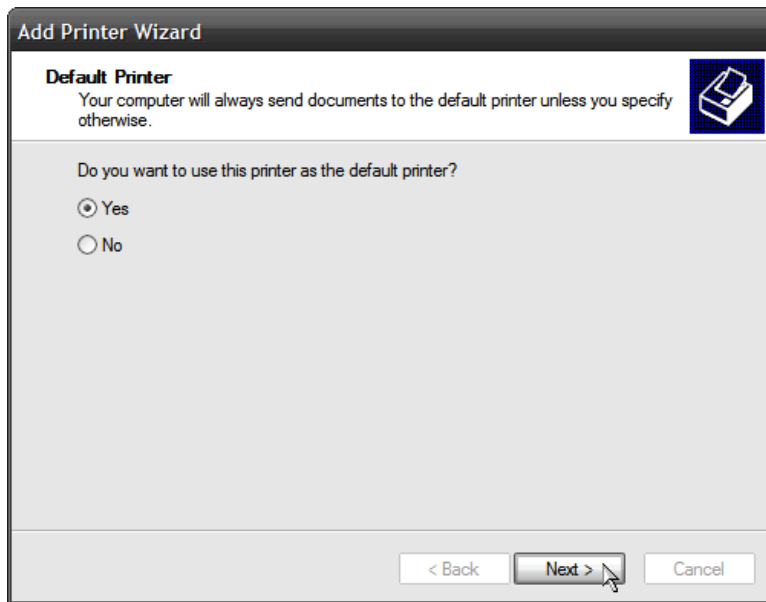
Next, select the **Connect to a printer on the Internet or on a home or office network** radio button. Afterwards, enter the HTTP address of the printer (as we did in Windows Vista/7): `http://<print_server_hostname>:631/printers/<printer_share_name>`. Alternatively, you can use the IP address: `http://<print_server_ip>:631/printers/<printer_share_name>`. Afterwards, click on the **Next** button.



Windows will ask you to either choose from a list of drivers or provide the drivers for the printer (the **Have Disk...** button). As with the example in Windows Vista/7, we'll choose a PS (PostScript) driver for the printer, since that's a language that CUPS understands and can communicate to the printer using PS. I've already mentioned that I've had most success with Samsung's ML series PS drivers (regarding zoom settings), but basically any PS driver will do. Some might print a bit zoomed out or zoomed in, but they will work. In this case, I'll choose the **Samsung ML-6060 Series PS** driver. Afterwards, click the **OK** button.



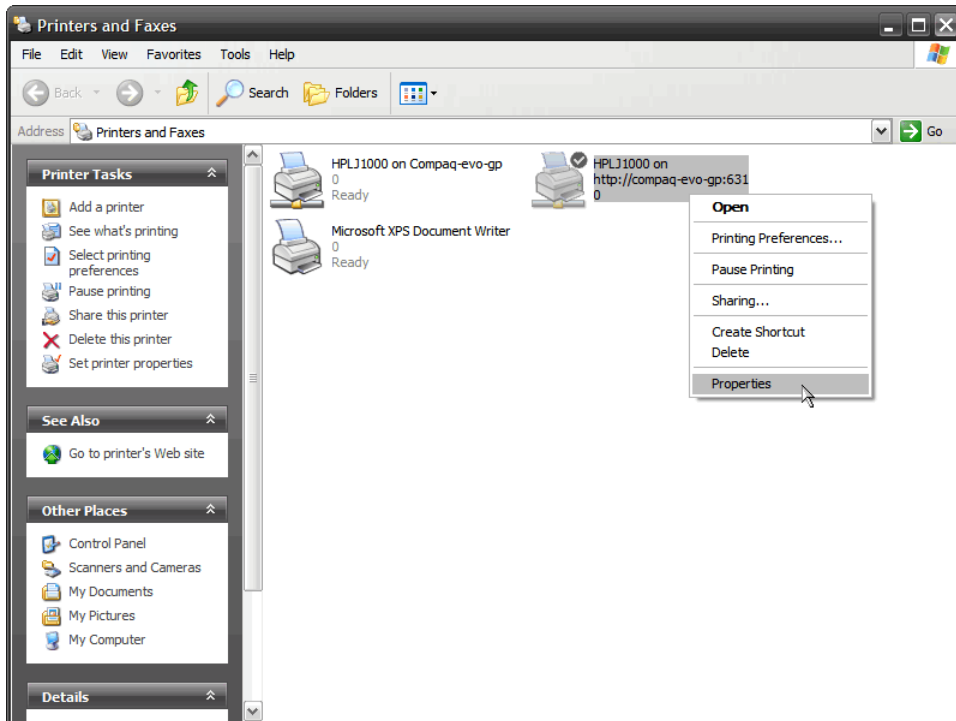
The **Add Printer Wizard** will now ask you if you'd like to use this printer as the default printer. Choose whatever you'd prefer, I'll choose **Yes**. Click the **Next** button afterwards.



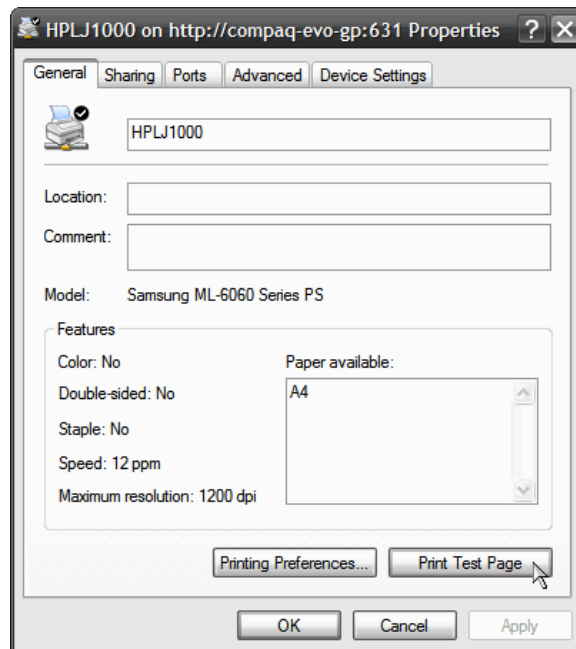
And to finish the process, click the **Finish** button.



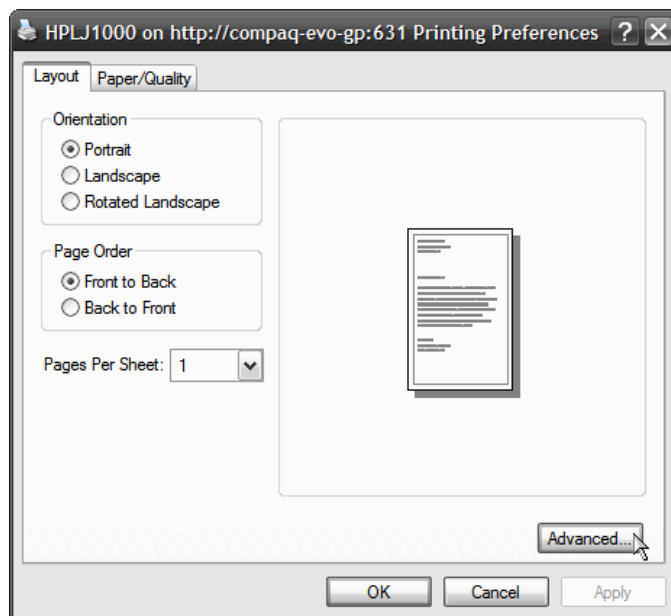
Next, we'd like to print a test page, to see if the driver is working. After the **Add Printer Wizard** finishes adding the printer, it'll bring you back to the **Printers and Faxes** section in **Control Panel**. Find the printer we've just added, right click on it and choose **Properties**.



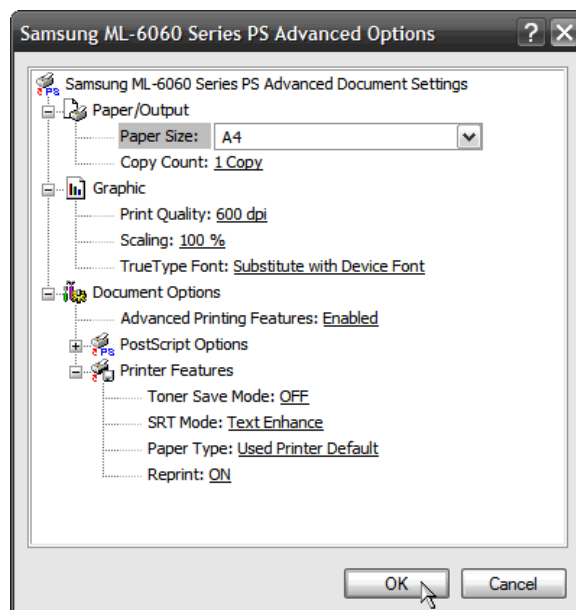
The printer properties page will appear. Click on the **Print Test Page** button. The printer should print out a test page.



Let's click on the **Printing Preferences** button and see what options are available in this driver.

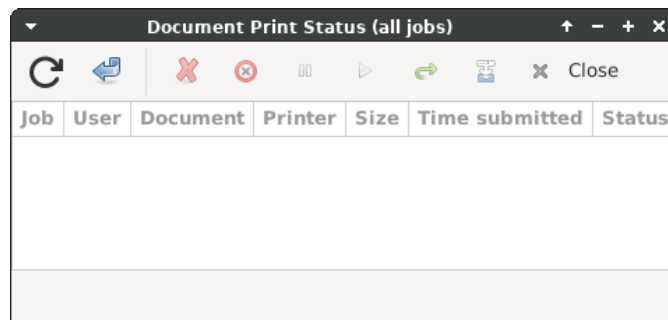
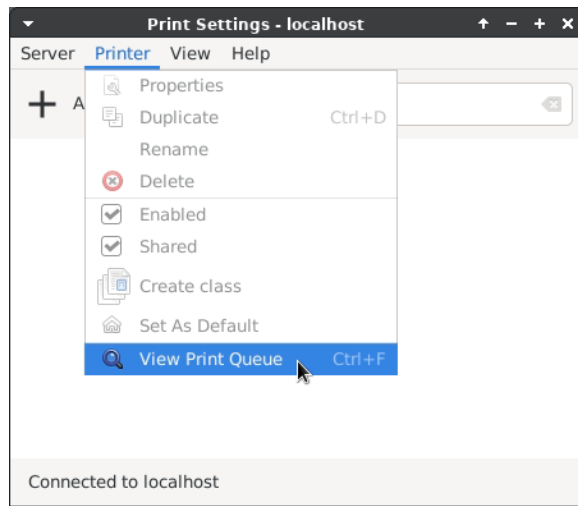


As in the Windows Vista/7 scenario, not a lot of options available. This is expected when using a generic PS driver. Let's see what the **Advanced** button offers us.

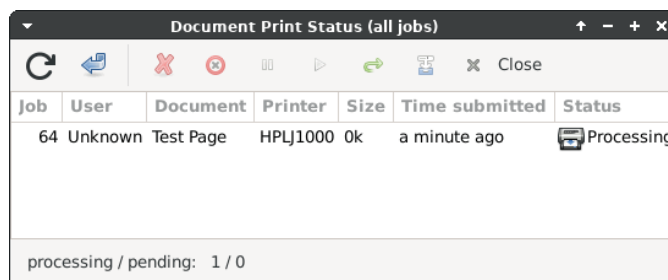


Once again, not a lot of options. Set up your paper size and click the **OK** button afterwards. When the **Advanced Printing Preferences** window brings you back to the **Printing Preferences** window, click the **OK** button again, and that will bring you back to the **Printer Properties** window. Click the **OK** button and we're done installing and setting up the printer. You can click the **X** button on the **Printers and Faxes** window in **Control Panel** now.

One more thing I'd like to add (that I forgot to mention earlier in this tutorial, LOL ☺) is to check the printer's print queue when testing the Windows drivers. Open up the **Print Settings** UI on the print server and go to **Printer** → **View Print Queue**.



Print jobs sent from client computers will show up here. So, if a driver in Windows is somehow misconfigured or just out right doesn't work with this setup, no print jobs will show here. Let's see how the print queue looks when we send a print job to the print server and the print setup does work.



As expected, the print queue reports there's a test page being sent to the printer 😊. You can leave the print queue open when testing print drivers. I usually use a remote desktop client running on the print server (the Linux install, usually AnyDesk) so I don't have to switch monitors or chairs when configuring the print server, just switch the application on the rig you're configuring the print server from to see the print queue 😊.

Also, note the **Document** field in the window above. It reads **Test Page**. This basically means that CUPS understands what this documents is, so in this case, we can be pretty sure that the output (the printed page) is not garbled or raw data. If this fields reads **Unknown**, this may be a sign of a misconfigured Windows driver (maybe you chose a PCL driver instead of a PS one, maybe the PS driver you chose is somehow *special* and just doesn't work as a regular PS driver, etc.). If the printer prints out garbled or raw data, just change the PS driver in Windows, try a different driver from the same manufacturer (or a different manufacturer) or use the generic Microsoft PS Class driver (if it's available on your Windows install), I'm sure you'll eventually find a PS driver that works for you 😊.

Well, this about wraps up this tutorial 😊. Any comments or suggestions, write me at 0x4e4f@gmail.com or on Reddit ([u/PCChipsM922U](https://www.reddit.com/user/PCChipsM922U)).