PrinterDetect (for TrakCare&CUPS)

Version History

* 20131104 – additional diagnostic/debugging info, Glen Pitt-Pladdy (InterSystems)
* 20130923 – updates to tools, Glen Pitt-Pladdy (InterSystems)
* 20130809 – initial version, Glen Pitt-Pladdy (InterSystems)
* 20131121 – updated docs, Glen Pitt-Pladdy (InterSystems)

# Important

These tools are developed for the needs of the TrakCare UK Infrastructure team and reflect conventions and approaches used to meet the needs of deployments in the UK. They are released on the basis that they may be useful to others within InterSystems.

The development process is on an ad-hoc basis, generally based on adding or updating tools as needed for deployments. There is negligible testing and no QD process nor any sort of managed release process. These are simply rolling snapshots of work in progress and may be in a severely broken state.

Use at your own risk.

# Purpose

Even the smallest TrakCare deployments may have a few hundred printers to manage. Initial configuration of these in CUPS is an onerous task to do manually.

To speed up this process this tool was developed to take a list of printers (.csv format), test, detect, and output scripts to add/remove these printers from CUPS rapidly. This helps sites prepare their printers with minimal assistance and minimizes the impact of printer configuration on deployment projects.

# Printer Detection Script

PrinterDetect.pl <input file.csv> <output prefix>

This is a Perl script that takes a .csv file containing at minimum the printer IP addresses (or hostnames), and outputs a set of files prefixed with the “output prefix”:

* .log – a log of activity (also printed to the terminal)
* -printeradd.sh – a shell script to add all the printers detected to the local CUPS instance
* -printerdel.sh – a shell script to delete all the printers detected from the local CUPS instance
* -errors.csv – a replica of the lines from the input file which failed with an added “Error” column
* -TrakPrinterImport.tsv – a Tab Separated Value file for importing printers into TrakCare (**UNTESTED – work in progress!**)

In the top of the script there are hashes for mapping .csv columns to CUPS parameters as well as drivers to different Printer ID strings.

## Dependencies

The script makes use of many system tools to pick up all the information necessary:

* lpinfo – Part of CUPS
* snmpget – Part (client tools) of Net-SNMP, used for probing SNMP interfaces on printers
* snmpwalk – Part (client tools) of Net-SNMP, used for probing SNMP interfaces on printers
* w3m – Text mode web browser, used for probing HTTP interfaces on printers
* wget – HTTP grabber, used for probing HTTP interfaces on printers
* curl – HTTP grabber, used for probing HTTP interfaces on printers
* PrinterCommon.pm – this is a Perl Module of procedures used by tools

## Detection Process

1. Ping – the printer IP is pinged to establish if it is available
2. The printer IP is probed with first snmp, then http, then tries the “model” field from the .csv to detect the printer model
3. Model string is matched against "lpinfo -m" (overridden by %DRIVERMAP) to establish the most appropriate driver
4. Tests TCP connections to ports 9100 and 515 to establish best method to print
5. Outputs commands to add/delete printer based on naming/fields from %FIELDMAP

## Example

An example file “PrinterDetect-ISCINTERNALExamle.csv” is provided containing printers available on the internal UK (mainly Windsor) networks.

To test this try something like:

# ./PrinterDetect.pl PrinterDetect-ISCINTERNALExample.csv /tmp/testprinterdetect

This should produce a set of files in /tmp/ with prefix “testprinterdetect” that contain the detected info. These include “add” and “del” scripts which can then be run on all systems which need these printers adding / removing.

## Input CSV File

The **first row of the file is always treated as column headers** – ensure there is something valid for it.

Technically the input .csv file only needs a single column with the IP address or hostname of the printer and will detect and generate all the other information where possible. For completeness the following information should be available in columns:

* IP address
* Model (for fallback)
* Name (if specific names are needed)
* Location (for CUPS, else blank)
* Description (for CUPS, else Model is used)
* TODO Multi-Tray (if an instance per tray is needed)
* TODO additional paper sizes

The exact column name mapping can be set in the script.

### Column Mapping (%FIELDMAP)

Fields used for establishing the model and adding the printer are set in the top of the script. Possible items to set are:

* ip – The IP address of the printer
* model – Optional column which is used for the model string when probing fails
* name – Optional column which contains the name that should be used for the printer, otherwise reverse-DNS, or the IP Address is used to generate a name
* description – Optional column containing the string used for the CUPS description, otherwise the model string is used
* location – Optional column containing the string used for the CUPS location, otherwise it is left blank

For example:

my %FIELDMAP = (

'ip' => 'IP address',

'model' => 'Model',

# CUPS fields

'name' => 'CUPS Name',

'description' => 'CUPS Description',

'location' => 'Comment',

);

## Debugging errors

There are some errors which are quite obvious but others that can be more subtle to debug. As well as the -errors.csv output file also see the .log file for more detailed info on the particular printers.

### Connectivity Problems

These often turn out to be as simple as printers turned off or disconnected, but where multiple sites / subnets exist there may be more complex routing and/or firewalling at play.

The simplest thing to check is if you can ping the address from the server:

ping <address>

On Unix you would typically use Control-C to stop the ping. Other tools that may be useful if this fails are traceroute / tracepath and other network diagnostics tools that may be used across the network to diagnose problems.

Ping only tests with ICMP Type 8 (echo request / ping) and response of ICMP Type 0 (echo reply / pong), however other ICMP types may be needed for stable operation, as well as UDP for SNMP detection / management and TCP for HTTP detection / management as well as printing.

### SNMP Problems

A quick check here is to run snmpwalk against the printer and look at the output:

snmpwalk -v1 -c public <address>

Optionally you may want to pipe this into "less" as it normally produces a lot of output.

The OIDs that are used for detection are:

* HOST-RESOURCES-MIB::hrDeviceDescr.1
* SNMPv2-SMI::mib-2.43.5.1.1.16.1
* SNMPv2-SMI::mib-2.43.14.1.1.9.1.7
* SNMPv2-SMI::mib-2.43.14.1.1.9.1.8
* SNMPv2-MIB::sysDescr.0

Generic printer servers (eg. Network to USB) will not have any useful information in these fields normally, though it may be possible to set some fields via the configuration for the print server.

If timeouts or other network errors occur then these need further investigation. SNMP normally uses UDP on port 161.

### HTTP Problems

HTTP detection is far less reliable as the format and URLs of pages varies massively between different models and even different versions of firmware with the same model. A general check would be to put the address into a browser and verify connectivity, but otherwise any improvements to detection to handle new printer models would likely have to be done programmatically in the detection script.

HTTP normally uses TCP on port 80, however other port numbers (81, 8080) are also seen on some devices as is SSL (port 443) depending on the particular printer.

### Print Connectivity Problems

Printing almost always uses Appsocket (TCP on port 9100) and a quick and easy test for this would be to telnet to the port from the server:

telnet <address> 9100

If that fails to connect then it indicates it isn't possible to make TCP connections to the port. This may be due to firewalls, configuration of the print server, connectivity or a number of other reasons. If connection occurs then you will normally need to use the escape sequence to exit telnet (on Unix normally: Control-], then type: quit<enter>)

If LPR is used (rare) these tools can also use that which would be TCP on port 515.

## Custom Driver Mapping (%DRIVERMAP)

Overrides and additions for the Model⬄Driver mapping from “lpinfo -m” can be set manually. All printer model strings are converted to lower case, but for completeness it’s worth pasting in the exact model string that is detected in case changes in behavior to the script occur at a later date and the case sensitivity is used for some detection.

When new driver mapping is added on-site, it should be reviewed and incorporated into the master script so that future uses benefit from these updates.

This is also needed for setting “raw” drivers used by Zebra label printers.

For example:

my %DRIVERMAP = (

# HP

'HP LaserJet p2015' => 'drv:///hp/hpcups.drv/hp-laserjet\_p2015\_series-pcl3.ppd',

'HP LaserJet p2015 Series' => 'drv:///hp/hpcups.drv/hp-laserjet\_p2015\_series-pcl3.ppd',

# 'HP LaserJet 400 M401dn' => 'TODO', # doesn't seem to be supported in standard distro

'hp LaserJet 1300' => 'drv:///hp/hpcups.drv/hp-laserjet\_1300n-pcl3.ppd',

# Ricoh

# 'RICOH Aficio MP 2851' => 'TODO', # doesn't seem to be supported in standard distro

'RICOH Aficio MP 5002' => 'foomatic-db-ppds/Ricoh/PXL/Ricoh-Aficio\_MP\_5000\_PXL.ppd.gz',

'RICOH Aficio MP C4501' => 'foomatic-db-ppds/Ricoh/PXL/Ricoh-Aficio\_MP\_C4500\_PXL.ppd.gz',

# Zebra (label)

'Zebra Printer' => 'raw',

'ZebraNet PrintServer' => 'raw',

);

## Error types

The script produces a file named <output prefix>-errors.csv which contains failed lines from the input .csv file with an added “Detect Error” column. This contains a colon (:) separated list of errors encountered that prevented detection of the printer on this line.

Errors you are likely to encounter:

* No Ping – could not ping the printer: it’s not accessible on the network
* No SNMP – no SNMP connection was possible
* No Match from SNMP – printer model was not matched from SNMP
* No ID from SNMP – none of the identifying OIDs were available from SNMP
* No HTTP Connect – it was not possible to connect to the HTTP (80) port on the printer
* HTTP Bad “…” – w3m couldn’t load the URL
* HTTP Failed “…” – wget/curl failed to load the URL
* No Match from HTTP – no matching driver was fround using HTTP probes
* No Appsocket (TCP 9100) – failed test connect to Appsocket
* No LPR (TCP 515) – failed test connect to LPR
* No Print Port – failed to find a usable port for printing
* Unmatched Driver – no probing method (or data from .csv file) produced a matching driver for the printer

# Printer Set Tray Script

PrinterSetTray.pl <input file.csv> <output prefix>

Once printers have been added, this script may be run to set trays according to the configured column.

## Dependencies

The script makes use of many system tools to pick up all the information necessary:

* lpoptions – Part of CUPS
* PrinterCommon.pm – this is a Perl Module of procedures used by tools

## Example

An example file “PrinterDetect-ISCINTERNALExamle.csv” is provided containing printers available on the internal UK (mainly Windsor) networks.

To test this try something like:

# ./PrinterSetTray.pl PrinterDetect-ISCINTERNALExample.csv /tmp/testprintersettray

This should produce a set of files in /tmp/ with prefix “testprintersettray” that contain logs as well as failed printers.