NINJa Configuration

Academic Information Systems
Columbia University
8/2005

0. Introduction

The NINJa printing system was developed to meet the challenges of managing a university printing environment. It offers cost recovery, reduction of waste, and usage tracking.

Conceptually, the NINJa solution begins with a release station kiosk atop each printer. A user sends a job to the printer, which then appears in a queue on the kiosk display. To print, the user selects his/her job from the queue and authenticates. If the user is authorized to print, the station forwards the job to the printer, the job is printed, and the user's printing quota is deducted.

The main components of the system are: the printer; the release station/print server atop the printer; the authorization/page counting server; and the database.

NINJa is flexible in that it can work under various configurations. Since this is an initial installation-and-setup document, the descriptions here are representative of the specific setup at Columbia University. Feel free to make sensible modifications in your configuration, but it may be easier at the outset if you could follow the descriptions below.

1. System Requirements

Release station/print server

A basic Linux box is recommended for your NINJa release station/print server with the following configurations:

- Recommended Hardware
  - CPU (minimum recommended Pentium II, 150 MHZ, 128 MB RAM)
  - Keyboard
  - Monitor
  - HP LaserJet Printer (Tested with 5Si/4000 Family/8000 Family)
  - Networking

- Minimum Software Requirements
  - Linux (Various flavors of Linux should work. We have run it on Redhat 7.2 and a custom built Linux-from-scratch)
  - Java Runtime 1.4 (Java 1.5 currently being tested)
  - A recent X environment (X.Org 6.8.2+ or XFree86 4.4.0+)
  - LPRng 3.8.28
  - ifhp 3.5.20 (dependencies include Net::SNMP, net-snmp, foomatic_filters, and a2ps)
  - Kerberos 5 (Tested with MIT Kerberos 5 Release 1.4.0)
  - ntp 4.2.0
  - NINJa web files: see section below

Additional Servers

In addition to your release station servers,
- You need a (web) server from which the NINJa station will download its files from
- You also need a server on which to run your page control accounting software

2. Compile from Source

You may choose to compile the NINJa source code yourself. If you want to use the provided compiled files, you can skip this section.

Requirements

i. j2sdk-1.4.0 or greater
ii. ant from http://ant.apache.org

To build and install, unpack the file **ninja-0.10-src.tar.gz** in to a directory (e.g. `SRCDIR`) and follow the guidelines below (To install w/o building, skip to the next section):

1. `cd SRCDIR/edu/columbia/print/ninja/properties`
2. edit appropriate files for your environment (i.e. unix, windows)
    - The *src* variable should be the directory where you unpacked the source
    - The *build* and *www* variables can be determined by the builder
3. `cd ../core`
4. edit *build.xml*
    - The property *env.USERDOMAIN* should be the prefix of one of the property files in the ../properties directory
    - The property *env.TMP* should be a temporary directory on your system
    - You may also need to edit the *dist.dir* property, depending on your system
    - You should make sure that slashes are forward- if you are building on *nix and back- if you are building on windows.
    - You may need to explicitly state the path of *jarsigner* where applicable
5. `ant all`
6. `cd ../prix`
7. edit *build.xml*
    - The property *env.USERDOMAIN* should be the prefix of one of the property files in the ../properties directory
    - The property *env.TMP* should be a temporary directory on your system
    - You may also need to edit the *dist.dir* property, depending on your system
    - You should make sure that slashes are forward- if you are building on *nix and back- if you are building on windows.
    - You may need to explicitly state the path of *jarsigner* where applicable
8. `ant all`
9. To install,
    - `cd /homedir/htmldir/ninja`, where `/homedir/htmldir` was the specified *www* dir from the properties environment file you edited in step 2.
    - These are the files to be installed on the webserver (for the ninja to download its files from):

    - cybernode-dl.jar, cybernode-ui.jar, ninja-dl.jar, ninja.war, prix-dl.jar, prix-install.jar, prix.jar, prix.war, rio-dl.jar
- One way of first building everything would be to attempt to install from **ninja-0.10.tar.gz** (not from source) using the installation guidelines in the NINJa Configuration Manual (the next section below). Following the installation, you may replace components with pieces that you have compiled.

3. Installation

i. Web files (on webserver)

To install NINJa, start by downloading the file **ninja-0.10.tar.gz** (even if you compiled from source). The contents of this tar file are described below. Before proceeding to such descriptions, we briefly mention some setup notes.

Your release station can be set up so it installs (either initially or regularly) much of its files via download from a remote webserver. So, the tar file ninja-0.10.tar.gz should be untar'd or installed on your webserver.

The following is an overview of the different branches within this web tree:

a. /printers/ninja/prod/prix/code

The NINJa code is installed in `printers/ninja/prod/prix/code/` via jar files and a war file. They are listed below:

```
cybernode-dl.jar
cybernode-ui.jar
ninja-dl.jar
ninja.war
prix.war
prix.jar
prix-dl.jar
prix-install.jar
rio-dl.jar
```

If you make code changes and compile from source (following instructions in the previous section), this is a place where you would install results of your build.

b. /printers/ninja/prod/prix/html

The html in `printers/ninja/prod/prix/html` can be modified to tailor the GUI display of a specific deployment.

The file `printers/ninja/prod/prix/html/index.html` should be modified to point to your webserver to get the css style sheet and the logo image.

c. /printers/ninja/prod/prix/security

The files in `printers/ninja/prod/prix/security` can be modified to tailor access to a specific deployment. The sample *java.net.SocketPermission* entries should be replaced.

d. /printers/ninja/prod/prix/jnlp

The files in `printers/ninja/prod/prix/jnlp` configure tells *javaws* what to do and how to do it.

In the files `printers/ninja/prod/prix/jnlp/prix-install.jnlp` edit the *codebase* line to point to your webserver.

For the file `printers/ninja/prod/prix/jnlp/prix.jnlp` replace your webserver in the following properties:
java.security.policy
java.security.auth.login.config
com.sun.rio.service.codebase
prix.laf.base

The property *com.sun.rio.jini.lookup.locator* is optional. There will be a later attempt at documenting this.

The property prix.pcd.uri should point to your accounting server and port that is listening for connections. For example
<property name="prix.pcd.uri" value="pcd2://pcd.cc.columbia.edu:7775"/>

e. /printers/ninja/prod/js/printers.js

The setup in `printers/ninja/prod/js/printers.js` provides a means to designate printer models. For example, this file can be used for assigning the *ifhp* filter, as well as generating web page with a list of your printers.

f. /printer/js/ninja/ifhp.js

The file `printers/js/ninja/ifhp.js` provides mappings for HP printers and *ifhp*. This is specific for an environment running HP printers. It is referenced when generating the printcap on the release station.

g. htaccess files

.htaccess files have been included within the tree. Make sure that you edit these files appropriately to provide correct access restrictions. Specifically, the NINJa station needs access to theses files.

ii. Java Web Start

Verify that the *Java Runtime Environment* is installed on your print server release station. If the *Java Runtime Environment* does not automatically install *Java Web Start*, locate the installed java directory on the NINJa station (e.g. `/usr/java/j2re`) and run the install script to install *Java Web Start* (e.g. `$ /usr/java/j2re/install.sh`).

Additionally, to configure *javaws*, it helps to have a *javaws.cfg* file in `/root/.javaws` (where `/root` is the root user's `$HOME`).

4. Configuration

Prior to starting up NINJa, you have to make changes so that it works with your specific configuration. Note that some of the steps below are not required to use NINJa. However, we describe our configuration as an example.

i. Setup of GUI/X environment

A good place to start is to take a look at these two scripts:

- **start_prixc.sh**
- **prixc.sh**

We have them installed on the release station in `/usr/local/bin`. The script *start_prixc.sh* is set to run upon startup. It starts X and calls *prixc.sh*. The script *prixc.sh* configures print queues and the *printcap*, and calls *javaws* to bring up the GUI.

ii. Print Queues (on release station)

A good place to start is to look at this **sample /etc/printcap**

Spool directories

Set up three queues (for example, *public*, *mediator*, and *private*). Here is a **sample /etc/printcap**

You can set up the directories
`/spare/spool/public`
`/spare/spool/mediator`
`/spare/spool/private`
as the queues that the */etc/printcap* references.

Printcap

There are several parameters that must be specified in the *printcap* on the release station. As an alternative to following the sample, the LPRngTool (found at http://www.lprng.com) can aid in the creation of the *printcap*.

Otherwise, the following is an overview of what is laid out in the sample *printcap*:

The *public* queue is setup so that any user may print to it. This queue is where the NINJa station gets its print jobs from. The *mediator* queue sits between the *public* and *private* queues. The *private* queue should only be accessible from the local machine. This is the queue that is connected to the printer.

For all three queues (*public, mediator, private*),

- `filter=[path to ifhp filter]` – e.g. `/usr/lib/lpfilters/ifhp` or `/usr/libexec/filters/ifhp` (from Redhat, "`rpm -ql ifhp`" to locate the *ifhp* filter)

For the *public* queue,

- `@ah` – Do not hold jobs
- `lp=mediator@localhost` – Sends jobs to the mediator queue

For the *mediator* queue,
- `ah` – autohold jobs
- `lp=/dev/null`

For the private queue,
- `ifhp=model=[PRINTER MODEL ]` – This is the LaserJet model (e.g. hp8000)
- `lp=[printer hostname]%[port]`

To check if the queues are working properly, send a job to the release station. To do so, setup a client to print to the station. (Sample instructions for setting up a client to print to your station can be found at http://www.columbia.edu/acis/access/printing.) NINJa only recognizes PostScript data, so you must use a driver that generates PostScript data (e.g. *HP LaserJet 5Si/5Si MX PS* or *HP LaserJet 8000 PS*).

Once a client has been configured to print to your NINJa station's *public* queue, send a print job to it. Both generated *df* and *hf* files should be noticeable in the *public* queue's directory (e.g. `/spare/spool/public`). The processed postscript file for the job should be forwarded automatically to a temporary directory (e.g. `/tmp/queue`).

iii. jnlp

- Locate and modify the *prix.jnlp* file. This file should be located in `printers/ninja/prix/jnlp`. This file is specified on startup of NINJa and is installed on the webserver. The following will assume that `[ web_files_dir ]` is `printers/ninja/prod/prix` on your webserver and that `[web_files_url]` is the url for that directory.

  Edit `[ web_files_dir ]/jnlp/prix.jnlp`

  See here for a **sample prix.jnlp** file.

  Modify the codebase for the jnlp entry:
  ```
  <jnlp spec="1.0+" codebase="[web_files_url]"
  href="jnlp/prix.jnlp">
  ```

  Correct the codebases of the following properties:
  ```
  java.security.policy = [web_files_url]/security/policy.prix
  java.security.auth.login.config = [web_files_url]/security/login.config
  com.sun.rio.service.codebase = [web_files_url]/code
  prix.laf.base = [web_files_url]/html
  ```

  Modify the following properties:
  `prix.pcd.uri = pcd://[full hostname]:[port]` – this is where you specify your page control daemon or server (see step "vii Page Accounting")
  `prix.spool.directory =` Absolute path of input queue spool directory

`prix.inputPrinter` = Name of input queue (e.g. "mediator")
`prix.outputPrinters` = Name of output queue (e.g. "private")
`prix.spool.directory` = Name of spool directory (e.g. "/spare/spool/mediator")
`ninja.queue.dir` = storage location for NINJa files (This is an arbitrary location such as /tmp/queue)
`ninja.address.organization` = arbitrary single-word name

- Locate and modify the *prix-install.jnlp* file. This file is specified on startup of NINJa and is installed on the web server.

  See here for a **sample prix-install.jnlp** file.

  Edit `[ web_files_dir ]/jnlp/prix-install.jnlp`

  Modify the codebase for the jnlp entry:
  `<jnlp spec="1.0+" codebase="[web_files_url]" href="jnlp/prix-install.jnlp">`

- Modify the security policy for the NINJa

  See here for a **sample policy.prix** file.

  Edit `[ web_files_dir ]/security/policy.prix`

  Modify the *SocketPermissions* to enable/disable access to and from specific hosts and ports

iv. lpd

In your *lpd.conf* (e.g. `/etc/lpd/lpd.conf`) add or uncomment the property
`prefix_option_to_option=OS Z`

You may also need to update the *lpd.perms* file (e.g. `/etc/lpd/lpd.perms`) to control jobs. Here is a **sample lpd.perms** file.

v. Kerberos

Make sure Kerberos is set up properly for your domain. Set up your *krb5.conf* file (e.g. `/etc/krb5.conf`). More documentation is available at http://web.mit.edu/kerberos/www/.

vi. ifhp

If you do not have HP printers, you can skip this step. Depending on what models of printers you have, you may have to do some added configuration. There are workarounds (not yet documented here) to set up HP 9000s (not in the most recent release of ifhp).

vii. Page Accounting

This is a fairly important step. The print station is not configured to do page accounting. Instead, it contacts a server process that listens for requests from NINJa and contains the logic to authorize print jobs as well as debit printing accounts. For example, NINJa sends a string with a username and number of pages to the accounting server. In turn, the accounting server returns a response approving or rejecting the request.

Your page accounting server is specified in the jnlp configuration (see step "iii jnlp").

You can find out more about Columbia University's page accounting system when we actually post some documentation. The accounting system used for printing at Columbia University is called APRICOT, implemented as "pcd" (page control daemon). It is available for download at <http://www.columbia.edu/acis/dev/projects/ninja/dl/apricot-current.tgz>.

5. Test

- Start your page accounting server
- Start the NINJa Client

```
ninja_client$ [java_home]/javaws/javaws [web_files_url]/jnlp/prix.jnlp
```

Alternatively, you can refer to or use an automated script to configure the environment at the same time. See this aforementioned **prixc.sh script** (which can be setup to be automatically run in something like `/etc/X11/xinit/xinitrc`).

The *javaws* interface should show up on your display. Send a print job from the client that you configured to print to this station in the print queue installation step. This job should appear on your screen. The interface should be driven entirely by the keyboard. Select the job, authenticate, and wait for the print job to complete. The correct number of pages should be noted in your accounting system. If everything is working properly, you have successfully installed the NINJa printing system.

6. Additional Options

Once you have your NINJa station in place and working, there are additional options that you can setup on top of the stand-alone configuration.

Remote Monitoring and Configuration

[Documentation to come…]

Text to Postscript Conversion

[Documentation to come…]

7. Additional Configurations

Modifying NINJa skin

The skin is an html page with xml tags embedded within the page. You'll find the html in `[ web_files_dir ]/html`. A good way for viewing these are by using a standard html browser. These embedded tags are "hidden" by setting the font size to 0pt.

The dialog tag identifies the location of the dialog window. The location of the <dialog> begin and end tag indicates the bounds of the window.

The component tags identify components of skin. The location of the <component> begin and end tag indicates the bounds of the component of the skin. Additionally the component may have attributes such as "color" and "font-size."

If there are images in the html pages, these should have their width and height attributes specified in order for the components to be displayed properly.

- Edit `[web_files_dir]/html/*.html`
- To view updated html, type F1 and enter at the station

Configuring Distributions

NINJa stations can be grouped into multiple distributions. These can be organized at `[web_files_dir]`.
[Documentation to come…]

Helpful links

- LPRng – LPRng web page (http://www.lprng.org)
- RPM Find – RPM file locator for many packages (http://www.rpmfind.net)
- Red Hat Linux – Linux (http://www.redhat.com)
- Java Standard Edition – Download the Linux JRE (http://java.sun.com/j2se/1.4.2/download.html)

Additional System Configuration Notes

The following notes may be helpful in configuring NINJa:

- You can install LPRngTool to help configure the printcap
- For LPRng-3.8.28-1, you may need to install libcom_err.so.3
- There is a known incompatibility regarding: jdk1.5_01's javaws does not function with later version of glibc (2.3.3+), which is found in FedoraCore3.
- When setting up the queues, you may need to add the file status.pr in the spool directories

```
# touch /spare/spool/public/status.pr
# touch /spare/spool/mediator/status.pr
# touch /spare/spool/private/status.pr
```

- When a print job is successful, you should see *df* and *hf* files generated in the *private* queue directory and removed from the temporary spooling directory (e.g. `/tmp/queue`). Of course, you should also see the paper coming out of the printer.

- Rather than create a printcap for all printers manually, our script `/usr/local/bin/prixc.sh` automates the configuration in the printcap with the correct printer model. It references `/etc/printcap.local.src` to eventually create `/etc/printcap`, using
- The following files may be useful in debugging on the release station:
  - `/var/log/rmid`
  - `status.pr` in spool directories
  - *.java* and *.javaws* directories in `$HOME`
  - `lpq -L` or `lpq -v`
- The jar files may need to be jarsigned…
- To organize printers into various distributions, our release stations reference various javascript files on the webserver. In doing so, the script `/usr/local/bin/prixc.sh` uses **/files/js.jar**.
- More to come…