

# Realm Linux Training: Configuration Management with Bcfg2

Jack Neely  
linux@help.ncsu.edu

Campus Linux Services  
Office of Information Technology  
North Carolina State University

Thursday, July 21, 2011

Copyright © 2011 NC State University

Git Presentation Source:

Copyright © 2008 Karel Zak

Copyright © 2008 Tomas Janousek

Copyright © 2008 Florian Festi

Copyright © 2008 Bart Trojanowski

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Git Presentation Original Source: <http://kzak.fedorapeople.org/>

Source Code: <git://git.linux.ncsu.edu/bcfg2-training.git>

Please download this presentation and follow along.

<http://go.ncsu.edu/bcfg2-bcfg2>



# Table of Contents

- 1 Working with Bcfg2 Repositories
- 2 Conclusions

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
    - The Bcfg2 Data Model
    - Bcfg2 Client Operation
    - Working With Metadata
    - Building Abstract Configuration
    - Creating the Literal Specification
    - Review

# Bcfg2 Repositories in AFS

```
/afs/bp/system/config/linux-kickstart/bcfg2/<DEPT>
```

- Git Repositories of Bcfg2 Specification Live in AFS
- Normally, All Are Clones of the root Repo
- Servers Pull From These Repos Every 15 Minutes
- You Can Clone root and Make Your Own Bcfg2 Server

# Cloning root

```
$ git clone git://git.linux.ncsu.edu/bcfg2repo.git
Cloning into bcfg2repo...
remote: Counting objects: 1266, done.
remote: Compressing objects: 100% (773/773), done.
remote: Total 1266 (delta 570), reused 657 (delta 277)
Receiving objects: 100% (1266/1266), 157.45 KiB, done.
Resolving deltas: 100% (570/570), done.
```

- Clone the root Bcfg2 Repo – Let's Investigate

# Inside the Repository

```
$ ls -lF
total 52
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Base/
-rw-rw-r-- 1 slack slack 1054 Apr 25 17:16 bcfg2.conf
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Bundler/
drwxrwxr-x 4 slack slack 4096 Apr 25 17:16 Cfg/
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 etc/
-rw-rw-r-- 1 slack slack 562 Apr 25 17:16 GROUPS
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Metadata/
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Packages/
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Probes/
-rw-rw-r-- 1 slack slack 78 Apr 25 17:16 README
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 Rules/
drwxrwxr-x 3 slack slack 4096 Apr 25 17:16 TGenshi/
drwxrwxr-x 2 slack slack 4096 Apr 25 17:16 webkickstart/
```

The repository contains the configuration specification as well as various informative files and the server configuration.

For each repository there is a specific Bcfg2 Server Instance.

# Files - bcfg2.conf

- Server Instance Configuration (Not Clients)
- Created and Modified when Repository Was Cloned
- Probably Should Not be Altered
- Contains the Bcfg2 Password for Your Department
- Contains the URL/Port to Your Server Instance

```
[communication]
protocol = xmlrpc/ssl
password = foobar
key = /etc/bcfg2.key
certificate = /etc/bcfg2.crt

[components]
#####
# URL must have a unique port number
bcfg2 = https://cm.linux.ncsu.edu:6000
#####
```

# Files – GROUPS and README

- README – Notes. Used to Contain Something Interesting
- GROUPS – Listing of Defined Groups and Meanings

## Feature Groups

=====

```
feature-sssd -- When nss_ldap is replaced with SSSD
              From metaconfig
```

```
feature-selinux -- When you want to turn SELinux on
                  Specify manually
```

```
feature-tmpclean -- Aggressive /tmp cleaning
```

```
feature-support -- The Support Knob (turns on realm-cron)
```

```
feature-smartmail -- Enable smart hosting of mail to smtp.ncsu.edu.
                   Otherwise, host is responsible for sending
                   its own mail and is NOT masqueraded.
```

```
feature-nofirewall -- Do not manage /etc/sysconfig/iptables
```

# Directories

Directories in the Specification relate to Bcfg2 Server Plugins. Plugins provide specific functionality to the server.

**Probes** Scripts That Collect Information from the Client

**Metadata** Information about Groupings and Bundles

**webkickstart** Maps Web-Kickstart keywords to Bcfg2 Groups

**Bundles** Describes What is Managed and the Relationships Between. Like Restarting Apache When its Configuration Has Changed.

**Base** Unused

**TGenshi** Templating System to Generate Text Configuration Files

**Cfg** Plain Text Configuration Source (Superseded by TGenshi)

**Packages** Provides Information About Available RPM Packages.

**Rules** Resolves Symlinks, Directories, Actions, Services, Etc.

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - Bcfg2 Client Operation
  - Working With Metadata
  - Building Abstract Configuration
  - Creating the Literal Specification
  - Review

# Declarative Semantics

*“Declarative semantics maximize the utility of configuration management tools; they provide the most flexibility for the tool to determine the right course of action in any given situation. This means that users can focus on the task of describing the desired configuration, while leaving the task of transitioning clients states to the tool.” – Bcfg2 Documentation<sup>1</sup>*

---

<sup>1</sup><http://docs.bcfg2.org/architecture.html>

## 3 Types of Data

The Bcfg2 Repository contains 3 different types of Data.

- Metadata
- Abstract Configuration
- Literal Configuration (Sometimes Called "Binding")

# Metadata

Metadata is information about the specific client.

- Hostname
- Profile Group
- Other Groups
- Bundles
- Probes

A *Profile Group* is a special group that the Bcfg2 client directly binds to at initial registration. In our case the Profile Group defines the machine's base platform. Such as:

- `realmlinux-el6-i386`
- `realmlinux-EL5-x86_64`

# Abstract Configuration

Abstract Configuration is the list of objects that Bcfg2 will manage.

- File Names
- Services
- Packages
- Actions

This does not include specifics:

- No File Content or Permissions
- No Service States
- No Action Scriptlets

Abstract Configuration are primarily Bundles. (And Base if we used it.)

# Literal Configuration

Each abstract configuration item is bound to its client specific value.

- Rendered Templates
- Group Membership Can Affect Content
- Group Membership Can Affect Service States and Actions

# Plugins and Configuration Type

- Metadata
  - Probes
  - Metadata
  - webkickstart
- Abstract Configuration
  - Bundles
  - Base
- Literal Configuration
  - TGenshi
  - Cfg
  - Packages (Sorta...)
  - Rules

# Differences From Stock Bcfg2

Bcfg2, like any reasonable tool, wants to build a database of all of our clients. However, we have such a database in the Realm Linux Management Tools.

- Metadata/clients.xml Replaced
- Connection to RLMTTools
- RLMTTools Provides Much More Additional Metadata
  - UUIDs
  - Attributes
  - Departments
  - Link to Web-Kickstart Data

Bcfg2's Plugin Architecture allows us to easily import additional metadata and other sources of information about our deployment.

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - **Bcfg2 Client Operation**
  - Working With Metadata
  - Building Abstract Configuration
  - Creating the Literal Specification
  - Review

# Bcfg2 Client

- In Realm Linux, Run Every 4 Hours
- `/etc/cron.update/bcfg2.cron.sh`
- Run at Boot Time
- Run manually:

```
# bcfg2 -v
```

- Configuration Stored in `/etc/bcfg2.conf`
- Command Line Arguments Override Configuration File

```
# bcfg2 -v -S https://testbcfg2.linux.ncsu.edu:6000
```

# Tuning When and How the Bcfg2 Client Runs

`/etc/sysconfig/bcfg2`

- As Root, You Can Always Run Bcfg2 Manually
- Above File is Managed By Bcfg2
- Knobs For Hourly Running, Running on Boot

I have a very specific and well thought out reason why I want to turn Bcfg2 off. What's the right way to do that?

## Example

```
BCFG2_OPTIONS="-qvn"    # Default: "-qv"
```

Please do not disable Bcfg2 completely.

# Client Bootstrapping

How do Bcfg2 clients know the right Bcfg2 server instance, UUID, and password when these items are managed by Bcfg2?

Realm Linux 6 clients are bootstrapped into Bcfg2 during the Web-Kickstart %post scripts. If the bootstrap does not work you can manually bootstrap.

- 1 Register/Update Realm Linux Management Tools
- 2 Bootstrap the Bcfg2 Environment
- 3 Run Bcfg2 Again

```
# /usr/bin/ncsuclient  
# /usr/bin/ncsubootstrap -p <profile group>  
# /usr/sbin/bcfg2 -v
```

# Profile Groups

Profile groups define the base platform of the client. They should have a form like:

Product-Version-Architecture

Profile Groups (As of This Writing):

- `realmlinux-el6-i386`
- `realmlinux-el6-x86_64`
- `realmlinux-EL5-i386`
- `realmlinux-EL5-x86_64`

# Bcfg2 Client Process

Bcfg2 clients use a 4 step process to apply configuration.<sup>2</sup>

- 1 Download, Run Probes, Return Results
- 2 Configuration Download and Inventory
- 3 Configuration Update
- 4 Statistics Upload

---

<sup>2</sup><http://docs.bcfg2.org/architecture.html#the-bcfg2-client>

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - Bcfg2 Client Operation
  - **Working With Metadata**
  - Building Abstract Configuration
  - Creating the Literal Specification
  - Review

# Probes

Probes are scripts downloaded and executed on the Bcfg2 client. The results are added to the Metadata for the client.

- Any Useful String That Can Be Used By TGenshi Templates
- Assert Group Memberships

```
$ cat Probes/rlgroups
#!/bin/bash
# To assign Realm Linux machines running Bcfg2 to arbitrary groups with
# in your repository add the group name, one per line, to
#
# /etc/realmlinux-groups
#

G=/etc/realmlinux-groups
if [ -f $G ] ; then
    cat $G |
    while read line ; do
        echo "group: $line"
    done
fi
```

## Probes – Example 2

Examine Probes/hosers.

Its goal is to enforce the existence of `/etc/hosers.local` but the contents come from elsewhere.

```
#!/bin/bash
# Create a /etc/hosers.local based off what's already on the client

# If FILE doesn't exist just exit with no output
FILE=/etc/hosers.local
[ -f $FILE ] || exit 0

# Otherwise return the contents
cat $FILE
exit 0
```

A TGenshi Template uses the output as the literal contents of the managed file.

# Groupings

Groups for a client are asserted multiple ways.

- A Subgroup of the Profile Group – see `Metadata/groups.xml`
- Probes or the Client's `/etc/realmlinux-groups` File
- In RLMTools via the `bcfg2.groups` Attribute (Whitespace Delimited)

# Altering Metadata/groups.xml

If Groups should assert membership in other Groups or, importantly, new Bundles, they must define that in Metadata/groups.xml.

## Metadata/groups.xml Snippet

```
<Group name="realmlinux-el6-i386" profile="true" public="true" default="false">
  <Group name="realmlinux-el6"/>
  <Group name="i386"/>
</Group>
<Group profile="true" public="true" default="false" name="realmlinux-el6">
  <Group name="realmlinux"/>
  <Group name="rhel"/>
  <Group name="feature-sssds"/>
  <Group name="feature-gdm-2.30"/>
  <Group name="feature-gnome-clock"/>
</Group>
<Group name="realmlinux">
  <Bundle name="rlbase"/>
  <Bundle name="selinux"/>
  <Bundle name="krb5"/>
  <Bundle name="rlyum"/>
  <Bundle name="gpg-pubkeys"/>
  ...

```

# SysAdmin Tip: Altering Metadata/groups.xml

Use XInclude! The namespace in Metadata/groups.xml is already defined. Just use the `xi:include` tag to reference other files.

## Metadata/groups.xml Snippet

```
<Groups version='3.0' xmlns:xi="http://www.w3.org/2001/XInclude">  
...  
  <xi:include href="function-sftpd.xml" />
```

## Metadata/function-sftpd.xml

```
<Groups version='3.0' xmlns:xi="http://www.w3.org/2001/XInclude">  
  <Group name="function-sftpd" profile="false" public="true" default="false">  
    <Bundle name="sftpd"/>  
  </Group>  
</Groups>
```

# Naming Groups

Groups represent features or functions. Name them that way.

Realm Linux Groups:

- `feature-smartmail`
- `feature-gdm-2.30`
- `feature-gnome-clock`
- `function-sftpd`
- `function-web-sys-server`

Other Good Names:

- `print-server`
- `web-sys-server`

# Web-Kickstart Groups

Examine `webkickstart/groups.conf`.

A Realm Linux specific Plugin to convert Web-Kickstart keywords into Bcfg2 groups.

## `webkickstart/groups.conf` Snippet

```
enable.notmpclean : feature-notmpclean
```

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - Bcfg2 Client Operation
  - Working With Metadata
  - **Building Abstract Configuration**
  - Creating the Literal Specification
  - Review

# Bundler

Look in Bundler/.

- A Bundle<sup>3</sup> is a Set of Interdependent Configuration Entries
- Packages, Files, Services That Make Up a Unix Daemon or Service
- Abstract Configuration

## Bundler/ntp.xml

```
<Bundle name="ntp" version="2.0">
  <Path name="/etc/ntp.conf" />
  <Path name="/var/lib/ntp" />
  <Package name="ntp"/>
  <Service name="ntpd"/>
</Bundle>
```

This example would restart the NTP daemon if the `/etc/ntp.conf` file is updated.

---

<sup>3</sup><http://docs.bcfg2.org/server/plugins/structures/bundler>

# Groups in Bundles

## Bundler/cups.xml

```
<Bundle name="cups" version="2.0">
  <Service name="cups" />
  <Group name="feature-printing">
    <Path name="/etc/cups/cupsd.conf" />
    <Path name="/etc/rc.conf.d/HostPrinter" />
    <Path name="/usr/share/cups/model/postscript.ppd" />
    <Action name="cups-load-printers" />
    <Action name="cups-default-printer" />
  </Group>
</Bundle>
```

- Groups Affect Bundles
- Bundles Trigger Actions (See the Rules Plugin)

# Templates with Bundles

- Complex Cases May Require Using a Genshi Template.
- Used in Bundler/bcfg2.genshi to Make Decisions Based on RLMTTools Attribute Data
- Templated Bundles are Probably a Rare Case

## Contrived Example

```
<Bundle name="I-Make-No-Sense" xmlns:py="http://genshi.edgewall.org/">
  <?python
    files = $metadata.Probes["getmacs"].split("\n")
  ?>
  <Path py:for="file in files"
    name="/etc/sysconfig/network/ifcfg-eth-${file}"
    altsrc="/etc/ifcfg-template"/>
  <Package name="NetworkManager" py:if="'fedora' in metadata.groups"/>
</Bundle>
```

# Help with Packages

- Specifying a Package Means Bcfg2 Will Verify That RPM
- If the Verify Fails Bcfg2 Will Reinstall the RPM
- Sometimes the Mechanics in an RPM Will Re-Generate Files
- Use the Ignore Tag

## Bundler/sendmail.xml

```
<Bundle name="sendmail" version="2.0">
  <Path name="/etc/Mutttrc.local"/>
  <Path name="/etc/mail/sendmail.mc"/>
  <Path name="/etc/cron.hourly/local-mail-users-sync.sh"/>
  <Package name="sendmail-cf"/>

  <Package name="sendmail">
    <Ignore name="/etc/mail/sendmail.cf"/>
    <Ignore name="/etc/mail/submit.cf"/>
  </Package>

  <Action name="restart_sendmail"/>
</Bundle>
```

# Bundles Vs. Groups

## Groups Are:

- Features (Enable Printing)
- Functions (Specific Web Server Pool)
- Differences (Versions of GDM)
- Platforms (Realm Linux 8 x86\_128)

## Bundles Are:

- Logically Interrelated Files and Services
- Physical Configuration Elements

## Warning Signs:

- Multiple Bundles Referencing the Same File/Package
- Package in Bundle A Stepping on File in Bundle B
- File in Bundle A That Bundle B Ignores
- Unrelated Items in the Same Bundle

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - Bcfg2 Client Operation
  - Working With Metadata
  - Building Abstract Configuration
  - **Creating the Literal Specification**
  - Review

# Rules

Examine `Rules/realmlinux.xml`

## The Rules<sup>4</sup> Plugin

- Literal Configuration
- Define Service States
- Permissions
- Directories and Symlinks
- Actions
- Use Groups to Differentiate
- No XIncludes Needed, All Files in `Rules/` Processed

We do not use the Package functionality.

---

<sup>4</sup><http://docs.bcfg2.org/server/plugins/generators/rules.html>

## Rules: Using Groups with Services

If a Bundle references the Service `tmpclean` this Rules snippet figures out what state that service should be in.

### Rules/realmlinux.xml Snippet

```
<Service name="bcfg2" status="on" type="chkconfig"/>
...
<Group name="feature-nottmpclean">
  <Service name="tmpclean" status="off" type="chkconfig"/>
</Group>
<Group name="feature-nottmpclean" negate="true">
  <Service name="tmpclean" status="on" type="chkconfig"/>
</Group>
```

# Rules: Symlinks and Directories

**Don't Forget:** Reference the Path from a Bundle. The Rules plugin only resolves that reference.

## Rules/realmlinux.xml Snippet

```
<!-- Symlinks -->
<Path type="symlink" name="/etc/sysconfig/selinux"
      to="/etc/selinux/config"/>

<!-- Directories -->
<Path type="directory" name="/var/lib/ntp"
      owner="ntp" group="ntp" perms="0755" prune="false"/>
<Path type="directory" name="/etc/rc.conf.d"
      owner="root" group="root" perms="0755" prune="false"/>
```

Path Types Also Supported:

- device
- hardlink
- nonexistent

# Rules: Actions

This Action<sup>5</sup> example loads GConf settings when the XML files are updated or added.

## Rules/realmlinux.xml Snippet

```
<Action timing="post" when="modified" name="load_gconf"  
  command="for f in `ls /usr/share/realmlinux/data/*.xml` ; do \  
    /usr/bin/gconftool-2 --direct --config-source \  
    xml:readwrite:/etc/gconf/gconf.xml.defaults --load $f ; \  
  done"  
  status="check" />
```

- **timing** – Controls When the Action is Executed with Respect to Make Changes on Disk
- **when** – Only When Modifications are Made or Every Run
- **status** – Whether to Report Success/Failure

---

<sup>5</sup><http://docs.bcfg2.org/client/tools/actions.html>

# The Filesystem

Check out the Cfg/ and TGenshi/ directories.

- 2 Similar Plugins Resolve Literal File Content
- Inside Each Directory Is a Directory Tree That Mirrors an Actual Filesystem
- Both Support a Mechanism to Resolve File Content According to Groups and Hostnames
- Both Support Setting Ownership and Permissions on These Files
- Only One Should Resolve a Specific File
- Expect These Plugins to Merge
- Bcfg2 Also Supports a TCheetah Plugin That We Do Not Use

**SysAdmin Tip:** Realm Linux bits currently use both plugins. For new items only use TGenshi.

# Mapping Files on the Filesystem

Inside these plugins specific files live in similarly named directories.

## Locations in the Directory Tree

```
/etc/openafs/ThisCell ⇒ Cfg/etc/openafs/ThisCell/*
```

```
/etc/pam.d/sshhd ⇒ TGenshi/etc/pam.d/sshhd/*
```

# Permissions on Managed Files

Take a look at `Cfg/etc/cups/cupsd.conf/info.xml`.

- Info<sup>6</sup> Files Handle Owners, Permissions, Etc.
- Use the XML Variant as it Supports Grouping

## Contrived Example

```
<FileInfo>
  <Group name="webserver">
    <Info owner="root" group="root" perms="0652"/>
  </Group>
  <Info owner="root" group="sys" perms="0651"/>
</FileInfo>
```

- owner – Textual String
- group – Textual String
- perms – Octal (Leading '0')
- encoding – Set to base64 for Binary Files

<sup>6</sup><http://docs.bcfg2.org/server/info.html>

# Groups and Hosts

You can have different templates for specific groups or hosts.<sup>7</sup>

```
template.G##_groupname
```

`template` Template Filename

`##` A 2 Digit Priority Number (00 - 99)

`groupname` A Group Name

```
template.H_fqdn
```

`template` Template Filename

`fqdn` The Full Qualified Domain Name of the Host

---

<sup>7</sup><http://docs.bcfg2.org/server/plugins/generators/cfg.html>

# Groups and Hosts Example

## /etc/nsswitch.conf

```
$ ls -l Cfg/etc/nsswitch.conf/  
total 8  
-rw-rw-r-- 1 slack slack 296 Apr 27 14:46 nsswitch.conf  
-rw-rw-r-- 1 slack slack 290 Apr 27 14:46 nsswitch.conf.G50_feature-sssd
```

## Contrived /etc/sysconfig/iptables

```
$ ls -l TGenshi/etc/sysconfig/iptables/  
template.newtxt  
template.newtxt.G50_function-squid-server  
template.newtxt.G50_nagios-merlin-server  
template.newtxt.G50_ntp-server  
template.newtxt.G50_web-dl-server  
template.newtxt.G50_web-sec-server  
template.newtxt.G50_web-sys-server  
template.newtxt.H_cthulhu.unity.ncsu.edu
```

# Differences in Cfg and TGenshi

## Cfg<sup>8</sup>:

- Flat File Based
- File Name Equal Directory Name:  
Cfg/etc/openafs/ThisCell/ThisCell

## TGenshi<sup>9</sup>:

- Template Based
- Simplest Template Is a Flat Text File (Mostly)
- Use '\$\$' For One Literal Dollar Sign
- Full Power of the Python Language
- Templates Named `template.newtxt`
- Example: TGenshi/etc/pam.d/sshd/template.newtxt

---

<sup>8</sup><http://docs.bcfg2.org/server/plugins/generators/cfg.html>

<sup>9</sup><http://docs.bcfg2.org/server/plugins/generators/tgenshi>

# TGenshi Examples: Cfg Vs. TGenshi

## Cfg Snippet: /etc/yum.repos.d/realmlinux.repo

```
$ cat realmlinux.repo.G50_realmlinux-el6
...
[realmlinux-base]
name=Realm Linux $releasever - $basearch
baseurl=http://install.linux.ncsu.edu/pub/yum/CLS/RealmLinux/6/base/$basearch
enabled=1
gpgkey=http://www.linux.ncsu.edu/realmlinux/realmlinux.gpg
gpgcheck=1
```

## TGenshi Form

```
$ cat template.newtxt.G50_realmlinux-el6
...
[realmlinux-base]
name=Realm Linux $$releasever - $$basearch
baseurl=http://install.linux.ncsu.edu/pub/yum/CLS/RealmLinux/6/base/$$basearch
enabled=1
gpgkey=http://www.linux.ncsu.edu/realmlinux/realmlinux.gpg
gpgcheck=1
```

# TGenshi Examples: Probes

```
TGenshi/etc/hosers.local/template.newtxt
```

```
{# We just want to ensure this file exists #}\n${metadata.Probes["hosers"].strip()}
```

Read up on the Genshi Text Template Syntax<sup>10</sup>

---

<sup>10</sup><http://genshi.edgewall.org/wiki/Documentation/text-templates.html>

# TGenshi Examples: Referencing Groups

```
TGenshi/etc/mail/sendmail.mc/template.newtxt
```

```
...
define('LOCAL_RELAY', 'ncsu.edu.')
```

`dnl`

```
LOCAL_USER_FILE('/etc/mail/local-mail-users')
```

`dnl`

```
LOCAL_USER('root postmaster mailer-daemon')
```

`dnl`

```
{% if "feature-smartmail" not in metadata.groups %}\
MASQUERADE_AS('ncsu.edu')
```

`dnl`

```
FEATURE(masquerade_envelope)
```

`dnl`

```
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')
```

`dnl`

```
define('SMART_HOST', 'smtp.ncsu.edu')
```

`dnl`

```
{% end %}\
{% if "feature-smartmail" in metadata.groups %}\
DAEMON_OPTIONS('Port=smtp, Name=MTA')
```

`dnl`

```
EXPOSED_USER('root')
```

`dnl`

```
{% end %}\
MAILER(smtp)
```

`dnl`

```
MAILER(procmail)
```

`dnl`

```
dnl MAILER(cyrusv2)
```

`dnl`

# TGenshi Examples: Using Python

```
TGenshi/etc/sysconfig/network/template.newtxt
```

```
{% python

    # Has RLMTTools modified the hostname due to a collision?

    hostname = metadata.hostname
    if ":id:" in hostname:
        hostname = hostname[0:hostname.find(":id:")]
    if ":ID:" in hostname:
        hostname = hostname[0:hostname.find(":ID:")]

%}\
NETWORKWAIT=yes
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=${hostname}
DHCP_HOSTNAME=$$HOSTNAME
```

# TGenshi Examples: Referencing RLMTools

```
TGenshi/etc/rc.conf.d/HostDept/template.newtxt
```

```
{% python
  if 'dept' in metadata.Probes:
      probedDept = metadata.Probes['dept'].strip()
  else:
      probedDept = 'ncsu'

  if hasattr(metadata, 'RLAttributes'):
      if 'dept' in metadata.RLAttributes and \
          metadata.RLAttributes['dept'] is not None:
          dept = metadata.RLAttributes['dept']
      else:
          dept = probedDept
  else:
      dept = probedDept

  # Normalization
  dept = dept.split('\n')[0].strip()
  dept = dept.replace(' ', '-').lower()
%}\
${dept}
```

# Section Outline

- 1 Working with Bcfg2 Repositories
  - Where Things Live
  - The Bcfg2 Data Model
  - Bcfg2 Client Operation
  - Working With Metadata
  - Building Abstract Configuration
  - Creating the Literal Specification
  - Review

# The 6 Step Program

- ① Name Your Group
- ② Does Your Group Need to Include More Bundles?
  - Edit Metadata/\*.xml
  - `git add filenames`
- ③ Edit Bundle Files as Needed
  - Reference the Paths, Packages, Services
  - Group Where Needed
  - `git add filenames`
- ④ Edit Rules/\*.xml if Needed
  - Tip: Use a New File for Each Group/Function
  - `git add filenames`
- ⑤ Edit Templates in TGenshi/
  - `git add filenames`
- ⑥ `git commit`
  - Write a Useful Commit Message

# Getting Help

- Support Tickets: `linux@help.ncsu.edu`
- Jabber: `cls@conference.jabber.eos.ncsu.edu`
- Mailing Lists
  - `realmlinux-users@lists.ncsu.edu`
  - `realmlinux-dev@lists.ncsu.edu`
- CLS Wiki: <https://secure.linux.ncsu.edu/moin/>
- See the Following References

# Bcfg2 References

-  [Bcfg2 Website](http://bcfg2.org/)  
`http://bcfg2.org/`
-  [Bcfg2 Documentation](http://docs.bcfg2.org/)  
`http://docs.bcfg2.org/`
-  [Genshi Website](http://genshi.edgewall.org/)  
`http://genshi.edgewall.org/`
-  [Python Documentation](http://docs.python.org/release/2.6.6/)  
`http://docs.python.org/release/2.6.6/`

## Feedback and Questions