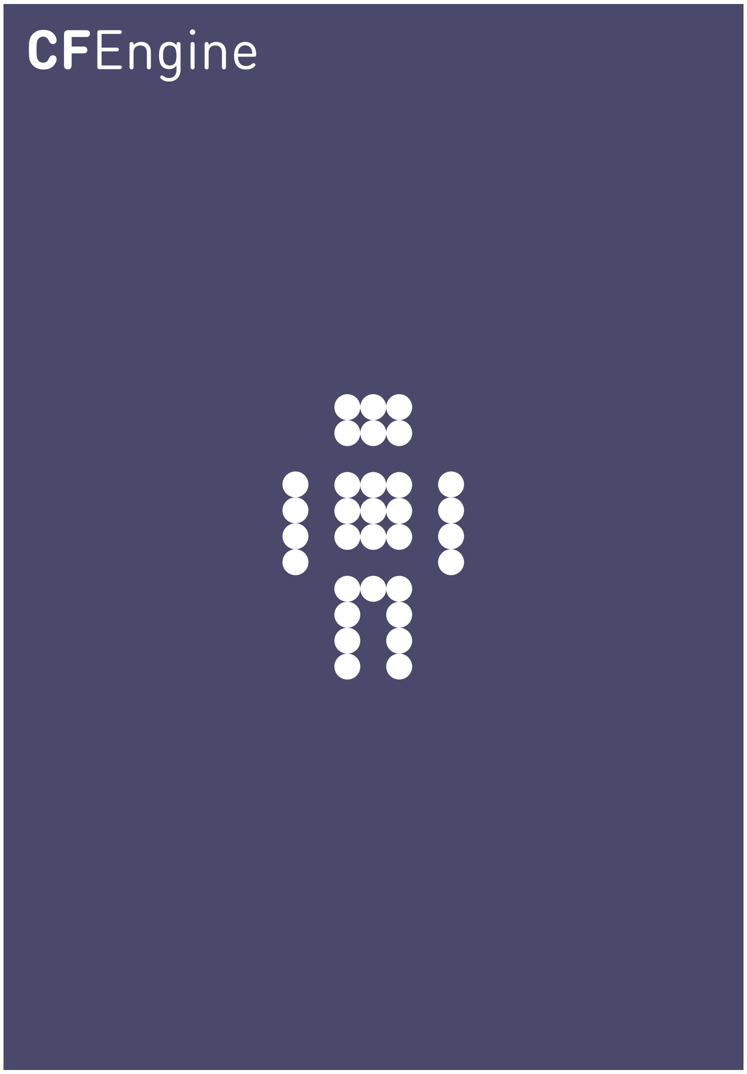

CFEngine 3 Solution Guide

A CFEngine Handbook

CFEngine AS

Table of Contents

1 Introduction

CFEngine 3 has been redesigned to allow modular solution building in terms of a simple, regular language. This guide explains how to use the CFEngine Community Open Promise-Body Library to express some simple idioms and approaches for solving common system problems.

The solutions presented here are necessarily generic and incomplete, since local environmental issues always define the boundaries of a solution in a real setting. A solution guide does not present finished solutions, but rather hints about how to achieve such solutions with standardized idioms.

CFEngine gives you great freedom to craft specific solutions, without the need for low level coding. For most tasks, you will find the standardized templates sufficient for your needs, but the possibilties do not stop there.

The following resources are available as a supplement to this guide:

- Community Open Promise-Body Library, the nuts'n'bolts definitions for the solutions presented here: `http://cfengine.com/manuals/CfengineStdLibrary.html`
- Tutorial: `http://cfengine.com/manuals/cf-manuals/cf3-tutorial.html`
- Reference manual: `http://cfengine.com/manuals/cf-manuals/cf3-Reference.html`

If you need assistance in formulating specific solutions for your system environment, contact CFEngine's Professional Services at `contact@CFEngine.com`;

1.1 Begin - Get Started

To get started with CFEngine, you can imagine the following template for entering examples. This part of the code is common to all the examples.

```
body common control
{
bundlesequence => { "main" };
inputs => { "cfengine_stdlib.cf" };
}


bundle agent main
{
# example
}
```

Then you enter the cases as below. The general pattern of the syntax is like this (colors in html version: red, CFEngine word; blue, user-defined word):

```
# The general pattern

bundle component name(parameters)
{
what_type:
 where_when::

  # Traditional comment
  "promiser" -> { "promisee1", "promisee2" },
        comment => "The intention ...",
         handle => "unique_id_label",
    attribute_1 => body_or_value1,
    attribute_2 => body_or_value2;
}
```

1.2 Create files and directories

Create files and directories and set permissions.

```
########################################################
#
# Simple test create files
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/test_plain"

       perms => system,
       create => "true";

  "/home/mark/tmp/test_dir/."

       perms => system,
       create => "true";

}

#########################################################

body perms system
```

```
{
mode  => "0640";
}

#########################################################
```

1.3 Copy single files

Copy single files, locally (`local_cp`) or from a remote site (`secure_cp`). The Community Open Promise-Body Library (COPBL; ‘`cfengine_stdlib.cf`’) should be included in the ‘`/var/cfengine/inputs/`’ directory and input as below.

```
body common control
{
bundlesequence  => { "mycopy" };
inputs => { "cfengine_stdlib.cf" };
}

bundle agent mycopy
{
files:

  "/home/mark/tmp/test_plain"

    copy_from => local_cp("$(sys.workdir)/bin/file");

  "/home/mark/tmp/test_remote_plain"

    copy_from => secure_cp("$(sys.workdir)/bin/file","serverhost");
}
```

1.4 Copy directory trees

Copy directory trees, locally (`local_cp`) or from a remote site (`secure_cp`). (`depth_search => recurse("")`) defines the number of sublevels to include, (`"inf"`) gets entire tree.

```
body common control
{
bundlesequence  => { "my_recursive_copy" };
inputs => { "cfengine_stdlib.cf" };
}

bundle agent my_recursive_copy
{
```

```
files:

  "/home/mark/tmp/test_dir"

      copy_from => local_cp("$(sys.workdir)/bin/."),
   depth_search => recurse("inf");

  "/home/mark/tmp/test_dir"

      copy_from => secure_cp("$(sys.workdir)/bin","serverhost"),
   depth_search => recurse("inf");

}
```

1.5 Editing password or group files

To change the password of a system, we need to edit a file. A file is a complex object – once open there is a new world of possible promises to make about its contents. CFEngine has bundles of promises that are specially for editing.

```
body common control
{
bundlesequence => { "edit_passwd" };
}

bundle agent edit_passwd
{

vars:

 "userset" slist => { "user1", "user2", "user3" };

files:

  "/etc/passwd"
     edit_line =>
        set_user_field("mark","7","/set/this/shell");


  "/etc/group"
     edit_line =>
        append_user_field("root","4","@(main.userset)");

}
```

1.6 Editing password or group files custom

In this example the bundles from the Community Open Promise-Body Library are included directly in the policy instead of being input as a separate file.

```
body common control
{
bundlesequence => { "addpasswd" };
}

bundle agent addpasswd
{
vars:

  # want to set these values by the names of their array keys

  "pwd[mark]" string => "mark:x:1000:100:Mark Burgess:/home/mark:/bin/bash";
  "pwd[fred]" string => "fred:x:1001:100:Right Said:/home/fred:/bin/bash";
  "pwd[jane]" string => "jane:x:1002:100:Jane Doe:/home/jane:/bin/bash";

files:


  "/tmp/passwd"

        create => "true",
     edit_line => append_users_starting("addpasswd.pwd");

}

############################################################
# Library stuff
############################################################

bundle edit_line append_users_starting(v)

{
vars:

  "index"        slist => getindices("$(v)");

classes:

  "add_$(index)" not => userexists("$(index)");
insert_lines:

  "$($(v)[$(index)])",

      ifvarclass => "add_$(index)";

}

############################################################

bundle edit_line append_groups_starting(v)

{
vars:

  "index"        slist => getindices("$(v)");

classes:

  "add_$(index)" not => groupexists("$(index)");

insert_lines:

  "$($(v)[$(index)])",

      ifvarclass => "add_$(index)";

}
```

1.7 Disabling and rotating files

Use the following simple steps to disable and rotate files. See the Community Open Promise-Body Library if you wish more details on what `disable` and `rotate` does.

```
body common control
{
bundlesequence  => { "my_disable" };
inputs => { "cfengine_stdlib.cf" };
}

bundle agent my_disable
{

files:

  "/home/mark/tmp/test_create"
```

```
      rename => disable;

 "/home/mark/tmp/rotate_my_log"
      rename => rotate("4");

}
```

1.8 Hashing for change detection (tripwire)

Change detection is a powerful and easy way to monitor your environment, increase awareness and harden your system against security breaches.

```
########################################################
#
# Change detect
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/web" -> "me"

   changes      => detect_all_change,
   depth_search => recurse("inf");
}

#########################################################

body changes detect_all_change

{
report_changes => "all";
update_hashes  => "true";
}
```

```
##########################################################

body depth_search recurse(d)

{
depth        => "$(d)";
}
```

1.9 Command or script execution

Execute a command, for instance to start a MySQL service. Note that simple shell commands like `rm` or `mkdir` cannot be managed by CFEngine, so none of the protections that CFEngine offers can be applied to the process. Moreover, this starts a new process, adding to the burden on the system. See CFEngine 3 Best Practices http://cfengine.com/manuals/cf3-bestpractice.html for more information on how to best write policies.

```
body common control
{
bundlesequence  => { "my_commands" };
inputs => { "cfengine_stdlib.cf" };
}


bundle agent my_commands
{
commands:

 Sunday.Hr04.Min05_10.myhost::

  "/usr/bin/update_db";

 any::

  "/etc/mysql/start"

      contain => setuid("mysql");

}
```

1.10 Kill process

```
body common control
{
```

```
bundlesequence => { "test" };
}



bundle agent test
{
processes:

 "sleep"

   signals => { "term", "kill" };

}
```

1.11 Restart process

A basic pattern for restarting processes:

```
body common control
{
bundlesequence => { "process_restart" };
}

#########################################################

bundle agent process_restart
{
processes:

  "/usr/bin/daemon"
        restart_class => "launch";

commands:

  launch::

   "/usr/bin/daemon";

}
```

This can be made more sophisticated to handle generic lists:

```
body common control
{
bundlesequence => { "process_restart" };
}
```

```
#########################################################

bundle agent process_restart
{
vars:

  "component" slist => {
                       "cf-monitord",
                       "cf-serverd",
                       "cf-execd"
                       };
processes:

  "$(component)"
        restart_class => canonify("start_$(component)");

commands:

   "/var/cfengine/bin/$(component)"
       ifvarclass => canonify("start_$(component)");

}
```

Why? Separating this into two parts gives a high level of control and conistency to CFEngine. There are many options for command execution, like the ability to run commands in a sandbox or as 'setuid'. These should not be reproduced in `processes`.

1.12 Check filesystem space

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

bundle agent example

{
vars:

  "free" int => diskfree("/tmp");

reports:
  cfengine_3::

    "Freedisk $(free)";

}
```

1.13 Mount a filesystem

```
#
# cfengine 3
#
# cf-agent -f ./cftest.cf -K
#

body common control

{
bundlesequence => { "mounts" };
}

#

bundle agent mounts

{
storage:

  "/mnt" mount  => nfs("slogans.iu.hio.no","/home");

}

######################################################################

body mount nfs(server,source)

{
mount_type => "nfs";
mount_source => "$(source)";
mount_server => "$(server)";
#mount_options => { "rw" };
edit_fstab => "true";
unmount => "true";
}
```

1.14 Software and patch installation

Example for Debian:

```
# to see list of packages type "apt-cache pkgnames"
# to see list of installed packages type "dpkg --get-selections"
#
# Package managment
#

body common control
{
bundlesequence => { "packages" };
}

body agent control
{
environment => { "DEBIAN_FRONTEND=noninteractive" };
}

#############################################

bundle agent packages
{
vars:

 # Test the simplest case -- leave everything to the yum smart manager

 "match_package" slist => {
                          "apache2"
#                          "apache2-mod_php5",
#                          "apache2-prefork",
#                          "php5"
                          };
packages:

  "$(match_package)"

     package_policy => "add",
     package_method => apt;

}

#############################################

body package_method apt
```

```
{
any::

# ii  acpi      0.09-3ubuntu1

 package_changes => "bulk";
 package_list_command => "/usr/bin/dpkg -l";

 package_list_name_regex    => "ii\s+([^\s]+).*";
 package_list_version_regex => "ii\s+[^\s]+\s+([^\s]+).*";

# package_list_arch_regex    => "none";

 package_installed_regex => ".*"; # all reported are installed

 #package_name_convention => "$(name)_$(version)_$(arch)";
 package_name_convention => "$(name)";

 # Use these only if not using a separate version/arch string
 # package_version_regex => "";
 # package_name_regex => "";
 # package_arch_regex => "";

package_add_command => "/usr/bin/apt-get --yes install";
package_delete_command => "/usr/bin/apt-get --yes remove";
package_update_command =>  "/usr/bin/apt-get --yes dist-upgrade";
#package_verify_command => "/bin/rpm -V";
}
```

Examples MSI for Windows, by name:

```
#
# MSI package managment using file name
#

body common control
{
bundlesequence => { "packages" };
}

#############################################

bundle agent packages
{
vars:
 "match_package" slist => {
                          "7zip-4.65-x86_64.msi"
                          };
packages:

  "$(match_package)"

     package_policy => "add",
     package_method => msi_fmatch;

}

#############################################

body package_method msi_fmatch

{
 package_changes => "individual";
 package_file_repositories => { "$(sys.workdir)\software_updates\windows", "s:\su" };

 package_installed_regex => ".*";

 package_name_regex    => "^(\S+)-(\d+\.?)+";
 package_version_regex => "^\S+-((\d+\.?)+)";
 package_arch_regex    => "^\S+-(\d+\.?)+(^.+)";

 package_name_convention => "$(name)-$(version)-$(arch).msi";

 package_add_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /i";
 package_update_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /i";
 package_delete_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /x";
 }
```

Windows MSI by version:

```
#
# MSI package managment using version criteria
#

body common control
{
bundlesequence => { "packages" };
}

#############################################

bundle agent packages
```

```
{
vars:

 "match_package" slist => {
                          "7zip"
                          };
packages:

  "$(match_package)"

     package_policy => "update",
     package_select => ">=",
     package_architectures => { "x86_64" },
     package_version => "3.00",
     package_method => msi_vmatch;

}

#############################################

body package_method msi_vmatch

{
 package_changes => "individual";
 package_file_repositories => { "$(sys.workdir)\software_updates\windows", "s:\su" };

 package_installed_regex => ".*";

 package_name_convention => "$(name)-$(version)-$(arch).msi";

 package_add_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /i";
 package_update_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /i";
 package_delete_command => "\"$(sys.winsysdir)\msiexec.exe\" /qn /x";
}
```

Examples for solaris are more complex:

```
#
# Package managment
#

body common control
{
bundlesequence => { "packages" };
inputs => { "cfengine_stdlb.cf" };
```

```
}

#############################################

bundle agent packages
{
vars:

  "solaris_packages[SMCzlib]" string => "zlib-1.2.3-sol10-sparc-local";
  "admin_file"                string => "cfengine_admin_file";

  "package_names"              slist => getindices("solaris_packages");

files:

  "/tmp/$(admin_file)"
         create => "true",
         edit_defaults => empty_file,
         edit_line => create_solaris_admin_file;

packages:

  "$(package_names)"

     package_policy => "add",
     package_method => solaris("$(package_names)", "$(solaris_packages[$(package_names)])", "$(admin

}
```

Examples for yum based systems:

```
#
# Package managment
#

body common control
{
bundlesequence => { "packages" };
inputs => { "cfengine_stdlib.cf" };
}

#############################################

bundle agent packages
{
```

```
vars:

 # Test the simplest case -- leave everything to the yum smart manager

 "match_package" slist => {
                          "apache2",
                          "apache2-mod_php5",
                          "apache2-prefork",
                          "php5"
                          };
packages:

  "$(match_package)"

     package_policy => "add",
     package_method => yum;

}
```

SuSE Linux's package manager zypper is the most powerful alternative:

```
#
# Package managment
#

body common control
{
bundlesequence => { "packages" };
inputs => { "cfengine_stdlib.cf" };
}

#############################################

bundle agent packages
{
vars:

 # Test the simplest case -- leave everything to the zypper smart manager

 "match_package" slist => {
                          "apache2",
                          "apache2-mod_php5",
                          "apache2-prefork",
                          "php5"
                          };
packages:

  "$(match_package)"

     package_policy => "add",
     package_method => zypper;

}
```

2 High level

2.1 Centralized Management

These examples show a simple setup for starting with a central approach to management of servers. Centralization of management is a simple approach suitable for small environments with few requirements. It is useful for clusters where systems are all alike.

2.2 All hosts the same

This shows the simplest approach in which all hosts are the same. It is too simple for most environments, but it serves as a starting point. Compare it to the next section that includes variation.

```
body common control
   {
   bundlesequence  => { "central" };
   }


############################################

bundle agent central

{
vars:

  "policy_server" string => "myhost.domain.tld";

  "mypackages" slist => {
                        "nagios"
                        "gcc",
                        "apache2",
                        "php5"
                        };

files:

  # Password management can be very simple if all hosts are identical

  "/etc/passwd"

    comment   => "Distribute a password file",
    perms     => mog("644","root","root"),
    copy_from => secure_cp("/home/mark/LapTop/words/RoadAhead","$(policy_server)");

packages:

  "$(mypackages)"

     package_policy => "add",
     package_method => generic;


# Add more promises below ...

}


#########################################################
# Server config
#########################################################

body server control

{
allowconnects         => { "127.0.0.1" , "::1", "10.20.30" };
allowallconnects      => { "127.0.0.1" , "::1", "10.20.30" };
trustkeysfrom         => { "127.0.0.1" , "::1", "10.20.30" };
# allowusers
}

#########################################################

bundle server access_rules()

{
access:

 # myhost.domain.tld makes this file available to 10.20.30*

 myhost_domain_tld::

  "/etc/passwd"

     admit   => { "127.0.0.1", "10.20.30" };
}
```

2.3 Variation in hosts

```
body common control
   {
   bundlesequence  => { "central" };
   }
```

```
############################################

bundle agent central

{
classes:

  "mygroup_1" or => { "myhost", "host1", "host2", "host3" };
  "mygroup_2" or => { "host4", "host5", "host6" };

vars:

  "policy_server" string => "myhost.domain.tld";

 mygroup_1::

  "mypackages" slist => {
                        "nagios"
                        "gcc",
                        "apache2",
                        "php5"
                        };

 mygroup_2::

  "mypackages" slist => {
                        "apache"
                        "mysql",
                        "php5"
                        };


files:

  # Password management can be very simple if all hosts are identical

  "/etc/passwd"

    comment   => "Distribute a password file",
    perms     => mog("644","root","root"),
    copy_from => secure_cp("/etc/passwd","$(policy_server)");

packages:

  "$(mypackages)"

     package_policy => "add",
     package_method => generic;


# Add more promises below ...

}


#########################################################
# Server config
#########################################################

body server control

{
allowconnects         => { "127.0.0.1" , "::1", "10.20.30" };
allowallconnects      => { "127.0.0.1" , "::1", "10.20.30" };
trustkeysfrom         => { "127.0.0.1" , "::1", "10.20.30" };
# allowusers
}

#########################################################

bundle server access_rules()

{
access:

 # myhost.domain.tld makes this file available to 10.20.30*

 myhost_domain_tld::

  "/etc/passwd"

     admit   => { "127.0.0.1", "10.20.30" };
}
```

2.4 Updating from a central hub

The configuration bundled with the CFEngine source code contains an example of centralized updating of policy that covers more subtleties than this example, and handles fault tolerance. Here is the main idea behind it. For simplicity, we assume that all hosts are on network 10.20.30.* and that the central policy server/hub is 10.20.30.123.

```
bundle agent update
{
```

```
vars:

 "master_location" string => "/var/cfengine/masterfiles";

 "policy_server"   string => "10.20.30.123";
                   comment => "IP address to locate your policy host.";

files:

  "$(sys.workdir)/inputs"

    perms => system("600"),
    copy_from => remote_cp("$(master_location)",$(policy_server)),
    depth_search => recurse("inf");
}

#######################################################

body server control

{
allowconnects         => { "127.0.0.1" , "10.20.30" };
allowallconnects      => { "127.0.0.1" , "10.20.30" };
trustkeysfrom         => { "127.0.0.1" , "10.20.30" };
}

#######################################################

bundle server access_rules()
{
access:

 10_20_30_123::

  "/var/cfengine/masterfiles"

    admit   => { "127.0.0.1", "10.20.30" };
}
```

2.5 Change detection

```
body common control

{
bundlesequence  => { "testbundle"  };

inputs => { "cfengine_stdlib.cf" };
}

########################################################

bundle agent testbundle

{
files:

  "/usr"

       changes      => detect_all_change,
       depth_search => recurse("inf"),
       action       => background;
}
```

2.6 Garbage collection

```
body common control
{
bundlesequence => { "garbage_collection" };
inputs => { "cfengine_stdlib.cf" };
}


bundle agent garbage_collection
{
files:

 Sunday::

  "$(sys.workdir)/nova_repair.log"

    comment => "Rotate the promises repaired logs each week",
    rename => rotate("7"),
    action => if_elapsed("10000");

  "$(sys.workdir)/nova_notkept.log"

    comment => "Rotate the promises not kept logs each week",
    rename => rotate("7"),
    action => if_elapsed("10000");

  "$(sys.workdir)/promise.log"

    comment => "Rotate the promises not kept logs each week",
    rename => rotate("7"),
    action => if_elapsed("10000");

 any::

  "$(sys.workdir)/outputs"

    comment => "Garbage collection of any output files",
    delete => tidy,
    file_select => days_old("3"),
    depth_search => recurse("inf");

  "$(sys.workdir)/"

    comment => "Garbage collection of any output files",
    delete => tidy,
    file_select => days_old("14"),
    depth_search => recurse("inf");

  # Other resources

  "/tmp"

    comment => "Garbage collection of any temporary files",
    delete => tidy,
    file_select => days_old("3"),
    depth_search => recurse("inf");

  "/var/log/apache2/.*bz"

    comment => "Garbage collection of rotated log files",
    delete => tidy,
    file_select => days_old("30"),
    depth_search => recurse("inf");

  "/var/log/apache2/.*gz"

    comment => "Garbage collection of rotated log files",
    delete => tidy,
    file_select => days_old("30"),
    depth_search => recurse("inf");

  "/var/log/zypper.log"

    comment => "Prevent the zypper log from choking the disk",
    rename => rotate("0"),
    action => if_elapsed("10000");

}
```

2.7 Distribute root passwords

```
######################################################################
#
# Root password distribution
#
######################################################################

body common control

{
version => "1.2.3";
bundlesequence  => { "SetRootPassword" };
}

########################################################

bundle common g
{
vars:

  "secret_keys_dir" string => "/tmp";
}

########################################################

bundle agent SetRootPassword

{
files:

  "/var/cfengine/ppkeys/rootpw.txt"

      copy_from => scp("$(sys.fqhost)-root.txt","master_host.example.org");

      # or $(pw_class)-root.txt

  # Or get variables directly from server woth Nova

 "remote-passwd" string => remotescalar("rem_password","127.0.0.1","yes");

  # Test this on a copy

  "/tmp/shadow"

       edit_line => SetRootPw;
}

########################################################

bundle edit_line SetRootPw
  {
  vars:

   # Assume this file contains a single string of the form root:passwdhash:
   # with : delimiters to avoid end of line/file problems

   "pw" int => readstringarray("rpw","$(sys.workdir)/ppkeys/rootpw.txt",
                                                    "#[^\n]*",":","1","200");

  field_edits:

   "root:.*"

      # Set field of the file to parameter

      edit_field => col(":","2","$(rpw[1])","set");
  }

########################################################

bundle server passwords
{
vars:

  # Read a file of format
  #
  # classname: host1,host2,host4,IP-address,regex.*,etc
  #

      "pw_classes" int => readstringarray("acl","$(g.secret_keys_dir)/classes.txt",
                                                       "#[^\n]*",":","100","4000");
  "each_pw_class" slist => getindices("acl");

access:

  "/secret/keys/$(each_pw_class)-root.txt"

        admit   => splitstring("$(acl[$(each_pw_class)][1])" , ":" , "100"),
    ifencrypted => "true";

}
```

2.8 Distribute ssh keys

```
# Assume that we have collected all users' public keys into a single source area
# on the server. First copy the ones we need to localhost, and then edit them into
# the the user's local keyring.

  # vars:
  #
  #  "users" slist => { "user1", "user2", ...};
  #
  # methods:
  #
  #  "any" usebundle => allow_ssh_login_from_authorized_keys(@(users),"sourcehost");
  #

#######################################################################

bundle agent allow_ssh_rootlogin_from_authorized_keys(user,sourcehost)
{
vars:

  "local_cache"       string => "/var/cfengine/ssh_cache";
  "authorized_source" string => "/master/CFEngine/ssh_keys";

files:

   "$(local_cache)/$(user).pub"

         comment => "Copy public keys from a an authorized cache into a cache on localhost",
           perms => mo("600","root"),
       copy_from => remote_cp("$(authorized_source)/$(user).pub","$(sourcehost)"),
          action => if_elapsed("60");

   "/root/.ssh/authorized_keys"

         comment => "Edit the authorized keys into the user's personal keyring",
       edit_line => insert_file_if_no_line_matching("$(user)","$(local_cache)/$(user).pub"),
          action => if_elapsed("60");
}

#######################################################################

bundle agent allow_ssh_login_from_authorized_keys(user,sourcehost)
{
vars:

  "local_cache"       string => "/var/cfengine/ssh_cache";
  "authorized_source" string => "/master/CFEngine/ssh_keys";

files:

   "$(local_cache)/$(user).pub"

         comment => "Copy public keys from a an authorized cache into a cache on localhost",
           perms => mo("600","root"),
       copy_from => remote_cp("$(authorized_source)/$(user).pub","$(sourcehost)"),
          action => if_elapsed("60");

   "/home/$(user)/.ssh/authorized_keys"

         comment => "Edit the authorized keys into the user's personal keyring",
       edit_line => insert_file_if_no_line_matching("$(user)","$(local_cache)/$(user).pub"),
          action => if_elapsed("60");
}

########################################################################

bundle edit_line insert_file_if_no_line_matching(user,file)
{
classes:

  "have_user" expression => regline("$(user).*","$(this.promiser)");

insert_lines:

  !have_user::

    "$(file)"
         insert_type => "file";
}
```

2.9 Laptop support configuration

Laptops do not need a lot of confguration support. IP addresses are set by DHCP and conditions are changeable. But you want to set your DNS search domains to familiar settings in spite of local DHCP configuration, and another useful trick is to keep a regular backup of disk changes on the local disk. This won't help against disk destruction, but it is a huge advantage when your user accidentally deletes files while travelling or offline.

```
#######################################################
#
# Laptop
#
#######################################################

body common control

{
bundlesequence  => {
                   "update",
                   "garbage_collection",
                   "main",
                   "backup",
                   };

inputs          => {
                   "update.cf",
                   "site.cf",
                   "library.cf"
                   };
}

#######################################################

body agent control
{
# if default runtime is 5 mins we need this for long jobs
ifelapsed => "15";
}

#######################################################

body monitor control
{
forgetrate => "0.7";
}

#######################################################
```

```
body executor control

{
splaytime => "1";
mailto => "mark@iu.hio.no";
smtpserver => "localhost";
mailmaxlines => "30";

# Instead of a separate update script, now do this

exec_command => "$(sys.workdir)/bin/cf-agent -f failsafe.cf && $(sys.workdir)/bin/cf-agent";
}

#######################################################
# General site issues can be in bundles like this one
#######################################################

bundle agent main

{
vars:

  "component" slist => { "cf-monitord", "cf-serverd" };

 # - - - - - - - - - - - - - - - - - - - - - - - -

files:

  "$(sys.resolv)"  # test on "/tmp/resolv.conf" #

     create        => "true",
     edit_line     => resolver,
     edit_defaults => def;

processes:

  "$(component)" restart_class => canonify("start_$(component)");

 # - - - - - - - - - - - - - - - - - - - - - - - -

commands:

   "$(sys.workdir)/bin/$(component)"

       ifvarclass => canonify("start_$(component)");
}
```

```
#######################################################
# Backup
#######################################################

bundle agent backup
{
files:

  "/home/backup"

     copy_from => cp("/home/mark"),
  depth_search => recurse("inf"),
   file_select => exclude_files,
        action => longjob;

}

#######################################################
# Garbage collection issues
#######################################################

bundle agent garbage_collection
{
files:

  "$(sys.workdir)/outputs"

    delete => tidy,
    file_select => days_old("3"),
    depth_search => recurse("inf");


}
```

2.10 Find the MAC address

Finding the ethernet address can be hard, but on Linux it is straightforward.

```
bundle agent test
{
vars:

linux::
 "interface" string => execresult("/sbin/ifconfig eth0","noshell");

solaris::
 "interface" string => execresult("/usr/sbin/ifconfig bge0","noshell");

freebsd::
 "interface" string => execresult("/sbin/ifconfig le0","noshell");

darwin::
 "interface" string => execresult("/sbin/ifconfig en0","noshell");

classes:

 linux::

   "ok" expression => regextract(
                                ".*HWaddr ([^\s]+).*(\n.*)*",
                                "$(interface)",
                                "mac"
                                );

 solaris::

   "ok" expression => regextract(
                                ".*ether ([^\s]+).*(\n.*)*",
                                "$(interface)",
                                "mac"
                                );

 freebsd::

   "ok" expression => regextract(
                                ".*ether ([^\s]+).*(\n.*)*",
                                "$(interface)",
                                "mac"
                                );

 darwin::
   "ok" expression => regextract(
                                 "(?s).*ether ([^\s]+).*(\n.*)*",
                                 "$(interface)",
                                 "mac"
                                 );

reports:

ok::

  "MAC address is $(mac[1])";

}
```

2.11 Log rotation

```
body common control
   {
   bundlesequence  => { "testbundle" };
   }


############################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/rotateme"

      rename => rotate("4");
}

############################################

body rename rotate(level)

{
rotate => "$(level)";
}
```

2.12 Manage a system file

2.13 Simple template

```
bundle agent hand_edited_config_file
{
vars:

  "file_template" string =>

"
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost
::1             localhost ipv6-localhost ipv6-loopback
fe00::0         ipv6-localnet
ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
10.0.0.100      host1.domain.tld host1
10.0.0.101      host2.domain.tld host2
10.0.0.20       host3.domain.tld host3
10.0.0.21       host4.domain.tld host4
";

##############################################################

files:

   "/etc/hosts"

       comment => "Define the content of all host files from this master source",
        create => "true",
     edit_line => append_if_no_lines("$(file_template)"),
 edit_defaults => empty,
         perms => mo("$(mode)","root"),
        action => if_elapsed("60");
}
```

2.14 Simple versioned template

The simplest approach to managing a file is to maintain a master copy by hand, keeping it in a version controlled repository (e.g. svn), and installing this version on the end machine.

We'll assume that you have a version control repository that is located on some independent server, and has been checked out manually once (with authentication) in ‘/mysite/masterfiles’.

```
bundle agent hand_edited_config_file
{
vars:

  "masterfiles"   string => "/mysite/masterfiles";
  "policy_server" string => "policy_host.domain.tld";

files:

   "/etc/hosts"

        comment => "Synchronize hosts with a hand-edited template in svn",
          perms => m("644"),
      copy_from => remote_cp("$(masterfiles)/trunk/hosts_master","$(policy_server)");

commands:


   "/usr/bin/svn update"

        comment => "Update the company document repository including manuals to a local copy",
        contain => silent_in_dir("$(masterfiles)/trunk"),
     ifvarclass => canonify("$(policy_server)");

}
```

2.15 Macro template

The next simplest approach to file management is to add variables to the template that will be expanded into local values at the end system, e.g. using variables like ‘$(sys.host)’ for the name of the host within the body of the versioned template.

```
bundle agent hand_edited_template
{
vars:

  "masterfiles"   string => "/mysite/masterfiles";
  "policy_server" string => "policy_host.domain.tld";

files:
   "/etc/hosts"

        comment => "Synchronize hosts with a hand-edited template in svn",
          perms => m("644"),
        create => "true",
     edit_line => expand_template("$(masterfiles)/trunk/hosts_master"),
 edit_defaults => empty,
        action => if_elapsed("60");

commands:


   "/usr/bin/svn update"

        comment => "Update the company document repository including manuals to a local copy",
        contain => silent_in_dir("$(masterfiles)/trunk"),
     ifvarclass => canonify("$(policy_server)");

}
```

The macro template file may contain variables, as below, that get expanded by CFEngine.

```
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost $(sys.host)
::1             localhost ipv6-localhost ipv6-loopback
fe00::0         ipv6-localnet
ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
10.0.0.100      host1.domain.tld host1
10.0.0.101      host2.domain.tld host2
10.0.0.20       host3.domain.tld host3
10.0.0.21       host4.domain.tld host4

# Add below this line

$(definitions.more_hosts)
```

2.16 Custom editing

If you do not control the starting state of the file, because it is distributed by an operating system vendor for instance, then editing the final state is the best approach. That way, you will get changes that are made by the vendor, and will ensure your own modifications are kept even when updates arrive.

```
bundle agent modifying_managed_file
{
vars:
  "data"   slist => { "10.1.2.3 sirius", "10.1.2.4 ursa-minor", "10.1.2.5 orion"};

files:

   "/etc/hosts"

        comment => "Append a list of lines to the end of a file if they don’t exist",█
          perms => m("644"),
        create => "true",
     edit_line => append_if_no_lines("modifying_managed_file.data"),
        action => if_elapsed("60");


}
```

Another example shows how to set the values of variables using a data-driven approach and methods from the standard library.

```
#######################################################
#
# Edit variable = value in a text file
#
#######################################################

body common control

{
bundlesequence  => { "testsetvar" };
}

#######################################################

bundle agent testsetvar

{
vars:

  "v[variable_1]" string => "value_1";
  "v[variable_2]" string => "value_2";

files:

  "/tmp/test_setvar"

     edit_line => set_variable_values("testsetvar.v");
}
```

2.17 Manage a system process

2.18 Ensure running

The simplest example might look like this:

```
bundle agent restart_process
{
processes:

  "httpd"

      comment => "Make sure apache web server is running",
      restart_class => "restart_httpd";

commands:

 restart_httpd::

     "/etc/init.d/apache2 restart";

}
```

This example shows how the CFEngine components could be started using a pattern.

```
bundle agent CFEngine_processes
{
vars:

  "components" slist => { "cf-execd", "cf-monitord", "cf-serverd", "cf-hub" };

processes:

  "$(components)"

      comment => "Make sure server parts of CFEngine are running",
      restart_class => canonify("start_$(component)");

commands:

   "$(sys.workdir)/bin/$(component)"

          comment => "Make sure server parts of CFEngine are running",
       ifvarclass => canonify("start_$(components)");

}
```

2.19 Ensure not running

```
bundle agent restart_process
{
vars:

  "killprocs" slist => { "snmpd", "gameserverd", "irc", "crack" };

processes:

  "$(killprocs)"

      comment => "Make processes are not running",
      signals => { "term", "kill" };
;
}
```

2.20 Prune processes

This example kills processes owned by a particular user that have exceeded 100000 bytes of resident memory.

```
body common control
{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
processes:

 ".*"

    process_select  => big_processes("mark"),
            signals => { "term" };
}

########################################################

body process_select big_processes(o)
```

```
{
process_owner => { "$(o)" };
rsize => irange("100000","900000");
process_result => "rsize.process_owner";
}
```

2.21 Manage users

There are many approaches to managing users. You can edit system files like '/etc/passwd' directly, or you can use commands on some systems like 'useradd' or 'adduser'. In all cases it is desirable to make this a data-driven process.

2.22 Add users

A simple approach which adds new users to the password file, and to a group called 'users' in the group file. Is shown below. This example does not edit the shadow file. A simple pattern that can be modified for use is shown below.

Note that, although this is a simple minded approach, it is the most efficient of the approaches shown here as all operations can be carried out in a single operation for each file.

```
bundle agent addusers
{
vars:

  # Add some users

  "pw[mark]" string => "mark:x:1000:100:Mark Burgess:/home/mark:/bin/bash";
  "pw[fred]" string => "fred:x:1001:100:Right Said:/home/fred:/bin/bash";
  "pw[jane]" string => "jane:x:1002:100:Jane Doe:/home/jane:/bin/bash";

  "users" slist => getindices("pw");

files:

  "/etc/passwd"
     edit_line => append_users_starting("addusers.pw");

#  "/etc/shadow"
#     edit_line => append_users_starting("$(users):defaultpasswd:::::::");

  "/etc/group"
       edit_line => append_user_field("users","4","@(addusers.users)");

  "/home/$(users)/."

     create => "true",
      perms => mog("755","$(users)","users");
}
```

A second approach is to use the shell commands supplied by some operating systems; this assumes that suitable defaults have been set up manually. Also the result is not repairable in a simple convergent manner. The command needs to edit multiple files for each user, and is quite inefficient.

```
bundle agent addusers
{
```

```
vars:

  "users" slist => { "mark", "fred", "jane" };

commands:

   "/usr/sbin/useradd $(users)";
}
```

An alternative approach is to use a method to wrap around the handling of a user. Although this looks nice, it is less efficient than the first method because it must edit the files multiple times.

```
bundle agent addusers
{
vars:

  # Add some users

  "pw[mark]" string => "mark:x:1000:100:Mark Burgess:/home/mark:/bin/bash";
  "pw[fred]" string => "fred:x:1001:100:Right Said:/home/fred:/bin/bash";
  "pw[jane]" string => "jane:x:1002:100:Jane Doe:/home/jane:/bin/bash";

  "users" slist => getindices("pw");

methods:

  "any" usebundle => user_add("$(users)","$(pw[$(users)])");

}

bundle agent user_add(x,pw)
{
files:

  "/etc/passwd"
     edit_line => append_users_starting("addusers.pw");

#  "/etc/shadow"
#     edit_line => append_users_starting("$(users):defaultpasswd:::::::");

  "/etc/group"
      edit_line => append_user_field("users","4","@(addusers.users)");

  "/home/$(users)/."

     create => "true",
      perms => mog("755","$(users)","users");
}
```

2.23 Remove users

2.24 Postfix mail configuration

```
#######################################################
#
# Postfix
#
#######################################################

body common control

{
any::

  bundlesequence  => {
                     postfix
                     };
}

#######################################################

bundle agent postfix

{
vars:

 "prefix"     string => "/etc";
 "smtpserver" string => "localhost";
 "mailrelay"  string => "mailx.example.org";

files:

  "$(prefix)/main.cf"
      edit_line => prefix_postfix;

  "$(prefix)/sasl-passwd"
      create    => "true",
      perms     => mo("0600","root"),
      edit_line => append_if_no_line("$(smtpserver) _$(sys.fqhost):chmsxrcynz4etfrejizhs22");
}

#######################################################
# For the library
#######################################################

bundle edit_line prefix_postfix

{
```

```
#
# Value have the form NAME = "quoted space separated list"
#
vars:

  "ps[relayhost]"                  string => "[$(postfix.mailrelay)]:587";
  "ps[mydomain]"                   string => "iu.hio.no";
  "ps[smtp_sasl_auth_enable]"      string => "yes";
  "ps[smtp_sasl_password_maps]"    string => "hash:/etc/postfix/sasl-passwd";
  "ps[smtp_sasl_security_options]" string => "";
  "ps[smtp_use_tls]"               string => "yes";
  "ps[default_privs]"              string => "mailman";
  "ps[inet_protocols]"             string => "all";
  "ps[inet_interfaces]"            string => "127.0.0.1";

  "parameter_name" slist => getindices("ps");

delete_lines:

  "$(parameter_name).*";

insert_lines:

  "$(parameter_name) = $(ps[$(parameter_name)])";

}

########################################################

bundle edit_line AppendIfNSL(parameter)
  {
  insert_lines:

    "$(parameter)"; # This is default
  }
```

2.25 Set up HPC clusters

HPC cluster machines are usually all identical, so the CFEngine configuration is very simple. HPC clients value CPU and memory resources, so we can shut down unnecessary services to save CPU. We can also change the scheduling rate of CFEngine to run less frequently, and save a little:

```
#######################################################

body executor control
```

```
{
splaytime => "1";
mailto => "cfengine@example.com";
smtpserver => "localhost";
mailmaxlines => "30";

# Once per hour, on the hour

schedule     => { "Min00_05" };
}

#######################################################

bundle agent services_disable
{
vars:

   # list all of xinetd services (case sensitive)

   "xinetd_services" slist => {
                               "imap",
                               "imaps",
                               "ipop2",
                               "ipop3",
                               "krb5-telnet",
                               "klogin",
                               "kshell",
                               "ktalk",
                               "ntalk",
                               "pop3s",

                              };
methods:

   # perform the actual disable all xinetd services according to the list above

   "any"  usebundle => disable_xinetd("$(xinetd_services)");

processes:

  "$(xinetd_services)"

           signals => { "kill" };

}

###############################################################################
```

```
bundle agent disable_xinetd(name)
{
 vars:
   "status" string => execresult("/sbin/chkconfig --list $(name)", "useshell");

 classes:
   "on"  expression => regcmp(".*on.*","$(status)");

 commands:
   on::
      "/sbin/chkconfig $(name) off",
        comment => "disable $(name) service";

 reports:
   on::
      "disable $(name) service.";

}
```

2.26 Set up name resolution

There are many ways to do name resolution setup[1] We write a reusable bundle using the editing features.

A simple and straightforward approach is to maintain a separate modular bundle for this task. This avoids too many levels of abstraction and keeps all the information in one place. We implement this as a simple editing promise for the '/etc/resolv.conf' file.

```
bundle agent system_files

{
files:

  "$(sys.resolv)"  # test on "/tmp/resolv.conf" #

     comment       => "Add lines to the resolver configuration",
     create        => "true",
     edit_line     => resolver,
     edit_defaults => std_edits;

   # ...other system files ...

}

#######################################################

bundle edit_line resolver

{
delete_lines:

  # delete any old name servers or junk we no longer need

  "search.*";
  "nameserver 80.65.58.31";
  "nameserver 80.65.58.32";
  "nameserver 82.103.128.146";
  "nameserver 78.24.145.4";
  "nameserver 78.24.145.5";
  "nameserver 128.39.89.10";

insert_lines:

  "search mydomain.tld" location => start;
```

[1] In CFEngine 2 there is a separate action type for configuring the system resolver. In CFEngine 3 this has been deprecated for this standard method using the basic functionality of CFEngine.

```
 special_net::

  "nameserver 128.39.89.8";
  "nameserver 128.39.74.66";

 !special_net::

  "nameserver 128.38.34.12";

 any::

  "nameserver 212.112.166.18";
  "nameserver 212.112.166.22";
}
```

A second approach is to try to conceal the operational details behind a veil of abstraction.

```
bundle agent system_files
{
vars:

 "searchlist"  string => "iu.hio.no CFEngine.com";

 "nameservers" slist => {
                        "128.39.89.10",
                        "128.39.74.16",
                        "192.168.1.103"
                        };
files:

  "$(sys.resolv)"  # test on "/tmp/resolv.conf" #
      create        => "true",
      edit_line     => doresolv("$(s)","@(this.n)"),
      edit_defaults => empty;

 # ....

}

#######################################################

bundle edit_line doresolv(search,names)

{
insert_lines:
  "search $(search)";
  "nameserver $(names)";
}
```

DNS is not the only name service, of course. Unix has its older '/etc/hosts' file which can also be managed using file editing. We simply append this to the system_files bundle.

```
bundle agent system_files
{

# ...

files:

   "/etc/hosts"

     comment => "Add hosts to the /etc/hosts file",
   edit_line => fix_etc_hosts;
}

###########################################################

bundle edit_line fix_etc_hosts
{
vars:

 "names[127.0.0.1]"    string => "localhost localhost.CFEngine.com";
 "names[128.39.89.12]" string => "myhost myhost.CFEngine.com";
 "names[128.39.89.13]" string => "otherhost otherhost.CFEngine.com";

 # etc

 "i" slist => getindices("names");

insert_lines:

 "$(i)     $(names[$(i)])";

}
```

2.27 Set up sudo

Setting up sudo is straightforward, and is best managed by copying trusted files from a repository.

```
bundle agent system_files
{
vars:

 "masterfiles" string => "/subversion_projects/masterfiles";

# ...

files:

   "/etc/sudoers"

     comment => "Make sure the sudo configuration is secure and up to date",
       perms => mog("440","root","root"),
   copy_from => secure_cp("$(masterfiles)/sudoers","$(policy_server)");

}
```

2.28 Set up a web server

Adapt this template to your operating system by adding multiple classes. Each web server runs something like the present module, which is entered into the bundlesequence like this:

```
#####################################################
#
# Apache webserver module
#
#####################################################

bundle agent web_server(state)
{
vars:

  "document_root" string => "/";

 ####################################################
 # Site specific configuration - put it in this file
 ####################################################

  "site_http_conf" string => "/home/mark/CFEngine-inputs/httpd.conf";

 ####################################################
 # Software base
 ####################################################

  "match_package" slist => {
                           "apache2",
                           "apache2-mod_php5",
                           "apache2-prefork",
                           "php5"
                           };

 #########################################################

processes:

  web_ok.on::

   "apache2"

     restart_class => "start_apache";

  off::

   "apache2"
```

```
     process_stop => "/etc/init.d/apache2 stop";


 #########################################################

commands:

 start_apache::

   "/etc/init.d/apache2 start"; # or startssl

 #########################################################

packages:

  "$(match_package)"

     package_policy => "add",
     package_method => zypper,
     classes => if_ok("software_ok");

 #########################################################

files:

 software_ok::

  "/etc/sysconfig/apache2"

     edit_line => fixapache,
     classes => if_ok("web_ok");

 #########################################################

reports:

 !software_ok.on::

    "The web server software could not be installed";

 #########################################################

classes:

  "on"  expression => strcmp("$(state)","on");
  "off" expression => strcmp("$(state)","off");
}
```

```
########################################################
# For the library
########################################################

bundle edit_line fixapache

{
vars:

 "add_modules"     slist => {
                            "ssl",
                            "php5"
                            };

 "del_modules"     slist => {
                            "php3",
                            "php4",
                            "jk"
                            };

insert_lines:

 "APACHE_CONF_INCLUDE_FILES=\"$(web_server.site_http_conf)\"";

field_edits:

 ######################################################################
 # APACHE_MODULES="actions alias ssl php5 dav_svn authz_default jk" etc..
 ######################################################################

   "APACHE_MODULES=.*"

      # Insert module "columns" between the quoted RHS
      # using space separators

      edit_field => quotedvar("$(add_modules)","append");

   "APACHE_MODULES=.*"

      # Delete module "columns" between the quoted RHS
      # using space separators

      edit_field => quotedvar("$(del_modules)","delete");

   # if this line already exists, edit it
}
```

2.29 Templating

With CFEngine you have a choice between editing 'deltas' into files or distributing more-or-less finished templates. Which method you should choose depends should be made by whatever is easiest.

- If you are managing only part of the file, and something else (e.g. a package manager) is managing most of it, then it makes sense to use CFEngine file editing.
- If you are managing everything in the file, then it makes sense to make the edits by hand and install them using CFEngine. You can use variables within source text files and let CFEngine expand them locally in situ, so that you can make generic templates that apply netwide.

Example template:

```
#
# System file X
#

MYVARIABLE = something or other
HOSTNAME = $(sys.host)           # CFEngine fills this in

# ...
```

To copy and expand this template, you can use a pattern like this:

```
bundle agent test
{
methods:

 "any" usebundle => get_template("/etc/sudoers","400");
 "any" usebundle => get_template("/etc/hosts","644");

}
```

The the following driving code (based on 'copy then edit') can be placed in a library, after configuring to your environmental locations:

```
bundle agent get_template(final_destination,mode)
{
vars:

 # This needs to ne preconfigured to your site

 "masterfiles"   string => "/home/mark/tmp";
 "this_template" string => lastnode("$(final_destination)","/");

files:

  "$(final_destination).staging"

       comment => "Get template and expand variables for this host",
         perms => mo("400","root"),
     copy_from => remote_cp("$(masterfiles)/templates/$(this_template)","$(policy_server)"),
        action => if_elapsed("60");

  "$(final_destination)"

       comment => "Expand the template",
        create => "true",
     edit_line => expand_template("$(final_destination).staging"),
 edit_defaults => empty,
         perms => mo("$(mode)","root"),
        action => if_elapsed("60");

}
```

3 Low level

3.1 Aborting execution

```
body common control

{
bundlesequence  => { "testbundle"  };

version => "1.2.3";
}

###########################################

body agent control

{
abortbundleclasses => { "invalid.Hr16" };
}

###########################################

bundle agent testbundle
{
vars:

 "userlist" slist => { "xyz", "mark", "jeang", "jonhenrik", "thomas", "eben" };

methods:

 "any" usebundle => subtest("$(userlist)");

}

###########################################

bundle agent subtest(user)

{
classes:

  "invalid" not => regcmp("[a-z][a-z][a-z][a-z]","$(user)");

reports:
 !invalid::

  "User name $(user) is valid at 4 letters";

 invalid::

  "User name $(user) is invalid";
}
```

3.2 ACL file example

```
body common control
{
bundlesequence => { "acls" };
}

#########################################

bundle agent acls

{
files:

  "/media/flash/acl/test_dir"

    depth_search => include_base,
    acl => template;
}

#########################################

body acl template

{
acl_method => "overwrite";
acl_type => "posix";
acl_directory_inherit => "parent";
aces => { "user:*:r(wwx),-r:allow", "group:*:+rw:allow", "mask:x:allow", "all:r"};
}

#########################################

body acl win
```

```
{
acl_method => "overwrite";
acl_type => "ntfs";
acl_directory_inherit => "nochange";
aces => { "user:Administrator:rw", "group:Bad:rwx(Dpo):deny" };
}

#########################################

body depth_search include_base

{
include_basedir => "true";
}
```

3.3 ACL generic example

```
body common control
{
bundlesequence => { "acls" };
}

#########################################

bundle agent acls

{
files:

  "/media/flash/acl/test_dir"

    depth_search => include_base,
    acl => test;
}

#########################################

body acl test

{
acl_type => "generic";
aces => {"user:bob:rwx", "group:staff:rx", "all:r"};
}

#########################################
```

```
body depth_search include_base

{
include_basedir => "true";
}
```

3.4 ACL secret example

```
body common control
{
bundlesequence => { "acls" };
}

#########################################

bundle agent acls

{
files:

 windows::

  "c:\Secret"
    acl => win,
    depth_search => include_base,
    comment => "Secure the secret directory from unauthorized access";
}

#########################################

body acl win

{
acl_method => "overwrite";
aces => { "user:Administrator:rwx" };
}

#########################################

body depth_search include_base

{
include_basedir => "true";
}
```

3.5 Active directory example

```
#########################################################################
#   active_directory.cf - Extract Data From Windows Domain Controllers
#
#   NOTE: Since we don’t supply any credentials in this policy file,
#         the Domain Controller must allow anonymous bind. Also,
#         the user "NT AUTHORITY\ANONYMOUS LOGON" must be granted access
#         to the resources we want to read.
#
#########################################################################

bundle agent active_directory
{
vars:
# NOTE: Edit this to your domain, e.g. "corp", may also need more DC’s after it
  "domain_name" string => "cftesting";
  "user_name"    string => "Guest";


# NOTE: We can also extract data from remote Domain Controllers

dummy.DomainController::
  "domain_controller"  string => "localhost";

  "userlist"    slist => ldaplist(
                                  "ldap://$(domain_controller)",
                                  "CN=Users,DC=$(domain_name),DC=com",
                                  "(objectClass=user)",
                                  "sAMAccountName",
                                  "subtree",
                                  "none");

classes:

dummy.DomainController::

   "gotuser" expression => ldaparray(
                                    "userinfo",
                                    "ldap://$(domain_controller)",
                                    "CN=$(user_name),CN=Users,DC=$(domain_name),DC=com",
                                    "(name=*)",
                                    "subtree",
                                    "none");
reports:
dummy.DomainController::
  "Username is \"$(userlist)\"";

dummy.gotuser::
  "Got user data; $(userinfo[name]) has logged on $(userinfo[logonCount]) times";

}
```

3.6 Active list users directory example

```
# List users from Active Directory through LDAP
# Note: Anonymous LDAP binding must be allowed, and the Anonymous user
# must have read access to CN=Users

bundle agent ldap
{
vars:
   "userlist" slist => ldaplist(
                                    "ldap://cf-win2003",
                                    "CN=Users,DC=domain,DC=cf-win2003",
                                    "(objectClass=user)",
                                    "sAMAccountName",
                                    "subtree",
                                    "none");
reports:
Yr2010::
  "Username: \"$(userlist)\"";
}
```

3.7 Active directory show users example

```
# List users from Active Directory through LDAP
# Note: Anonymous LDAP binding must be allowed, and the Anonymous user
# must have read access to CN=Users and CN=theusername
# Run the agent in verbose mode to see the data

bundle agent ldap
{
classes:
   "gotdata" expression => ldaparray(
                                    "myarray",
                                    "ldap://cf-win2003",
                                    "CN=Test Pilot,CN=Users,DC=domain,DC=cf-win2003",
                                    "(name=*)",
                                    "subtree",
                                    "none");
reports:
gotdata::
  "Got user data";
!gotdata::
  "Did not get user data";
}
```

3.8 Add lines to a file

There are numerous approaches to adding lines to a file. Often the order of a configuration file is unimportant, we just need to ensure settings within it. A simple way of adding lines is show below.

```
body common control

{
any::

  bundlesequence  => { "insert" };
}

#######################################################

bundle agent insert

{
vars:

  "lines" string =>
                "
                One potato
                Two potato
                Three potatoe
                Four
                ";

files:

  "/tmp/test_insert"

            create => "true",
         edit_line => append_if_no_line("$(insert.lines)");

}
```

Also you could write this using a list variable:

```
body common control

{
any::

  bundlesequence  => { "insert" };
}

#######################################################

bundle agent insert

{
vars:

  "lines" slist => { "One potato", "Two potato",
                "Three potatoe", "Four" };

files:

  "/tmp/test_insert"

            create => "true",
         edit_line => append_if_no_line("@(insert.lines)");

}
```

3.9 Add users to passwd and group

Add lines to the password file, and users to group if they are not already there.

```
body common control
{
bundlesequence => { "addpasswd" };
inputs => { "cf_std_library.cf" };
}

bundle agent addpasswd
{
vars:

  # want to set these values by the names of their array keys

  "pwd[mark]" string => "mark:x:1000:100:Mark Burgess:/home/mark:/bin/bash";
  "pwd[fred]" string => "fred:x:1001:100:Right Said:/home/fred:/bin/bash";
  "pwd[jane]" string => "jane:x:1002:100:Jane Doe:/home/jane:/bin/bash";

  "users" slist => getindices("pwd");

files:

  "/etc/passwd"

        create => "true",
     edit_line => append_users_starting("addpasswd.pwd");

  "/etc/group"

       edit_line => append_user_field("users","4","@(addpasswd.users)");

}
```

3.10 Add software packages to the system

```
#
# Package managment
#

body common control
{
bundlesequence => { "packages" };
}

#############################################

bundle agent packages
{
vars:

 "match_package" slist => {
                          "apache2",
                          "apache2-mod_php5",
                          "apache2-prefork",
                          "php5"
                          };
packages:

 solaris::

  "$(match_package)"

     package_policy => "add",
     package_method => solaris;

 redhat|SuSE::

  "$(match_package)"

     package_policy => "add",
     package_method => yum;

}
```

Note you can also arrange to hide all the differences between package managers on an OS basis, but since some OSs have multiple managers, this might not be 100 percent correct.

3.11 Add variable definitions to a file e.g. '/etc/system'

```
body common control
{
bundlesequence => { "setvars" };
inputs => { "cf_std_library.cf" };
}


bundle agent setvars
{
vars:

  # want to set these values by the names of their array keys

  "rhs[lhs1]" string => " Mary had a little pig";
  "rhs[lhs2]" string => "Whose Fleece was white as snow";
  "rhs[lhs3]" string => "And everywhere that Mary went";

  # oops, now change pig -> lamb

files:

  "/tmp/system"

        create => "true",
     edit_line => set_variable_values("setvars.rhs");

}
```

Results in:

```
lhs1= Mary had a little pig
lhs2=Whose Fleece was white as snow
lhs3=And everywhere that Mary went
```

An example of this would be to add variables to '/etc/sysctl.conf' on Linux:

```
body common control
{
bundlesequence => { "setvars" };
inputs => { "cf_std_library.cf" };
}


bundle agent setvars
{
vars:
  # want to set these values by the names of their array keys

  "rhs[net/ipv4/tcp_syncookies]" string => "1";
  "rhs[net/ipv4/icmp_echo_ignore_broadcasts]" string => "1";
  "rhs[net/ipv4/ip_forward]" string => "1";

  # oops, now change pig -> lamb

files:

  "/etc/sysctl"

        create => "true",
     edit_line => set_variable_values("setvars.rhs");

}
```

3.12 Application baseline

```
#########################################################################
#
#   app_baseline.cf - Verify Existence of Applications
#
#   NOTE: Sometimes applications are not correctly installed even
#         though the native package manager reports them to be.
#         Cfengine can check for application-specific configuration
#         and act upon or report any anomalies.
#
#########################################################################


bundle agent app_baseline
{

methods:
windows::
"any" usebundle => detect_adobereader;

}

###

bundle agent detect_adobereader
{
```

```
vars:

windows::
  "value1" string => registryvalue("HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Reader\9.0\Installer",
  "value2" string => registryvalue("HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Reader\9.0\Installer",
  "value3" string => registryvalue("HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\Acrobat Reader\9.0\Installer",

classes:

windows::
  "is_correct" and => {
                       strcmp("$(value1)", "{AC76BA86-7AD7-1033-7B44-A93000000001}"),
                       strcmp("$(value2)", "90003"),
                       islessthan("$(value3)", "10001" )
                      };

 reports:

windows.!is_correct::
 "Adobe Reader is not correctly deployed - got \"$(value1)\", \"$(value2)\", \"$(value3)\"";
}
```

3.13 Array example

```
body common control
{
bundlesequence => { "array" };
}


bundle common g
{
vars:

  "array[1]" string => "one";
  "array[2]" string => "two";
}

bundle agent array
{
vars:

  "localarray[1]" string => "one";
  "localarray[2]" string => "two";
reports:

 linux::

   "Global $(g.array[1]) and $(localarray[2])";
}
```

3.14 Backreferences in filenames

```
######################################################################
#
# File editing - back reference
#
######################################################################


body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  # The back reference in a path only applies to the last link
  # of the pathname, so the (tmp) gets ignored

  "/tmp/(cf3)_(.*)"

       edit_line => myedit("second $(match.2)");


  # but ...

#  "/tmp/cf3_test"
#       create    => "true",
#       edit_line => myedit("second $(match.1)");


}

########################################################

bundle edit_line myedit(parameter)
  {
  vars:
   "edit_variable" string => "private edit variable is $(parameter)";

  insert_lines:

     "$(edit_variable)";

  }
```

3.15 BSD flags

```
body common control
{
bundlesequence => { "test" };
}

bundle agent test
{
files:

 freebsd::

   "/tmp/newfile"

       create => "true",
       perms => setbsd;

}


body perms setbsd
{
bsdflags => { "+uappnd","+uchg", "+uunlnk", "-nodump" };
}
```

3.16 Change directory for command

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

body contain cd(dir)
{
chdir => "${dir}";
useshell => "true";
}

bundle agent example
{
commands:

   "/bin/pwd"
       contain => cd("/tmp");
}
```

3.17 Check file or directory permissions

```
bundle agent check_perms
{
vars:

  "ns_files" slist => {
                      "/local/iu/logs/admin",
                      "/local/iu/logs/security",
                      "/local/iu/logs/updates",
                      "/local/iu/logs/xfer"
                      };
files:

   NameServers::

    "/local/dns/pz"

            perms => mo("644","dns")
```

```
     depth_search => recurse("1"),
      file_select => exclude("secret_file");

    "/local/iu/dns/pz/FixSerial"

            perms => m("755"),
      file_select => plain;

    "$(ns_files)"

            perms => mo("644","dns"),
      file_select => plain;


    "$(ftp)/pub"
             perms => mog("644","root","other");

    "$(ftp)/pub"
             perms => m("644"),
      depth_search => recurse("inf");

    "$(ftp)/etc"        perms => mog("111","root","other");
    "$(ftp)/usr/bin/ls" perms => mog("111","root","other");
    "$(ftp)/dev"        perms => mog("555","root","other");
    "$(ftp)/usr"        perms => mog("555","root","other");
}
```

3.18 Class match example

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

bundle agent example

{
classes:

  "do_it" and => { classmatch(".*_3"), "linux" };

reports:
  do_it::

    "Host matches pattern";

}
```

3.19 Client-server example

```
########################################################
#
# Simple test copy from server connection to cfServer
#
########################################################

 #
 # run this as follows:
 #
 # cf-serverd -f runtest_1.cf [-v]
 # cf-agent   -f runtest_2.cf
 #
 # Notice that the same file configures all parts of cfengine


########################################################

body common control
{
bundlesequence  => { "testbundle" };
version => "1.2.3";
#fips_mode => "true";
}

########################################################

bundle agent testbundle
{
files:

  "/tmp/testcopy"
        comment  => "test copy promise",
    copy_from    => mycopy("/tmp/test-src","127.0.0.1"),
    perms        => system,
    depth_search => recurse("inf"),
    classes      => satisfied("copy_ok");
  "/tmp/testcopy/single_file"

        comment  => "test copy promise",
    copy_from    => mycopy("/tmp/test-src/README","127.0.0.1"),
    perms        => system;

reports:

  copy_ok::

    "Files were copied..";
}

#########################################################

body perms system

{
mode  => "0644";
}

#########################################################

body depth_search recurse(d)

{
depth => "$(d)";
}

#########################################################

body copy_from mycopy(from,server)

{
source      => "$(from)";
servers     => { "$(server)" };
compare     => "digest";
encrypt     => "true";
verify      => "true";
copy_backup => "true";                  #/false/timestamp
purge       => "false";
type_check  => "true";
force_ipv4  => "true";
trustkey => "true";
}
```

```
#########################################################

body classes satisfied(x)
{
promise_repaired => { "$(x)" };
persist_time => "0";
}

#########################################################
# Server config
#########################################################

body server control

{
allowconnects         => { "127.0.0.1" , "::1" };
allowallconnects      => { "127.0.0.1" , "::1" };
trustkeysfrom         => { "127.0.0.1" , "::1" };
# allowusers
}

#########################################################

bundle server access_rules()

{

access:

  "/tmp/test-src"

    admit   => { "127.0.0.1" };
}
```

3.20 Commands example

```
body common control
{
bundlesequence  => { "my_commands" };
inputs => { "cfengine_stdlib.cf" };
}
```

```
bundle agent my_commands
{
commands:

 Sunday.Hr04.Min05_10.myhost::

  "/usr/bin/update_db";

 any::

  "/etc/mysql/start"

      contain => setuid("mysql");

}
```

3.21 Commenting lines in a file

```
#######################################################################
#
# File editing
#
# Normal ordering:
# - delete
# - replace | colum_edit
# - insert
#
#######################################################################


body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
```

```
files:

  "/home/mark/tmp/cf3_test"

       create    => "true",
       edit_line => myedit("second");
}

########################################################

bundle edit_line myedit(parameter)
  {
  vars:

   "edit_variable" string => "private edit variable is $(parameter)";


  replace_patterns:

  # replace shell comments with C comments

   "#(.*)"

      replace_with => C_comment,
     select_region => MySection("New section");

  }

########################################
# Bodies
########################################

body replace_with C_comment

{
replace_value => "/* $(match.1) */"; # backreference 0
occurrences => "all";  # first, last all
}

########################################################

body select_region MySection(x)

{
select_start => "\[$(x)\]";
select_end => "\[.*\]";
}
```

```
######################################################################
#
# Comment lines
#
######################################################################

body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/comment_test"

       create    => "true",
       edit_line => comment_lines_matching;
}

########################################################

bundle edit_line comment_lines_matching
  {
  vars:

    "regexes" slist => { "one.*", "two.*", "four.*" };

  replace_patterns:

   "^($(regexes))$"
      replace_with => comment("# ");
  }

########################################
# Bodies
########################################

body replace_with comment(c)
```

```
{
replace_value => "$(c) $(match.1)";
occurrences => "all";
}



######################################################################
#
# Uncomment lines
#
######################################################################

body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

# try this on some test data like

# one
# two
# mark one
#mark two

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/comment_test"

       create    => "true",
       edit_line => uncomment_lines_matching("\s*mark.*","#");
}

########################################################

bundle edit_line uncomment_lines_matching(regex,comment)
{
replace_patterns:

 "#($(regex))$" replace_with => uncomment;
}

########################################################

body replace_with uncomment
{
replace_value => "$(match.1)";
occurrences => "all";
}
```

3.22 Copy files

```
files:

  "/var/cfengine/inputs"

    handle => "update_policy",
    perms => m("600"),
    copy_from => u_scp("$(master_location)",@(policy_server)),
    depth_search => recurse("inf"),
    file_select => input_files,
    action => immediate;
```

3.23 Copy and flatten directory

```
########################################################
#
# Simple test copy from server connection to cfServer
#
########################################################

 #
 # run this as follows:
 #
 # cf-serverd -f runtest_1.cf [-d2]
 # cf-agent   -f runtest_2.cf
 #
 # Notice that the same file configures all parts of cfengine

########################################################

body common control

{
bundlesequence  => { "testbundle" };
version => "1.2.3";
}

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/testflatcopy"

        comment  => "test copy promise",
    copy_from    => mycopy("/home/mark/LapTop/words","127.0.0.1"),
    perms        => system,
    depth_search => recurse("inf"),
    classes      => satisfied("copy_ok");


  "/home/mark/tmp/testcopy/single_file"

        comment  => "test copy promise",
    copy_from    => mycopy("/home/mark/LapTop/Cfengine3/trunk/README","127.0.0.1"),
    perms        => system;

reports:

  copy_ok::

    "Files were copied..";
}

#########################################################

body perms system

{
mode  => "0644";
}

#########################################################

body depth_search recurse(d)

{
depth => "$(d)";
}
```

```
#########################################################

body copy_from mycopy(from,server)

{
source      => "$(from)";
servers     => { "$(server)" };
compare     => "digest";
verify      => "true";
copy_backup => "true";                  #/false/timestamp
purge       => "false";
type_check  => "true";
force_ipv4  => "true";
trustkey => "true";
collapse_destination_dir => "true";
}

#########################################################

body classes satisfied(x)
{
promise_repaired => { "$(x)" };
persist_time => "0";
}

#########################################################
# Server config
#########################################################

body server control

{
allowconnects         => { "127.0.0.1" , "::1" };
allowallconnects      => { "127.0.0.1" , "::1" };
trustkeysfrom         => { "127.0.0.1" , "::1" };
}

#########################################################

bundle server access_rules()

{
access:

  "/home/mark/LapTop"
```

```
    admit   => { "127.0.0.1" };
}
```

3.24 Copy then edit a file convergently

To convergently chain a copy followed by edit, you need a staging file. First you copy to the staging file. Then you edit the final file and insert the staging file into it as part of the editing. This is convergent with respect to both stages of the process.

```
bundle agent master
{
files:

  "$(final_destination)"

         create => "true",
     edit_line => fix_file("$(staging_file)"),
 edit_defaults => empty,
         perms => mo("644","root"),
        action => ifelapsed("60");
}

#

bundle edit_line fix_file(f)
{
insert_lines:

  "$(f)"

     # insert this into an empty file to reconstruct

     insert_type => "file";

replace_patterns:

    "searchstring"

          replace_with => With("replacestring");
}
```

3.25 Creating files and directories

```
########################################################
#
```

```
# Simple test create files
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/test_plain"

       perms => system,
       create => "true";

  "/home/mark/tmp/test_dir/."

       perms => system,
       create => "true";

}

#########################################################

body perms system

{
mode  => "0640";
}

#########################################################
```

3.26 Database creation

```
body common control
{
bundlesequence => { "databases" };
}

bundle agent databases

{
databases:

  "knowledge_bank/topics"

    database_operation => "create",
    database_type => "sql",
    database_columns => {
                        "demo_name,varchar,256",
                        "demo_comment,varchar,1024",
                        "demo_id,varchar,256",
                        "demo_type,varchar,256",
                        "demo_extra,varchar,26"
                        },

    database_server => myserver;
}

################################################


body database_server myserver
{
none::
 db_server_owner => "postgres";
 db_server_password => "";
 db_server_host => "localhost";
 db_server_type => "postgres";
 db_server_connection_db => "postgres";
any::
 db_server_owner => "root";
 db_server_password => "";
 db_server_host => "localhost";
 db_server_type => "mysql";
 db_server_connection_db => "mysql";
}
```

3.27 Deleting lines from a file

```
body common control
{
bundlesequence => { "test" };
}



bundle agent test
{
files:

  "/tmp/resolv.conf"  # test on "/tmp/resolv.conf" #

     create        => "true",
     edit_line     => resolver,
     edit_defaults => def;

}


#######################################################
# For the library
#######################################################

bundle edit_line resolver

{
vars:

 "search" slist => { "search iu.hio.no cfengine.com", "nameserver 128.39.89.10" };

delete_lines:

  "search.*";

insert_lines:

  "$(search)" location => end;
}

#######################################################

body edit_defaults def
{
empty_file_before_editing => "false";
edit_backup => "false";
max_file_size => "100000";
}

########################################################

body location start

{
# If not line to match, applies to whole text body
before_after => "before";
}

########################################################

body location end

{
# If not line to match, applies to whole text body
before_after => "after";
}
```

3.28 Deleting lines exception

```
########################################################
#
# Simple test editfile
#
########################################################

#
# This assumes a file format like:
#
# [section 1]
#
# lines....
#
# [section 2]
#
```

```
# lines... etc


body common control

{
bundlesequence  => { "testbundle" };
}

########################################################

bundle agent testbundle

{
files:

  "/tmp/passwd_excerpt"

       create    => "true",
       edit_line => MarkNRoot;
}

########################################################

bundle edit_line MarkNRoot
  {
  delete_lines:

       "mark.*|root.*" not_matching => "true";

  }
```

3.29 Editing files

This is a huge topic. See also See Section 3.8 [Add lines to a file], page 69, See Section 3.30 [Editing tabular files], page 98, etc. Editing a file can be complex or simple, depending on needs.

Here is an example of how to comment out lines matching a number of patterns:

```
######################################################################
#
# Comment lines
#
######################################################################

body common control

{
version         =>   "1.2.3";
bundlesequence  => { "testbundle"  };
inputs          => { "cf_std_library.cf" };
}

########################################################

bundle agent testbundle

{
vars:

  "patterns" slist => { "finger.*", "echo.*", "exec.*", "rstat.*",
                                               "uucp.*", "talk.*" };

files:

  "/etc/inetd.conf"

      edit_line => comment_lines_matching("@(testbundle.patterns)","#");
}
```

3.30 Editing tabular files

```
######################################################################
#
# File editing
#
# Normal ordering:
# - delete
# - replace | colum_edit
# - insert
#
######################################################################


body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
vars:

 "userset" slist => { "one-x", "two-x", "three-x" };

files:

  # Make a copy of the password file

  "/home/mark/tmp/passwd"

       create    => "true",
       edit_line => SetUserParam("mark","6","/set/this/shell");

  "/home/mark/tmp/group"

       create    => "true",
       edit_line => AppendUserParam("root","4","@(userset)");

commands:
  "/bin/echo" args => "$(userset)";

}

########################################################

bundle edit_line SetUserParam(user,field,val)
  {
  field_edits:

   "$(user):.*"

      # Set field of the file to parameter

      edit_field => col(":","$(field)","$(val)","set");
  }

########################################################

bundle edit_line AppendUserParam(user,field,allusers)
  {
  vars:

    "val" slist => { @(allusers) };

  field_edits:

   "$(user):.*"

      # Set field of the file to parameter

      edit_field => col(":","$(field)","$(val)","alphanum");

  }

########################################
# Bodies
########################################

body edit_field col(split,col,newval,method)

{
field_separator => "$(split)";
select_field    => "$(col)";
value_separator  => ",";
field_value     => "$(newval)";
field_operation => "$(method)";
```

```
extend_fields => "true";
}
```

3.31 Environment variables

3.32 Execresult example

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

bundle agent example

{
vars:

  "my_result" string => execresult("/bin/ls /tmp","noshell");

reports:

  linux::

    "Variable is $(my_result)";

}
```

3.33 Inserting lines in a file

```
#######################################################
#
# Insert a number of lines with vague whitespace
#
#######################################################

body common control

{
any::

  bundlesequence  => { "insert" };
}


#######################################################

bundle agent insert

{
vars:

  "v" string => "  One potato";

files:

  "/tmp/test_insert"

            create => "true",
         edit_line => Insert("$(insert.v)");

}

#######################################################
# For the library
#######################################################

bundle edit_line Insert(name)

{
insert_lines:

  "  $(name)"
```

```
      whitespace_policy => { "ignore_leading", "ignore_embedded" };

}

#######################################################

body edit_defaults empty

{
empty_file_before_editing => "true";
}



#######################################################
#
# Insert a number of lines
#
#######################################################

body common control

{
any::

  bundlesequence  => { "insert" };
}


#######################################################

bundle agent insert

{
vars:

  "v" string => "
                One potato
                Two potato
                Three potatoe
                Four
                ";

files:

  "/tmp/test_insert"
```

```
            create => "true",
         edit_line => Insert("$(insert.v)"),
     edit_defaults => empty;

}

#######################################################
# For the library
#######################################################

bundle edit_line Insert(name)

{
insert_lines:

  "Begin$(const.n)$(name)$(const.n)End";

}

#######################################################

body edit_defaults empty

{
empty_file_before_editing => "false";
}



#######################################################
#
# Insert a number of lines
#
#######################################################

body common control

{
any::

  bundlesequence  => { "insert" };
}


#######################################################
```

```
bundle agent insert

{
vars:

  "v" slist => {
                "One potato",
                "Two potato",
                "Three potatoe",
                "Four"
               };

files:

  "/tmp/test_insert"

            create => "true",
         edit_line => Insert("@(insert.v)");
   #  edit_defaults => empty;

}

#######################################################
# For the library
#######################################################

bundle edit_line Insert(name)

{
insert_lines:

  "$(name)";

}

#######################################################

body edit_defaults empty

{
empty_file_before_editing => "true";
}
```

3.34 Get a list of users

```
#######################################################
#
# GetUsers
#
#######################################################

body common control

{
any::

  bundlesequence  => {
                     test
                     };
}

#######################################################

bundle agent test

{
vars:

  "allusers" slist => getusers("zenoss,mysql,at","12,0");

reports:

 linux::

  "Found user $(allusers)";

}
```

3.35 Global classes

```
body common control
{
bundlesequence => { "g","tryclasses_1", "tryclasses_2" };

}

#################################
```

```
bundle common g
{
classes:

  "one" expression => "any";

  "client_network" expression => iprange("128.39.89.0/24");
}

#################################

bundle agent tryclasses_1
{
classes:

  "two" expression => "any";
}

#################################

bundle agent tryclasses_2
{
classes:

  "three" expression => "any";

reports:

  one.three.!two::

    "Success";
}


#################################
```

3.36 Guest environments

```
#######################################################
#
# Guest environments
#
#######################################################

body common control
```

```
{
bundlesequence  => { "my_vm_cloud" };
}

#######################################################

bundle agent my_vm_cloud

{
guest_environments:

 scope||any::  # These should probably be in class "any" to ensure uniqueness

   "test2"

       environment_resources => my_environment,
       environment_interface => vnet("eth0,192.168.1.100/24"),
       environment_type      => "test",
       environment_state     => "create",
       environment_host      => "atlas";


   "test2"

       environment_resources => my_environment,
       environment_interface => vnet("eth0,192.168.1.101/24"),
       environment_type      => "test",
       environment_state     => "delete",
       environment_host      => "atlas";


   "test4"

       environment_resources => my_environment,
       environment_interface => vnet("eth0,192.168.1.102/24"),
       environment_type      => "test",
       environment_state     => "create",
       environment_host      => "atlas";

    "network1"
       environment_type      => "test_net",
       environment_state     => "create",
       environment_host      => "atlas";


      # default environment_state => "create" on host, and "suspended elsewhere"
}

#######################################################

body environment_resources my_environment
{
env_cpus => "2";
env_memory => "512"; # in KB
env_disk => "1024";  # in MB
}

#######################################################

body environment_interface vnet(primary)
{
env_name      => "$(this.promiser)";
env_addresses => { "$(primary)" };

host1::

  env_network => "default_vnet1";

host2::

  env_network => "default_vnet2";

}
```

3.37 Hello world

```
# Hard promises

body common control
{
bundlesequence => { "hello" };
}

# soft promises

bundle agent hello
{
reports:

 linux::

   "Hello world!";
}
```

3.38 LDAP interactions

```
#

body common control
{
bundlesequence => { "ldap" , "followup"};
}

###################################################################################################
#  NOTE!! relying on LDAP or other network data without validation is EXTREMELY dangerous.
#         You could destroy a system by assuming that the service will respond with a
#         sensible result. Cfengine does not recommend reliance on network services in configuration
###################################################################################################

bundle agent ldap
{
vars:

   # Get the first matching value for "uid"

  "value" string => ldapvalue("ldap://eternity.iu.hio.no","dc=cfengine,dc=com","(sn=User)","uid","su

   # Geta all matching values for "uid" - should be a single record match

  "list" slist =>  ldaplist("ldap://eternity.iu.hio.no","dc=cfengine,dc=com","(sn=User)","uid","subt

classes:

   "gotdata" expression => ldaparray("myarray","ldap://eternity.iu.hio.no","dc=cfengine,dc=com","(ui

   "found" expression => regldap("ldap://eternity.iu.hio.no","dc=cfengine,dc=com","(sn=User)","uid",

reports:

 linux::

   "LDAP VALUE $(value) found";
   "LDAP LIST VALUE $(list)";

 gotdata::

   "Found specific entry data  ...$(ldap.myarray[uid]),$(ldap.myarray[gecos]), etc";

  found::
    "Matched regex";

}

bundle agent followup

{
reports:

 linux::

   "Different bundle  ...$(ldap.myarray[uid]),$(ldap.myarray[gecos]), etc";

}
```

3.39 Linking files

```
######################################################################
#
# File editing
#
# Normal ordering:
# - delete
# - replace | colum_edit
# - insert
#
######################################################################


body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
files:

  # Make a copy of the password file
  "/home/mark/tmp/passwd"

       link_from     => linkdetails("/etc/passwd"),
       move_obstructions => "true";


  "/home/mark/tmp/linktest"

       link_from     => linkchildren("/usr/local/sbin");


#child links
}

#########################################################

body link_from linkdetails(tofile)

{
source        => "$(tofile)";
link_type     => "symlink";
when_no_source  => "force";      # kill
}

#########################################################

body link_from linkchildren(tofile)

{
source        => "$(tofile)";
link_type     => "symlink";
when_no_source  => "force";      # kill
link_children => "true";
when_linking_children => "if_no_such_file"; # "override_file";
}
```

Removing deadlinks from a directory:

```
#######################################################
#
# Test dead link removal
#
#######################################################

body common control
```

```
   {
   any::

      bundlesequence  => {
                         "testbundle"
                         };
   }


############################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/test_to" -> "someone"

      depth_search => recurse("inf"),
      perms => modestuff,
      action => tell_me;

}

############################################

body depth_search recurse(d)

{
rmdeadlinks => "true";
depth => "$(d)";
}

############################################

body perms modestuff

{
mode => "o-w";
}

############################################

body action tell_me

{
report_level => "inform";
}
```

3.40 Listing files-pattern in a directory

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

bundle agent example

{
vars:

  "ls" slist => lsdir("/etc","p.*","true");

reports:

  !sdfkjh::

    "ls: $(ls)";

}
```

3.41 Locate and transform files

```
#######################################################
#
# Compressing files
#
#######################################################

body common control
   {
   any::

      bundlesequence  => {
                         "testbundle"
                         };
   version => "1.2.3";
   }

############################################

bundle agent testbundle

{
files:

  "/home/mark/tmp/testcopy"

    file_select => pdf_files,
    transformer => "/usr/bin/gzip $(this.promiser)",
    depth_search => recurse("inf");

}

############################################

body file_select pdf_files

{
leaf_name => { ".*.pdf" , ".*.fdf" };
file_result => "leaf_name";
}

############################################

body depth_search recurse(d)

{
depth => "$(d)";
}
```

3.42 Logging

```
body common control
{
bundlesequence => { "test" };
}

bundle agent test
{
```

```
vars:

  "software" slist => { "/root/xyz", "/tmp/xyz" };

files:

  "$(software)"

    create => "true",
     action => logme("$(software)");

}

#

body action logme(x)
{
log_kept => "/tmp/private_keptlog.log";
log_failed => "/tmp/private_faillog.log";
log_repaired => "/tmp/private_replog.log";
log_string => "$(sys.date) $(x) promise status";
}



body common control
{
bundlesequence => { "one" };
}



bundle agent one
{
files:

  "/tmp/xyz"

       create => "true",
       action => log;

}

body action log
{
log_level => "inform";
}
```

3.43 Measurements

```
#cop measurements,example

#######################################################
#
# Test file:
#
# First line
# Blonk blonk bnklkygsuilnm
#
#######################################################

body common control
{
bundlesequence => { "report" };
}

#######################################################

body monitor control
{
forgetrate => "0.7";
histograms => "true";
}

#######################################################

bundle agent report
{
reports:

 cfengine_3::

   "
   Free memory read at $(mon.av_free_memory_watch)
   cf_monitord read $(mon.value_monitor_self_watch)
   ";
}

#######################################################

bundle monitor watch
{
measurements:
```

```
# Test 1 - extract string matching

"/home/mark/tmp/testmeasure"

    handle => "blonk_watch",
    stream_type => "file",
    data_type => "string",
    history_type => "weekly",
    units => "blonks",
    match_value => find_blonks,
    action => sample_min("10");

# Test 2 - follow a special process over time
# using cfengine's process cache to avoid resampling

 "/var/cfengine/state/cf_rootprocs"

    handle => "monitor_self_watch",
    stream_type => "file",
    data_type => "int",
    history_type => "static",
    units => "kB",
    match_value => proc_value(".*cf-monitord.*",
                              "root\s+[0-9.]+\s+[0-9.]+\s+[0-9.]+\s+[0-9.]+\s+([0-9]+).*");

# Test 3, discover disk device information

"/bin/df"

    handle => "free_disk_watch",
    stream_type => "pipe",
    data_type => "slist",
    history_type => "static",
    units => "device",
    match_value => file_system;
    # Update this as often as possible

# Test 4

 "/tmp/file"

       handle => "line_counter",
  stream_type => "file",
    data_type => "counter",
  match_value => scanlines("MYLINE.*"),
 history_type => "log";
```

```
}

##########################################################

body match_value scanlines(x)
{
select_line_matching => "^$(x)$";
}

##########################################################

body action sample_min(x)
{
ifelapsed => "$(x)";
expireafter => "$(x)";
}

##########################################################

body match_value find_blonks
{
select_line_number => "2";
extraction_regex => "Blonk blonk ([blonk]+).*";
}

##########################################################

body match_value free_memory # not willy!
{
select_line_matching => "MemFree:.*";
extraction_regex => "MemFree:\s+([0-9]+).*";
}

##########################################################

body match_value proc_value(x,y)
{
select_line_matching => "$(x)";
extraction_regex => "$(y)";
}

##########################################################

body match_value file_system
{
select_line_matching => "/.*";
extraction_regex => "(.*)";
```

```
}
```

3.44 Methods

```
body common control

{
bundlesequence  => { "testbundle"  };

version => "1.2.3";
}

#########################################

bundle agent testbundle
{
vars:

 "userlist" slist => { "mark", "jeang", "jonhenrik", "thomas", "eben" };

methods:

 "any" usebundle => subtest("$(userlist)");

}

#########################################

bundle agent subtest(user)

{
commands:

 "/bin/echo Fix $(user)";

reports:

 linux::

  "Finished doing stuff for $(user)";
}
```

3.45 Method validation

```
body common control

{
bundlesequence  => { "testbundle"  };

version => "1.2.3";
}

###########################################

body agent control

{
abortbundleclasses => { "invalid" };
}

###########################################

bundle agent testbundle
{
vars:

 "userlist" slist => { "xyz", "mark", "jeang", "jonhenrik", "thomas", "eben" };

methods:

 "any" usebundle => subtest("$(userlist)");

}

###########################################

bundle agent subtest(user)

{
classes:

  "invalid" not => regcmp("[a-z][a-z][a-z][a-z]","$(user)");

reports:

 !invalid::
  "User name $(user) is valid at 4 letters";

 invalid::

  "User name $(user) is invalid";
}
```

3.46 Mount NFS filesystem

```
#
# cfengine 3
#
# cf-agent -f ./cftest.cf -K
#

body common control

{
bundlesequence => { "mounts" };
}

#

bundle agent mounts

{
storage:

  "/mnt" mount  => nfs("slogans.iu.hio.no","/home");

}

######################################################################

body mount nfs(server,source)

{
mount_type => "nfs";
mount_source => "$(source)";
mount_server => "$(server)";
#mount_options => { "rw" };
edit_fstab => "true";
unmount => "true";
}
```

3.47 Ordering promises

This counts to five by default. If we change '`/bin/echo one`' to '`/bin/echox one`', then the command will fail, causing us to skip five and go to six instead.

This shows how dependencies can be chained in spite of the order of promises in the bundle.

Normally the order of promises in a bundle is followed, within each promise type, and the types are ordered according to *normal ordering*.

```
##################################################################
#
# cfengine 3 - ordering promises into dependent chains
#
##
#
# cf-agent -f ./cftest.cf -K
#
##################################################################

body common control

{
bundlesequence => { "order" };
}

##################################################################

bundle agent order

{
vars:

 "list" slist => { "three", "four" };

commands:

 ok_later::

   "/bin/echo five";

 otherthing::

   "/bin/echo six";

 any::


  "/bin/echo one"     classes => d("ok_later","otherthing");
  "/bin/echo two";
  "/bin/echo $(list)";

 preserved_class::

  "/bin/echo seven";

}

############################################

body classes d(if,else)

{
promise_repaired => { "$(if)" };
repair_failed => { "$(else)" };
persist_time => "0";
}
```

3.48 Process management

```
body common control
{
bundlesequence => { "test" };
}



bundle agent test
{
processes:

 "sleep"

   signals => { "term", "kill" };

}


########################################################
#
# Simple test processes
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
processes:

 "sleep"

    process_count   => up("sleep");

reports:

 sleep_out_of_control::
   "Out of control";
}

########################################################

body process_count up(s)

{
match_range => "5,10"; # or irange("1","10");
out_of_range_define => { "$(s)_out_of_control" };
}



########################################################
#
# Simple test processes
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
processes:

 ".*"

    process_select  => proc_finder("a.*"),
    process_count   => up("cfservd");
}

########################################################

body process_count up(s)

{
match_range => "1,10"; # or irange("1","10");
out_of_range_define => { "$(s)_out_of_control" };
}
```

```
########################################################

body process_select proc_finder(p)

{
stime_range => irange(ago("0","0","0","2","0","0"),now);
process_result => "stime";
}




########################################################
#
# Simple test processes
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
processes:

 ".*"

    process_select  => proc_finder("a.*"),
    process_count   => up("cfservd");
}

########################################################

body process_count up(s)

{
match_range => "1,10"; # or irange("1","10");
out_of_range_define => { "$(s)_out_of_control" };
}
```

```
########################################################

body process_select proc_finder(p)

{
process_owner  => { "avahi", "bin" };
command        => "$(p)";
pid            => "100,199";
vsize          => "0,1000";
process_result => "command.(process_owner|vsize)";
}



body common control
{
bundlesequence => { "process_restart" };
}

#########################################################

bundle agent process_restart
{
processes:

  "/usr/bin/daemon"
        restart_class => "launch";

commands:

  launch::

   "/usr/bin/daemon";

}


body common control
{
bundlesequence => { "process_restart" };
}

#########################################################

bundle agent process_restart
{
vars:
  "component" slist => {
                       "cf-monitord",
                       "cf-serverd",
                       "cf-execd"
                       };
processes:

  "$(component)"
        restart_class => canonify("start_$(component)");

commands:

   "/var/cfengine/bin/$(component)"
       ifvarclass => canonify("start_$(component)");

}


########################################################
#
# Simple test process restart
#
########################################################

body common control

{
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{
processes:

 "cfservd"

        process_count   => up("cfservd");

 cfservd_out_of_control::

   "cfservd"

        signals         => { "stop" , "term" },
        restart_class   => "start_cfserv";


commands:

  start_cfserv::

    "/usr/local/sbin/cfservd";

}

########################################################

body process_count up(s)

{
match_range => "1,10"; # or irange("1","10");
out_of_range_define => { "$(s)_out_of_control" };
}
```

3.49 Read from a TCP socket

```
body common control

{
bundlesequence  => { "example" };
}

###########################################################

bundle agent example

{
vars:

  "my80" string => readtcp("research.iu.hio.no","80","GET /index.php HTTP/1.1$(const.r)$(const.n)Hos

classes:

  "server_ok" expression => regcmp(".*200 OK.*\n.*","$(my80)");

reports:

  server_ok::

    "Server is alive";

  !server_ok::

    "Server is not responding - got $(my80)";
}
```

3.50 Resolver management

```
#######################################################
#
# Resolve conf
#
#######################################################

bundle common g # globals
{
```

```
vars:

 "searchlist"  slist => {
                        "search iu.hio.no",
                        "search cfengine.com"
                        };

 "nameservers" slist => {
                        "128.39.89.10",
                        "128.39.74.16",
                        "192.168.1.103"
                        };
classes:

  "am_name_server" expression => reglist("@(nameservers)","$(sys.ipv4[eth1])");
}

#######################################################

body common control

{
any::

  bundlesequence  => {
                     "g",
                     resolver(@(g.searchlist),@(g.nameservers))
                     };

  domain => "iu.hio.no";
}

#######################################################

bundle agent resolver(s,n)

{
files:

  # When passing parameters down, we have to refer to
  # a source context

  "$(sys.resolv)"  # test on "/tmp/resolv.conf" #

      create        => "true",
      edit_line     => doresolv("@(this.s)","@(this.n)"),
      edit_defaults => reconstruct;
 # or edit_defaults => modify
}

#######################################################
# For the library
#######################################################

bundle edit_line doresolv(s,n)

{
vars:

 "line" slist => { @(s), @(n) };

insert_lines:

  "$(line)";

}

#######################################################

body edit_defaults reconstruct
{
empty_file_before_editing => "true";
edit_backup => "false";
max_file_size => "100000";
}

#######################################################

body edit_defaults modify
{
empty_file_before_editing => "false";
edit_backup => "false";
max_file_size => "100000";
}
```

3.51 Search and replace text

```
######################################################################
#
# File editing
#
# Normal ordering:
# - delete
# - replace | colum_edit
# - insert
#
######################################################################


body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{

files:

  "/tmp/replacestring"

       create    => "true",
       edit_line => myedit("second");
}

########################################################

bundle edit_line myedit(parameter)
  {
  vars:

   "edit_variable" string => "private edit variable is $(parameter)";


  replace_patterns:
```

```
  # replace shell comments with C comments

   "puppet"

      replace_with => With("cfengine 3");

  }

########################################
# Bodies
########################################

body replace_with With(x)

{
replace_value => "$(x)";
occurrences => "first";
}

########################################

body select_region MySection(x)

{
select_start => "\[$(x)\]";
select_end => "\[.*\]";
}
```

3.52 Selecting a region in a file

```
body common control

{
version => "1.2.3";
bundlesequence  => { "testbundle"  };
}

########################################################

bundle agent testbundle

{

files:
  "/tmp/testfile"

       create    => "true",
       edit_line => myedit("second");
}

########################################################

bundle edit_line myedit(parameter)
  {
  vars:

   "edit_variable" string => "private edit variable is $(parameter)";


  replace_patterns:

  # comment out lines after start

   "([^#].*)"

      replace_with => comment,
     select_region => ToEnd("Start.*");

  }

########################################
# Bodies
########################################

body replace_with comment

{
replace_value => "# $(match.1)"; # backreference 0
occurrences => "all";  # first, last all
}

########################################################

body select_region ToEnd(x)

{
select_start => "$(x)";
}
```

3.53 Service management (windows)

```
body common control

{
bundlesequence  => { "winservice" };
}

###########################################################

bundle agent winservice

{
vars:

  "bad_services" slist => { "Alerter",  "ClipSrv" };

services:

 windows::

  "$(bad_services)"

       service_policy => "disable",
       comment => "Disable services that create security issues";
}
```

3.54 Set up a PXE boot server

Use CFEngine to set up a PXE boot server.

```
body common control
{
 bundlesequence => { "pxe" };
 inputs => { "/var/cfengine/inputs/cfengine_stdlib.cf" };
}



#
# PXE boot server
#



bundle agent pxe
{
vars:

  "software" slist => {
```

```
                      "atftp",
                      "dhcp-server",
                      "syslinux",
                      "apache2"
                      };


   "dirs" slist => {
                   "/tftpboot",
                   "/tftpboot/CFEngine/rpm",
                   "/tftpboot/CFEngine/inputs",
                   "/tftpboot/pxelinux.cfg",
                   "/tftpboot/kickstart",
                   "/srv/www/repos"
                   };

   "tmp_location"    string => "/tftpboot/CFEngine/inputs";

   # Distros that we can install


   "rh_distros" slist => { "4.7", "5.2" };
   "centos_distros" slist => { "5.2" };

   # File contents of atftp configuration


   "atftpd_conf" string =>
       "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promising state.      ###

###########################################


ATFTPD_OPTIONS=\"--daemon \"
ATFTPD_USE_INETD=\"no\"
ATFTPD_DIRECTORY=\"/tftpboot\"
ATFTPD_BIND_ADDRESSES=\"\"
       ";

   # File contents of DHCP configuration
   "dhcpd" string =>
       "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promising state.      ###

###########################################


DHCPD_INTERFACE=\"eth0\"
DHCPD_RUN_CHROOTED=\"yes\"
DHCPD_CONF_INCLUDE_FILES=\"\"
DHCPD_RUN_AS=\"dhcpd\"
DHCPD_OTHER_ARGS=\"\"
DHCPD_BINARY=\"\"
       ";

   "dhcpd_conf" string =>
       "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promising state.      ###

###########################################


allow booting;
allow bootp;
ddns-update-style none; ddns-updates off;
 subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.20 192.168.0.254;
   default-lease-time 3600;
   max-lease-time 4800;
   option routers 192.168.0.1;
   option domain-name \"test.CFEngine.com\";
   option domain-name-servers 192.168.0.1;
   next-server 192.168.0.1;
   filename \"pxelinux.0\";
 }
 group {
   host node1 {
     # Dummy machine

     hardware ethernet 00:0F:1F:94:FE:07;
     fixed-address 192.168.0.11;
     option host-name \"node1\";
   }
   host node2 {
     # Dell Inspiron 1150

     hardware ethernet 00:0F:1F:0E:70:E7;
     fixed-address 192.168.0.12;
     option host-name \"node2\";
   }
 }
        ";

   # File contains of Apache2 HTTP configuration


   "httpd_conf" string =>
        "
# Repository for RHEL5

<Directory /srv/www/repos>
Options Indexes
AllowOverride None
</Directory>
Alias /repos /srv/www/repos

# PXE boot server

<Directory /tftpboot/distro/RHEL/5.2>
Options Indexes
AllowOverride None
</Directory>
Alias /distro/rhel/5.2 /tftpboot/distro/RHEL/5.2

<Directory /tftpboot/distro/RHEL/4.7>
Options Indexes
AllowOverride None
</Directory>
Alias /distro/rhel/4.7 /tftpboot/distro/RHEL/4.7
```

```
<Directory /tftpboot/distro/CentOS/5.2>
Options Indexes
AllowOverride None
</Directory>
Alias /distro/centos/5.2 /tftpboot/distro/CentOS/5.2

<Directory /tftpboot/kickstart>
Options Indexes
AllowOverride None
</Directory>
Alias /kickstart /tftpboot/kickstart

<Directory /tftpboot/CFEngine>
Options Indexes
AllowOverride None
</Directory>
Alias /CFEngine /tftpboot/CFEngine
        ";

   # File contains of Kickstart for RHEL5 configuration


   "kickstart_rhel5_conf" string =>
        "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promissing state.     ###

###########################################


auth  --useshadow  --enablemd5
bootloader --location=mbr
clearpart --all --initlabel
graphical
firewall --disabled
firstboot --disable
key 77244a6377a8044a
keyboard no
lang en_US
logging --level=info
url --url=http://192.168.0.1/distro/rhel/5.2
network --bootproto=dhcp --device=eth0 --onboot=on
```

```
reboot
rootpw --iscrypted $1$eOnXdDPF$279sQ//zry6rnQktkATeM0
selinux --disabled
timezone --isUtc Europe/Oslo
install
part swap --bytes-per-inode=4096 --fstype=\"swap\" --recommended
part / --bytes-per-inode=4096 --fstype=\"ext3\" --grow --size=1

%packages
@core
@base
db4-devel
openssl-devel
gcc
flex
bison
libacl-devel
libselinux-devel
pcre-devel
#httpd

device-mapper-multipath
-sysreport

%post
cd /root
rpm -i http://192.168.0.1/CFEngine/rpm/CFEngine-3.0.1b1-1.el5.i386.rpm
#/sbin/chkconfig httpd on

cd /etc/yum.repos.d
wget http://192.168.0.1/repos/RHEL5.Base.repo
rpm --import /etc/pki/rpm-gpg/*
yum clean all
yum update
mkdir -p /root/CFEngine_init
cd /root/CFEngine_init
wget -nd -r http://192.168.0.1/CFEngine/inputs/
/var/cfengine/bin/cf-agent -B
/var/cfengine/bin/cf-agent
        ";

   # File contains of PXElinux boot menu


   "pxelinux_boot_menu" string =>
        "
###########################################
```

```
### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promissing state.     ###

###########################################


boot options:
     rhel5   - install 32 bit i386 RHEL 5.2             (MANUAL)
     rhel5w  - install 32 bit i386 RHEL 5.2             (AUTO)
     rhel4   - install 32 bit i386 RHEL 4.7 AS          (MANUAL)
     centos5 - install 32 bit i386 CentOS 5.2 (Desktop) (MANUAL)
        ";
   # File contains of PXElinux default configuration


   "pxelinux_default" string =>
        "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promissing state.     ###

###########################################


default rhel5
timeout 300
prompt 1
display pxelinux.cfg/boot.msg
F1 pxelinux.cfg/boot.msg

# install i386 RHEL 5.2

label rhel5
   kernel vmlinuz-RHEL5U2
   append initrd=initrd-RHEL5U2 load_ramdisk=1 ramdisk_size=16384 install=http://192.168.0.1/distro/

# install i386 RHEL 5.2 using Kickstart

label rhel5w
```

```
   kernel vmlinuz-RHEL5U2
   append initrd=initrd-RHEL5U2 load_ramdisk=1 ramdisk_size=16384 ks=http://192.168.0.1/kickstart/ki

# install i386 RHEL 4.7

label rhel4
   kernel vmlinuz-RHEL4U7
   append initrd=initrd-RHEL4U7 load_ramdisk=1 ramdisk_size=16384 install=http://192.168.0.1/distro/

# install i386 CentOS 5.2

label centos5
   kernel vmlinuz-CentOS5.2
   append initrd=initrd-CentOS5.2 load_ramdisk=1 ramdisk_size=16384 install=http://192.168.0.1/distr
        ";

   # File contains of specified PXElinux default to be a RHEL5 webserver


   "pxelinux_rhel5_webserver" string =>
        "
###########################################

### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promissing state.     ###

###########################################


# install i386 RHEL 5.2 using Kickstart

default rhel5w
label rhel5w
   kernel vmlinuz-RHEL5U2
   append initrd=initrd-RHEL5U2 load_ramdisk=1 ramdisk_size=16384 ks=http://192.168.0.1/kickstart/ki
        ";

   # File contains of a local repository for RHEL5


   "rhel5_base_repo" string =>
        "
###########################################
```

```
### This file is protected by CFEngine. ###

### Whatever you do, it will be changed ###

###     back to a promissing state.     ###

###########################################


# Local Repository

[Server]
name=Server
baseurl=http://192.168.0.1/repos/rhel5/Server/
enable=1
[VT]
name=VT
baseurl=http://192.168.0.1/repos/rhel5/VT/
enable=1
[Cluster]
name=Cluster
baseurl=http://192.168.0.1/repos/rhel5/Cluster/
enable=1
[ClusterStorage]
name=Cluster Storage
baseurl=http://192.168.0.1/repos/rhel5/ClusterStorage/
enable=1
        ";
#####################################################



files:

 packages_ok::

  # Create files/dirs and edit the new files


  "/tftpboot/distro/RHEL/$(rh_distros)/."
       create => "true";

  "/tftpboot/distro/CentOS/$(centos_distros)/."
       create => "true";

  "$(dirs)/."
       create => "true";
  "/tftpboot/pxelinux.cfg/boot.msg"
       create => "true",
       perms => mo("644","root"),
       edit_line => append_if_no_line("$(pxelinux_boot_menu)"),
       edit_defaults => empty;

  "/tftpboot/pxelinux.cfg/default"
       create => "true",
       perms => mo("644","root"),
       edit_line => append_if_no_line("$(pxelinux_default)"),
       edit_defaults => empty;

  "/tftpboot/pxelinux.cfg/default.RHEL5.webserver"
       create => "true",
       perms => mo("644","root"),
       edit_line => append_if_no_line("$(pxelinux_rhel5_webserver)"),
       edit_defaults => empty;

  "/tftpboot/kickstart/kickstart-RHEL5U2.cfg"
       create => "true",
       perms => mo("644","root"),
       edit_line => append_if_no_line("$(kickstart_rhel5_conf)"),
       edit_defaults => empty;

  "/srv/www/repos/RHEL5.Base.repo"
       create => "true",
       perms => mo("644","root"),
       edit_line => append_if_no_line("$(rhel5_base_repo)"),
       edit_defaults => empty;

  # Copy files


  "/tftpboot"

    copy_from => local_cp("/usr/share/syslinux"),
    depth_search => recurse("inf"),
    file_select => pxelinux_files,
    action => immediate;

  "$(tmp_location)"

    perms => m("644"),
    copy_from => local_cp("/var/cfengine/inputs"),
    depth_search => recurse("inf"),
    file_select => input_files,
    action => immediate;

  # Edit atftp, dhcp and apache2 configurations


  "/etc/sysconfig/atftpd"
     edit_line => append_if_no_line("$(atftpd_conf)"),
     edit_defaults => empty,
     classes => satisfied("atftpd_ready");

  "/etc/sysconfig/dhcpd"
     edit_line => append_if_no_line("$(dhcpd)"),
     edit_defaults => empty;

  "/etc/dhcpd.conf"
     edit_line => append_if_no_line("$(dhcpd_conf)"),
     edit_defaults => empty,
     classes => satisfied("dhcpd_ready");

  "/etc/apache2/httpd.conf"
     edit_line => append_if_no_line("$(httpd_conf)"),
     edit_defaults => std_defs,
     classes => satisfied("apache2_ok");

  # Make a static link


  "/tftpboot/pxelinux.cfg/C0A8000C"
     link_from => mylink("/tftpboot/pxelinux.cfg/default.RHEL5.webserver");

  # Hash comment some lines for apaches


  apache2_ok::
  "/etc/apache2/httpd.conf"
     edit_line => comment_lines_matching_apache2("#"),
     classes => satisfied("apache2_ready");

commands:

  # Restart services

  atftpd_ready::
  "/etc/init.d/atftpd restart";

  dhcpd_ready::
  "/etc/init.d/dhcpd restart";
  apache2_ready::
  "/etc/init.d/apache2 restart";


#####################################################


packages:

 ipv4_192_168_0_1::
 # Only the PXE boot server


 "$(software)"

     package_policy => "add",
     package_method => zypper,
     classes => satisfied("packages_ok");

}

#####################################################

########### *** Bodies are here *** #################

#####################################################


body file_select pxelinux_files

{
leaf_name => { "pxelinux.0" };

file_result => "leaf_name";
}

#####################################################


body copy_from mycopy_local(from,server)

{
source      => "$(from)";
compare     => "digest";
}
```

```
#########################################################


body link_from mylink(x)
{
source => "$(x)";
link_type => "symlink";
}

#######################################################


body classes satisfied(new_class)

{
promise_kept => { "$(new_class)"};
promise_repaired => { "$(new_class)"};
}

#######################################################


bundle edit_line comment_lines_matching_apache2(comment)
  {

  vars:
   "regex" slist => { "\s.*Options\sNone", "\s.*AllowOverride\sNone", "\s.*Deny\sfrom\sall" };

  replace_patterns:

   "^($(regex))$"
      replace_with => comment("$(comment)");
  }

#######################################################

body file_select input_files
{
 leaf_name => { ".*.cf",".*.dat",".*.txt" };
 file_result => "leaf_name";
}

#######################################################
```

3.55 Tidying garbage files

Emulating the 'tidy' feature of CFEngine 2.

```
#######################################################
#
# Deleting files, like cf2 tidy age=0 r=inf
#
#######################################################

body common control

{
 any::

  bundlesequence  => { "testbundle" };
}

############################################

bundle agent testbundle

{
files:

  "/tmp/test"

    delete => tidyfiles,
    file_select => zero_age,
    depth_search => recurse("inf");
}

#########################################################

body depth_search recurse(d)

{
#include_basedir => "true";
depth => "$(d)";
}

#########################################################

body delete tidy

{
dirlinks => "delete";
rmdirs   => "false";
}

#########################################################

body file_select zero_age

#
# we can build old "include", "exclude", and "ignore"
# from these as standard patterns - these bodies can
# form a library of standard patterns
#

{
mtime     => irange(ago(1,0,0,0,0,0),now);
file_result => "mtime";
}
```

3.56 Software distribution

```
#########################################################################
#
#   software_local.cf - Application Deployment From Directory Repository
#
#   NOTE: Windows needs to support WMI queries about installed msi files
#         in order for Cfengine to detect them. On Windows 2003,
#         go to Control Panel -> Add/Remove Programs ->
#         Windows Components -> Mgmnt and Monitoring Tools and check
#         WMI Windows Installer Provider.
#
#   NOTE: Naming conventions are important when updating packages.
#         By default, Cfengine expects "name-version-arch.msi"
#         on Windows, where name is lowercase, and arch is
#         i686 or x86_64. No spaces should be included in the filename.
#         The Caption and Version fields inside the msi package
#         are important. They must correspond to the file name as
#         follows: name = lowercase(spacetodash(Caption)),
#         version = Version. For any msi-file, use InstEd
#         (www.instedit.com) to check/modify the
#         Caption and Version fields
#         (Tables->Property->ProductName/ProductVersion).
#
#         For example, ProductName "CFEngine Nova" with ProductVersion
#         "1.1.2" for 32-bit Windows will correspond to the filename
#         "cfengine-nova-1.1.2-i686.msi".
#
#########################################################################

bundle agent check_software
{
vars:

# software to install if not installed
 "include_software" slist => {
                             "7-zip-4.50-$(sys.arch).msi"
                             };

# this software gets updated if it is installed
 "autoupdate_software" slist => {
                               "7-zip"
                               };

# software to uninstall if it is installed
 "exclude_software" slist => {
```

```
                             "7-zip-4.65-$(sys.arch).msi"
                             };

methods:
#  "any" usebundle => add_software( "@(check_software.include_software)", "$(sys.policy_hub)" );
#  "any" usebundle => update_software( "@(check_software.autoupdate_software)", "$(sys.policy_hub)"
#  "any" usebundle => remove_software( "@(check_software.exclude_software)", "$(sys.policy_hub)" );
}

#########################################################################

bundle agent add_software(pkg_name, srv)
{
vars:
# dir to install from locally - can also check multiple directories
 "local_software_dir" string => "C:\Program Files\Cfengine\software\add";

files:

  "$(local_software_dir)"
       copy_from => remote_cp("/var/cfengine/master_software_updates/$(sys.flavour)_$(sys.arch)/add"
    depth_search => recurse("1"),
         classes => if_repaired("got_newpkg"),
                   comment => "Copy software from remote repository";


packages:

# When to check if the package is installed ?
 got_newpkg|any::
  "$(pkg_name)"
    package_policy           => "add",
    package_method           => msi_implicit( "$(local_software_dir)" ),
    classes                  => if_else("add_success", "add_fail" ),
    comment                  => "Install new software, if not already present";

reports::
 add_fail::
   "Failed to install one or more packages";
}

#########################################################################

bundle agent update_software(sw_names, srv)
{
vars:
# dir to install from locally - can also check multiple directories
```

```
 "local_software_dir" string => "C:\Program Files\Cfengine\software\update";

files:

  "$(local_software_dir)"
       copy_from => remote_cp("/var/cfengine/master_software_updates/$(sys.flavour)_$(sys.arch)/upda
    depth_search => recurse("1"),
         classes => if_repaired("got_newpkg"),
                   comment => "Copy software updates from remote repository";


packages:

# When to check if the package is updated ?
 got_newpkg|any::
  "$(sw_names)"
    package_policy           => "update",
    package_select           => ">=",                 # picks the newest update available
    package_architectures    => { "$(sys.arch)" },    # install 32 or 64 bit package ?
    package_version          => "1.0",                # at least version 1.0
    package_method           => msi_explicit( "$(local_software_dir)" ),
    classes                  => if_else("update_success", "update_fail");


reports::
 update_fail::
   "Failed to update one or more packages";
}

#########################################################################

bundle agent remove_software(pkg_name, srv)
{
vars:
# dir to install from locally - can also check multiple directories
 "local_software_dir" string => "C:\Program Files\Cfengine\software\remove";

files:

  "$(local_software_dir)"
       copy_from => remote_cp("/var/cfengine/master_software_updates/$(sys.flavour)_$(sys.arch)/remo
    depth_search => recurse("1"),
         classes => if_repaired("got_newpkg"),
         comment => "Copy removable software from remote repository";

packages:
got_newpkg::
  "$(pkg_name)"
    package_policy           => "delete",
    package_method           => msi_implicit( "$(local_software_dir)" ),
    classes                  => if_else("remove_success", "remove_fail" ),
    comment                  => "Remove software, if present";

reports::
 remove_fail::
   "Failed to remove one or more packages";
}
```

3.57 Trigger classes

```
#######################################################
#
# Insert a number of lines and trigger a followup if edited
#
#######################################################

body common control

{
any::

  bundlesequence  => { "insert" };
}


#######################################################

bundle agent insert

{
vars:

  "v" string => "
                One potato
                Two potato
                Three potahto
                Four
                ";

files:

  "/tmp/test_insert"

     edit_line => Insert("$(insert.v)"),
     edit_defaults => empty,
     classes => trigger("edited");

commands:

 edited::

  "/bin/echo make bananas";
reports:

  edited::

    "The potatoes are bananas";

}

#######################################################
# For the library
#######################################################

bundle edit_line Insert(name)

{
insert_lines:

  "Begin$(const.n) $(name)$(const.n)End";

}

#######################################################

body edit_defaults empty

{
empty_file_before_editing => "true";
}

#######################################################

body classes trigger(x)

{
promise_repaired => { "$(x)" };
}
```

3.58 Unmount NFS filesystem

```
#####################################################################
# Mount NFS
#####################################################################

body common control

{
bundlesequence => { "mounts" };
}

#####################################################################

bundle agent mounts

{
storage:

  # Assumes the filesystem has been exported

  "/mnt" mount  => nfs("server.example.org","/home");
}

######################################################################

body mount nfs(server,source)

{
mount_type => "nfs";
mount_source => "$(source)";
mount_server => "$(server)";
edit_fstab => "true";
unmount => "true";
}
```

3.59 Web server modules

The problem of editing the correct modules into the list of standard modules for the Apache web server. This example is based on the standard configuration deployment of SuSE Linux. Simply provide the list of modules you want and another list that you don't want.

```
#######################################################
#
# Apache 2 reconfig - modelled on SuSE
#
#######################################################

body common control

{
any::

  bundlesequence  => {
                     apache
                     };
}

#######################################################

bundle agent apache

{
files:

 SuSE::

  "/etc/sysconfig/apache2"

     edit_line => fixapache;
}

#######################################################
# For the library
#######################################################

bundle edit_line fixapache

{
vars:

 "add_modules"     slist => {
                            "dav",
                            "dav_fs",
                            "ssl",
                            "php5",
                            "dav_svn",
                            "xyz",
                            "superduper"
                            };

 "del_modules"     slist => {
                            "php3",
                            "jk",
                            "userdir",
                            "imagemap",
                            "alias"
                            };

insert_lines:

 "APACHE_CONF_INCLUDE_FILES=\"/site/masterfiles/local-http.conf\"";

field_edits:

 #####################################################################
 # APACHE_MODULES="authz_host actions alias ..."
 #####################################################################

    # Values have the form NAME = "quoted space separated list"

   "APACHE_MODULES=.*"

      # Insert module "columns" between the quoted RHS
      # using space separators

      edit_field => quotedvar("$(add_modules)","append");

   "APACHE_MODULES=.*"

      # Delte module "columns" between the quoted RHS
      # using space separators

      edit_field => quotedvar("$(del_modules)","delete");

   # if this line already exists, edit it

}
```

3.60 Warn if matching line in file

```
########################################################
#
# Warn if line matched
#
########################################################

body common control

{
bundlesequence  => { "testbundle" };
}

########################################################

bundle agent testbundle

{
files:

  "/var/cfengine/inputs/.*"

       edit_line => DeleteLinesMatching(".*cfenvd.*"),
       action => WarnOnly;
}

########################################################

bundle edit_line DeleteLinesMatching(regex)
  {
  delete_lines:

    "$(regex)" action => WarnOnly;

  }

########################################################

body action WarnOnly
{
action_policy => "warn";
}
```

3.61 Windows registry

```
body common control
{
bundlesequence => { "reg" };
}

bundle agent reg
{
vars:

  "value" string => registryvalue("HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS\Cfengine","value3");

reports:

  windows::

   "Value extracted: $(value)";

}
```

3.62 unit_registry_cache.cf

```
body common control
{
 bundlesequence => {
#                   "registry_cache"
#                   "registry_restore"
                   };
}

#########################################

bundle agent registry_cache
{
 databases:
  windows::

     "HKEY_LOCAL_MACHINE\SOFTWARE\Adobe"
        database_operation => "cache",
        database_type      => "ms_registry",
        comment => "Save correct registry settings for Adobe products";
}

#########################################

bundle agent registry_restore
```

```
{
 databases:
  windows::

     "HKEY_LOCAL_MACHINE\SOFTWARE\Adobe"
        database_operation => "restore",
        database_type      => "ms_registry",
        comment => "Make sure Adobe products have correct registry settings";
}
```

3.63 unit_registry.cf

```
body common control
{

bundlesequence => { "databases" };
}


bundle agent databases

{
databases:

 windows::

  # Regsitry has (value,data) pairs in "keys" which are directories

#  "HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS"

#    database_operation => "create",
#    database_type     => "ms_registry";

#  "HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS\Cfengine"

#    database_operation => "create",
#    database_rows => { "value1,REG_SZ,new value 1", "value2,REG_SZ,new val 2"} ,
#    database_type     => "ms_registry";


  "HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS\Cfengine"

    database_operation => "delete",
    database_columns => { "value1", "value2" } ,
    database_type => "ms_registry";


# "HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS\Cfengine"

#    database_operation => "cache",   # cache,restore

#    registry_exclude => { ".*Windows.*CurrentVersion.*", ".*Touchpad.*", ".*Capabilities.FileAssoci

#    database_type     => "ms_registry";

"HKEY_LOCAL_MACHINE\SOFTWARE\Cfengine AS"

   database_operation => "restore",
   database_type      => "ms_registry";

}
```

CFEngine®