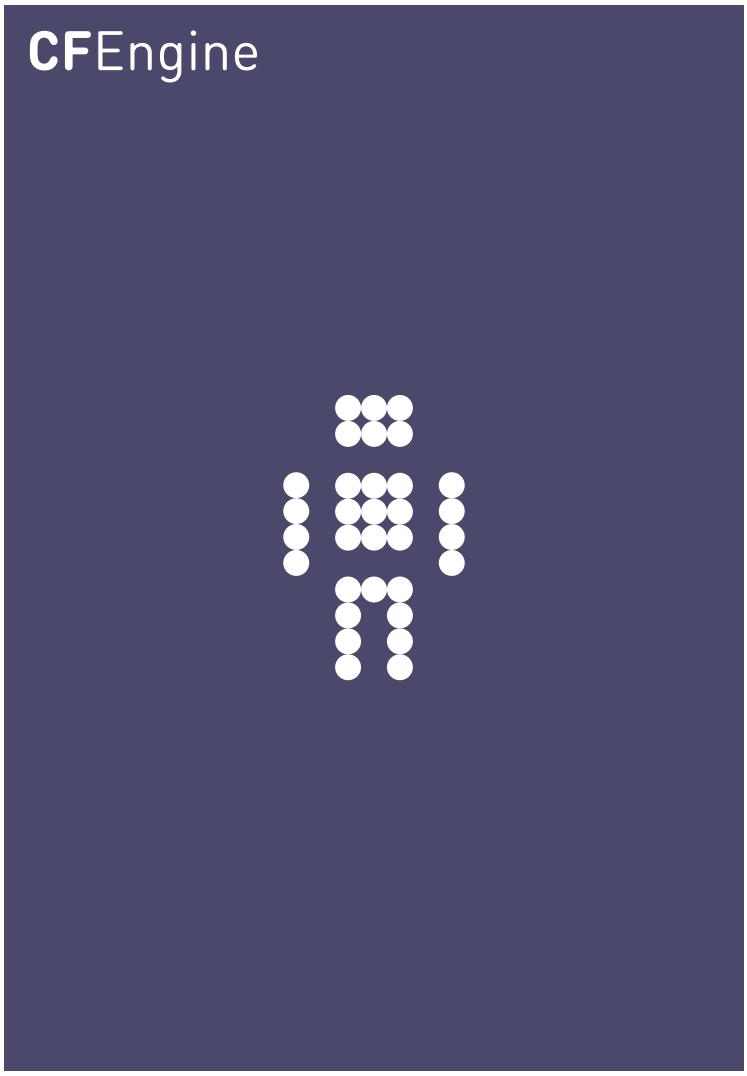

Community Open Promise Body Library

A CFEngine Standard

CFEngine AS

Table of Contents

1 The Purpose Of This Handbook

CFEngine is built on promises. Promises were chosen as the model for CFEngine's configuration language, because they represent an expression of intention.

If you are using custom scripts to manage your systems, you are using *recipes*. Take a look at any cookbook and you will see that all recipes look the same: take flour, eggs, butter, sugar ... and you know nothing because you can make a hundred things from these steps. If you don't make clear your intention, it is very hard to know what the recipe is supposed to be: is it a cake, a waffle, a pastry?

The same is true in system administration. Recipes are not merely scripts, they encapsulate knowledge and experience. Their value is in communicating *desired outcomes* or states.

This library of standard components is like a cookbook that tells you only how to make basics well. It gives these basic skills names and therefore gives you a common vocabulary – you will put together these basics in creative ways to build your systems.

Please contribute to this guide by helping to develop a repertoire of basic skills and names. This collection should be comprehensive but parsimonious. Basics are only basics if they are few and carefully thought out. This is a work in progress and your experience is welcome.

This library will be moderated by CFEngine, and contributions and discussions ca n be made to the help-cfengine@cfengine.org mailing list.

1.1 body acl access_generic(acl)

```
body acl access_generic(acl)
# default/inherited ACLs are left unchanged,
# applicable for both files and directories on all platforms
{
acl_method => "overwrite";
aces => { "@(acl)" };

windows::
acl_type => "ntfs";

!windows::
acl_type => "posix";
}
```

1.2 body acl ntfs(acl)

```
body acl ntfs(acl)
{
acl_type => "ntfs";
acl_method => "overwrite";
```

```
aces => { "@(acl)" };
}
```

1.3 body acl strict

```
body acl strict
# NOTE: May need to take ownership of file/dir
# to be sure no-one else is allowed access
{
acl_method => "overwrite";

windows::
aces => { "user:Administrator:rwx" };
!windows::
aces => { "user:root:rwx" };
}
```

1.4 body action bg(elapsed,expire)

```
body action bg(elapsed,expire)
{
ifelapsed   => "$(elapsed)";
expireafter => "$(expire)";
background  => "true";
}
```

1.5 body action if_elapsed(x)

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

1.6 body action if_elapsed_day

```
body action if_elapsed_day
{
ifelapsed => "1440";    # 60 x 24
expireafter => "1400";
}
```

1.7 body action ifwin_bg

```
body action ifwin_bg
{
windows::
background => "true";
}
```

1.8 body action immediate

```
body action immediate
{
ifelapsed => "0";
}
```

1.9 body action log_repaired(log,message)

```
body action log_repaired(log,message)
{
log_string => "$(sys.date), $(message)";
log_repaired => "$(log)";
}
```

1.10 body action log_verbose

```
body action log_verbose
{
log_level => "verbose";
}
```

1.11 body action measure_performance(x)

```
body action measure_performance(x)
{
measurement_class => "Detect changes in $(this.promiser)";
ifelapsed => "$(x)";
expireafter => "$(x)";
}
```

1.12 body action policy(p)

```
body action policy(p)
{
action_policy => "$(p)";
}


# Log a message to log=[/file|stdout]
```

1.13 body action sample_rate(x)

```
body action sample_rate(x)
{
ifelapsed => "$(x)";
expireafter => "10";
}
```

1.14 body action warn_only

```
body action warn_only
{
action_policy => "warn";
ifelapsed => "60";
}
```

1.15 body changes detect_all_change

```
body changes detect_all_change

# This is fierce, and will cost disk cycles

{
hash           => "best";
report_changes => "all";
update_hashes  => "yes";
}
```

1.16 body changes detect_content

```
body changes detect_content

# This is a cheaper alternative

{
hash           => "md5";
report_changes => "content";
update_hashes  => "yes";
}
```

1.17 body changes diff

```
body changes diff
# Generates diff report (Nova and above)
{
hash           => "sha256";
report_changes => "content";
report_diffs   => "true";
update_hashes  => "yes";
}
```

1.18 body changes diff_noupdate

```
body changes diff_noupdate
{
hash           => "sha256";
report_changes => "content";
report_diffs   => "true";
update_hashes  => "no";
}
```

1.19 body changes noupdate

```
body changes noupdate
# Use on (small) files that should never change
{
hash           => "sha256";
report_changes => "content";
update_hashes  => "no";
}
```

1.20 body classes always(x)

```
body classes always(x)

# Define a class no matter what the outcome of the promise is

{
  promise_repaired => { "$(x)" };
  promise_kept => { "$(x)" };
  repair_failed => { "$(x)" };
  repair_denied => { "$(x)" };
  repair_timeout => { "$(x)" };
}

# agent bundles
```

1.21 body classes cf2_if_else(yes,no)

```
body classes cf2_if_else(yes,no)

# meant to match cf2 semantics

{
promise_repaired => { "$(yes)" };
repair_failed    => { "$(no)" };
repair_denied    => { "$(no)" };
repair_timeout   => { "$(no)" };
}
```

1.22 body classes cmd_repair(code,cl)

```
body classes cmd_repair(code,cl)
{
repaired_returncodes => { "$(code)" };
promise_repaired => { "$(cl)" };
}
```

1.23 body classes enumerate(x)

```
body classes enumerate(x)

#
# This is used by commercial editions to count
# instances of jobs in a cluster
#

{
promise_repaired => { "mXC_$(x)" };
promise_kept => { "mXC_$(x)" };
persist_time => "15";
}
```

1.24 body classes if_else(yes,no)

```
body classes if_else(yes,no)

{
promise_kept     => { "$(yes)" };
promise_repaired => { "$(yes)" };
repair_failed    => { "$(no)" };
repair_denied    => { "$(no)" };
repair_timeout   => { "$(no)" };
}
```

1.25 body classes if_notkept(x)

```
body classes if_notkept(x)
{
repair_failed   => { "$(x)" };
repair_denied   => { "$(x)" };
repair_timeout  => { "$(x)" };
}
```

1.26 body classes if_ok(x)

```
body classes if_ok(x)
{
promise_repaired => { "$(x)" };
promise_kept => { "$(x)" };
}
```

1.27 body classes if_ok_cancel(x)

```
body classes if_ok_cancel(x)
{
cancel_repaired => { "$(x)" };
cancel_kept => { "$(x)" };
}
```

1.28 body classes if_repaired(x)

```
body classes if_repaired(x)
{
promise_repaired => { "$(x)" };
}
```

1.29 body classes state_repaired(x)

```
body classes state_repaired(x)
{
promise_repaired => { "$(x)" };
persist_time => "10";
}
```

1.30 body contain in_dir(s)

```
body contain in_dir(s)
{
chdir => "$(s)";
}
```

1.31 body contain in_dir_shell(s)

```
body contain in_dir_shell(s)
{
chdir => "$(s)";
useshell => "true";
}
```

1.32 body contain in_dir_shell_and_silent(dir)

```
body contain in_dir_shell_and_silent(dir)
{
useshell => "true";
no_output => "true";
chdir => "$(dir)";
}
```

1.33 body contain in_shell

```
body contain in_shell
{
useshell => "true";
}
```

1.34 body contain in_shell_and_silent

```
body contain in_shell_and_silent
{
useshell => "true";
no_output => "true";
}
```

1.35 body contain in_shell_bg

```
body contain in_shell_bg
{
useshell => "true";
background => "true";
}
```

1.36 body contain jail(owner,root,dir)

```
body contain jail(owner,root,dir)
{
exec_owner => "$(owner)";
useshell => "true";
chdir => "$(dir)";
```

```
chroot => "$(root)";
}
```

1.37 body contain setuid(x)

```
body contain setuid(x)
{
exec_owner => "$(x)";
useshell => "false";
}
```

1.38 body contain setuid_sh(x)

```
body contain setuid_sh(x)
{
exec_owner => "$(x)";
useshell => "true";
}
```

1.39 body contain setuidgid_sh(owner,group)

```
body contain setuidgid_sh(owner,group)
{
exec_owner => "$(owner)";
exec_group => "$(group)";
useshell => "true";
}
```

1.40 body contain silent

```
body contain silent
{
no_output => "true";
}
```

1.41 body contain silent_in_dir(s)

```
body contain silent_in_dir(s)
{
chdir => "$(s)";
no_output => "true";
}
```

1.42 body copy_from backup_local_cp(from)

```
body copy_from backup_local_cp(from)
# Local copy, keeping a backup of old versions
{
  source      => "$(from)";
  copy_backup => "timestamp";
}


# Copy only if the file does not already exist, i.e. seed the placement
```

1.43 body copy_from local_cp(from)

```
body copy_from local_cp(from)
{
source      => "$(from)";
}
```

1.44 body copy_from local_dcp(from)

```
body copy_from local_dcp(from)
{
source      => "$(from)";
compare     => "digest";
}
```

1.45 body copy_from no_backup_cp(from)

```
body copy_from no_backup_cp(from)
{
source      => "$(from)";
copy_backup => "false";
}
```

1.46 body copy_from no_backup_dcp(from)

```
body copy_from no_backup_dcp(from)
{
source      => "$(from)";
copy_backup => "false";
compare     => "digest";
}
```

1.47 body copy_from no_backup_rcp(from,server)

```
body copy_from no_backup_rcp(from,server)
{
servers     => { "$(server)" };
source      => "$(from)";
compare     => "mtime";
copy_backup => "false";
}
```

1.48 body copy_from perms_cp(from)

```
body copy_from perms_cp(from)
{
source      => "$(from)";
preserve    => "true";
}
```

1.49 body copy_from remote_cp(from,server)

```
body copy_from remote_cp(from,server)
{
servers     => { "$(server)" };
source      => "$(from)";
compare     => "mtime";
}
```

1.50 body copy_from remote_dcp(from,server)

```
body copy_from remote_dcp(from,server)
```

```
{
servers     => { "$(server)" };
source      => "$(from)";
compare     => "digest";
}
```

1.51 body copy_from secure_cp(from,server)

```
body copy_from secure_cp(from,server)
{
source      => "$(from)";
servers     => { "$(server)" };
compare     => "digest";
encrypt     => "true";
verify      => "true";
}
```

1.52 body copy_from seed_cp(from)

```
body copy_from seed_cp(from)
{
source      => "$(from)";
compare     => "exists";
}
```

1.53 body copy_from sync_cp(from,server)

```
body copy_from sync_cp(from,server)
{
servers     => { "$(server)" };
source      => "$(from)";
purge       => "true";
preserve    => "true";
type_check  => "false";
}
```

1.54 body database_server local_mysql(username, password)

```
body database_server local_mysql(username, password)
{
db_server_owner => "$(username)";
```

```
db_server_password => "$(password)";
db_server_host => "localhost";
db_server_type => "mysql";
db_server_connection_db => "mysql";
}
```

1.55 body database_server local_postgresql(username, password)

```
body database_server local_postgresql(username, password)
{
db_server_owner => "$(username)";
db_server_password => "$(password)";
db_server_host => "localhost";
db_server_type => "postgres";
db_server_connection_db => "postgres";
}
```

1.56 body delete tidy

```
body delete tidy

{
dirlinks => "delete";
rmdirs   => "true";
}
```

1.57 body depth_search include_base

```
body depth_search include_base
{
include_basedir => "true";
}
```

1.58 body depth_search recurse(d)

```
body depth_search recurse(d)

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

1.59 body depth_search recurse_ignore(d,list)

```
body depth_search recurse_ignore(d,list)
{
depth => "$(d)";
exclude_dirs => { @(list) };
}
```

1.60 body edit_defaults backup_timestamp

```
body edit_defaults backup_timestamp
{
empty_file_before_editing => "false";
edit_backup => "timestamp";
#max_file_size => "300000";
}
```

1.61 body edit_defaults empty

```
body edit_defaults empty
{
empty_file_before_editing => "true";
edit_backup => "false";
#max_file_size => "300000";
}
```

1.62 body edit_defaults no_backup

```
body edit_defaults no_backup
{
edit_backup => "false";
}
```

1.63 body edit_defaults std_defs

```
body edit_defaults std_defs
{
empty_file_before_editing => "false";
```

```
edit_backup => "false";
#max_file_size => "300000";
}
```

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

```
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";
allow_blank_fields => "true";
}
```

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

```
body edit_field line(split,col,newval,method)
{
field_separator    => "$(split)";
select_field       => "$(col)";
value_separator    => " ";
field_value        => "$(newval)";
field_operation    => "$(method)";
extend_fields      => "true";
allow_blank_fields => "true";
}
```

1.66 body edit_field quoted_var(newval,method)

```
body edit_field quoted_var(newval,method)
{
field_separator => "\"";
select_field    => "2";
value_separator  => " ";
field_value     => "$(newval)";
field_operation => "$(method)";
extend_fields => "false";
allow_blank_fields => "true";
}
```

1.67 body environment_resources kvm(name, arch, cpu_count, mem_kb, disk_file)

```
body environment_resources kvm(name, arch, cpu_count, mem_kb, disk_file)
{
env_spec =>
"<domain type='kvm'>
  <name>$(name)</name>
  <memory>$(mem_kb)</memory>
  <currentMemory>$(mem_kb)</currentMemory>
  <vcpu>$(cpu_count)</vcpu>
  <os>
    <type arch='$(arch)'>hvm</type>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
      <source file='$(disk_file)'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <interface type='network'>
      <source network='default'/>
    </interface>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes'/>
  </devices>
</domain>";
}
```

1.68 body file_select all

```
body file_select all
{
leaf_name => { ".*" };
file_result => "leaf_name";
```

```
}
```

1.69 body file_select by_name(names)

```
body file_select by_name(names)
{
leaf_name  => { @(names)};
file_result => "leaf_name";
}
```

1.70 body file_select days_old(days)

```
body file_select days_old(days)
{
mtime       => irange(0,ago(0,0,"$(days)",0,0,0));
file_result => "mtime";
}
```

1.71 body file_select dirs

```
body file_select dirs
{
file_types  => { "dir" };
file_result => "file_types";
}
```

1.72 body file_select ex_list(names)

```
body file_select ex_list(names)
{
leaf_name  => { @(names)};
file_result => "!leaf_name";
}
```

1.73 body file_select exclude(name)

```
body file_select exclude(name)
{
leaf_name  => { "$(name)"};
```

```
file_result => "!leaf_name";
}
```

1.74 body file_select name_age(name,days)

```
body file_select name_age(name,days)
{
leaf_name   => { "$(name)" };
mtime       => irange(0,ago(0,0,"$(days)",0,0,0));
file_result => "mtime.leaf_name";
}
```

1.75 body file_select plain

```
body file_select plain
{
file_types  => { "plain" };
file_result => "file_types";
}
```

1.76 body file_select size_range(from,to)

```
body file_select size_range(from,to)
{
search_size => irange("$(from)","$(to)");
file_result => "size";
}
```

1.77 body link_from linkchildren(tofile)

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

1.78 body link_from ln_s(x)

```
body link_from ln_s(x)
{
link_type => "symlink";
source => "$(x)";
when_no_source => "force";
}
```

1.79 body location after(str)

```
body location after(str)
{
before_after => "after";
select_line_matching => "$(str)";
}
```

1.80 body location before(str)

```
body location before(str)
{
before_after => "before";
select_line_matching => "$(str)";
}
```

1.81 body location start

```
body location start
{
before_after => "before";
}
```

1.82 body match_value line_match_value(line_match, extract_regex)

```
body match_value line_match_value(line_match, extract_regex)
{
select_line_matching => "$(line_match)";
extraction_regex => "$(extract_regex)";
}
```

1.83 body match_value scan_changing_file(line)

```
body match_value scan_changing_file(line)
{
select_line_matching => "$(line)";
track_growing_file => "false";
}
```

1.84 body match_value scan_log(line)

```
body match_value scan_log(line)
{
select_line_matching => "$(line)";
track_growing_file => "true";
}
```

1.85 body match_value single_value(regex)

```
body match_value single_value(regex)
{
select_line_matching => "$(regex)";
extraction_regex => "($(regex))";
}
```

1.86 body mount nfs(server,source)

```
body mount nfs(server,source)
{
mount_type => "nfs";
mount_source => "$(source)";
mount_server => "$(server)";
edit_fstab => "true";
}
```

1.87 body mount nfs_p(server,source,perm)

```
body mount nfs_p(server,source,perm)
```

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

1.88 body mount unmount

```
body mount unmount
{
mount_type => "nfs";
edit_fstab => "true";
unmount => "true";
}
```

1.89 body package_method alpinelinux

```
body package_method alpinelinux
{
 package_changes => "bulk";
 package_list_command => "/sbin/apk info -v";
 package_list_name_regex    => "([^\s]+)-.*";
 package_list_version_regex => "[^\s]+-([^\s]+).*";
 package_name_regex    => ".*";
 package_installed_regex => ".*";
 package_name_convention => "$(name)";
 package_add_command => "/sbin/apk add";
 package_delete_command => "/sbin/apk del";
}
```

1.90 body package_method apt

```
body package_method apt
{
package_changes => "bulk";
package_list_command => "/usr/bin/dpkg -l";
package_list_name_regex    => ".i\s+([^\s]+).*";
package_list_version_regex => ".i\s+[^\s]+\s+([^\s]+).*";
package_installed_regex => ".i.*"; # packages that have been uninstalled may be listed
package_name_convention => "$(name)";
```

```
# set it to "0" to avoid caching of list during upgrade
package_list_update_ifelapsed => "240";

have_aptitude::
   package_add_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude -o
   package_list_update_command => "/usr/bin/aptitude update";
   package_delete_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude
   package_update_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitud
   package_patch_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude
   package_verify_command =>  "/usr/bin/aptitude show";
   package_noverify_regex => "(State: not installed|E: Unable to locate package .*)";

   package_patch_list_command => "/usr/bin/aptitude --assume-yes --simulate --verbose full-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";

!have_aptitude::
   package_add_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get -o
   package_list_update_command => "/usr/bin/apt-get update";
   package_delete_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_update_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_patch_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_verify_command => "/usr/bin/dpkg -s";
   package_noverify_returncode => "1";

   package_patch_list_command => "/usr/bin/apt-get --just-print dist-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";

}
```

1.91 body package_method dpkg_version(repo)

```
body package_method dpkg_version(repo)
{
package_changes => "individual";
package_list_command => "/usr/bin/dpkg -l";

# set it to "0" to avoid caching of list during upgrade
package_list_update_command => "/usr/bin/apt-get update";
package_list_update_ifelapsed => "240";

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

```
package_installed_regex => ".i.*"; # packages that have been uninstalled may be listed

package_file_repositories => { "$(repo)" };

debian.x86_64::
   package_name_convention => "$(name)_$(version)_amd64.deb";

debian.i686::
   package_name_convention => "$(name)_$(version)_i386.deb";

have_aptitude::
   package_patch_list_command => "/usr/bin/aptitude --assume-yes --simulate --verbose full-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";
!have_aptitude::
   package_patch_list_command => "/usr/bin/apt-get --just-print dist-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";

debian::
   package_add_command => "/usr/bin/dpkg --install";
   package_delete_command => "/usr/bin/dpkg --purge";
   package_update_command =>  "/usr/bin/dpkg --install";
   package_patch_command =>  "/usr/bin/dpkg --install";
}
```

1.92 body package_method emerge

```
body package_method emerge
{
 package_changes => "individual";
 package_list_command => "/bin/sh -c '/bin/ls -d /var/db/pkg/*/* | cut -c 13-'";
 package_list_name_regex => ".*/([^\s]+)-\d.*";
 package_list_version_regex => ".*/[^\s]+-(\d.*)";
 package_installed_regex => ".*";                          # all reported are installed
 package_name_convention => "$(name)";
 package_list_update_command => "/bin/true";               # I prefer manual syncing
 #package_list_update_command => "/usr/bin/emerge --sync"; # if you like automatic
 package_list_update_ifelapsed => "240";                   # should happen every 4 hours

 package_add_command => "/usr/bin/emerge -q --quiet-build";
 package_delete_command => "/usr/bin/emerge --depclean";
 package_update_command => "/usr/bin/emerge --update";
 package_patch_command => "/usr/bin/emerge --update";
```

```
 package_verify_command => "/usr/bin/emerge -s";
 package_noverify_regex => ".*(Not Installed|Applications found : 0).*";
}
```

1.93 body package_method freebsd

```
body package_method freebsd
{
 package_changes => "individual";

 # Could use rpm for this
 package_list_command => "/usr/sbin/pkg_info";

 # Remember to escape special characters like |

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

 package_name_regex    => "([^\s]+)-.*";
 package_version_regex => "[^\s]+-([^\s]+).*";

 package_installed_regex => ".*";

 package_name_convention => "$(name)-$(version)";


 package_add_command => "/usr/sbin/pkg_add -r";
 package_delete_command => "/usr/sbin/pkg_delete";
}
```

1.94 body package_method generic

```
body package_method generic
{
SuSE::
 package_changes => "bulk";
 package_list_command => "/bin/rpm -qa --queryformat \"i | repos | %{name} | %{version}-%{release} |
 # set it to "0" to avoid caching of list during upgrade
 package_list_update_command => "/usr/bin/zypper list-updates";
 package_list_update_ifelapsed => "0";
 package_patch_list_command => "/usr/bin/zypper patches";
 package_installed_regex => "i.*";
 package_list_name_regex    => "[^|]+\|[^|]+\|\s+([^\s]+).*";
 package_list_version_regex => "[^|]+\|[^|]+\|[^|]+\|\s+([^\s]+).*";
```

```
 package_list_arch_regex    => "[^|]+\|[^|]+\|[^|]+\|[^|]+\|\s+([^\s]+).*";
 package_patch_installed_regex => ".*Installed.*|.*Not Applicable.*";
 package_patch_name_regex    => "[^|]+\|\s+([^\s]+).*";
 package_patch_version_regex => "[^|]+\|[^|]+\|\s+([^\s]+).*";
 package_name_convention => "$(name)";
 package_add_command => "/usr/bin/zypper --non-interactive install";
 package_delete_command => "/usr/bin/zypper --non-interactive remove --force-resolution";
 package_update_command => "/usr/bin/zypper --non-interactive update";
 package_patch_command => "/usr/bin/zypper --non-interactive patch$"; # $ means no args
 package_verify_command => "/usr/bin/zypper --non-interactive verify$";

redhat::
 package_changes => "bulk";
 package_list_command => "/bin/rpm -qa --qf '%{name} %{version}-%{release} %{arch}\n'";
 package_patch_list_command => "/usr/bin/yum --quiet check-update";
 package_list_name_regex    => "^(\S+?)\s\S+?\s\S+$";
 package_list_version_regex => "^\S+?\s(\S+?)\s\S+$";
 package_list_arch_regex    => "^\S+?\s\S+?\s(\S+)$";
 package_installed_regex => ".*";
 package_name_convention => "$(name)";
 package_list_update_command => "/usr/bin/yum --quiet check-update";
 package_list_update_ifelapsed => "0";     # sometimes, caching is pretty disturbing
 package_patch_installed_regex => "^\s.*";
 package_patch_name_regex    => "([^.]+).*";
 package_patch_version_regex => "[^\s]\s+([^\s]+).*";
 package_patch_arch_regex    => "[^.]+\.([^\s]+).*";
 package_add_command    => "/usr/bin/yum -y install";
 package_update_command => "/usr/bin/yum -y update";
 package_patch_command => "/usr/bin/yum -y update";
 package_delete_command => "/bin/rpm -e --nodeps --allmatches";
 package_verify_command => "/bin/rpm -V";

# package_changes => "bulk";
# package_list_command => "/usr/bin/yum list installed";
# package_patch_list_command => "/usr/bin/yum check-update";
# package_list_name_regex    => "([^.]+).*";
# package_list_version_regex => "[^\s]\s+([^\s]+).*";
# package_list_arch_regex    => "[^.]+\.([^\s]+).*";
# package_installed_regex => ".*(installed|\s+@).*";
# package_name_convention => "$(name).$(arch)";
# package_list_update_ifelapsed => "240";
# package_patch_installed_regex => "^\s.*";
# package_patch_name_regex    => "([^.]+).*";
# package_patch_version_regex => "[^\s]\s+([^\s]+).*";
# package_patch_arch_regex    => "[^.]+\.([^\s]+).*";
# package_add_command => "/usr/bin/yum -y install";
# package_delete_command => "/bin/rpm -e --nodeps";
```

```
# package_verify_command => "/bin/rpm -V";

debian::
 package_changes => "bulk";
 package_list_command => "/usr/bin/dpkg -l";
 package_list_name_regex    => ".i\s+([^\s]+).*";
 package_list_version_regex => ".i\s+[^\s]+\s+([^\s]+).*";
 package_installed_regex => ".i.*"; # packages that have been uninstalled may be listed
 package_name_convention => "$(name)";
 package_list_update_ifelapsed => "240";            # 4 hours

debian.have_aptitude::
   package_add_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude -o
   package_list_update_command => "/usr/bin/aptitude update";
   package_delete_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude
   package_update_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitud
   package_patch_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/aptitude
   package_verify_command =>  "/usr/bin/aptitude show";
   package_noverify_regex => "(State: not installed|E: Unable to locate package .*)";

   package_patch_list_command => "/usr/bin/aptitude --assume-yes --simulate --verbose full-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";

debian.!have_aptitude::
   package_add_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get -o
   package_list_update_command => "/usr/bin/apt-get update";
   package_delete_command => "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_update_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_patch_command =>  "/usr/bin/env DEBIAN_FRONTEND=noninteractive LC_ALL=C /usr/bin/apt-get
   package_verify_command => "/usr/bin/dpkg -s";
   package_noverify_returncode => "1";

   package_patch_list_command => "/usr/bin/apt-get --just-print dist-upgrade";
   package_patch_name_regex => "^Inst\s+(\S+)\s+.*";
   package_patch_version_regex => "^Inst\s+\S+\s+\[?\(?([^\],\s]+).*";

freebsd::
 package_changes => "individual";
 package_list_command => "/usr/sbin/pkg_info";
 package_list_name_regex    => "([^\s]+)-.*";
 package_list_version_regex => "[^\s]+-([^\s]+).*";
 package_name_regex    => "([^\s]+)-.*";
 package_version_regex => "[^\s]+-([^\s]+).*";
 package_installed_regex => ".*";
 package_name_convention => "$(name)-$(version)";
 package_add_command => "/usr/sbin/pkg_add -r";
```

```
 package_delete_command => "/usr/sbin/pkg_delete";

alpinelinux::
 package_changes => "bulk";
 package_list_command => "/sbin/apk info -v";
 package_list_name_regex    => "([^\s]+)-.*";
 package_list_version_regex => "[^\s]+-([^\s]+).*";
 package_name_regex    => ".*";
 package_installed_regex => ".*";
 package_name_convention => "$(name)";
 package_add_command => "/sbin/apk add";
 package_delete_command => "/sbin/apk del";

gentoo::
 package_changes => "individual";
 package_list_command => "/bin/sh -c '/bin/ls -d /var/db/pkg/*/* | cut -c 13-'";
 package_list_name_regex => ".*/([^\s]+)-\d.*";
 package_list_version_regex => ".*/[^\s]+-(\d.*)";
 package_installed_regex => ".*";                          # all reported are installed
 package_name_convention => "$(name)";
 package_list_update_command => "/bin/true";               # I prefer manual syncing
 #package_list_update_command => "/usr/bin/emerge --sync"; # if you like automatic
 package_list_update_ifelapsed => "240";                   # should happen every 4 hours

 package_add_command => "/usr/bin/emerge -q --quiet-build";
 package_delete_command => "/usr/bin/emerge --depclean";
 package_update_command => "/usr/bin/emerge --update";
 package_patch_command => "/usr/bin/emerge --update";
 package_verify_command => "/usr/bin/emerge -s";
 package_noverify_regex => ".*(Not Installed|Applications found : 0).*";

archlinux::
 package_changes => "bulk";
 package_list_command => "/usr/bin/pacman -Q";
 package_list_name_regex    => "(.*)\s+.*";
 package_list_version_regex => ".*\s+(.*)";
 package_installed_regex => ".*";
 package_name_convention => "$(name)";
 package_list_update_ifelapsed => "240";
 package_add_command => "/usr/bin/pacman -S --noconfirm --noprogressbar --needed";
 package_delete_command => "/usr/bin/pacman -Rs --noconfirm";
 package_update_command => "/usr/bin/pacman -S --noconfirm --noprogressbar --needed";
}
```

1.95 body package_method ips

```
body package_method ips
{
  package_changes => "bulk";
  package_list_command => "/usr/bin/pkg list -v --no-refresh";
  package_list_name_regex    => "pkg://.+?/([^\s]+)@.*$";
  package_list_version_regex => "[^\s]+@([^\s]+).*";
  package_installed_regex => ".*(i..)"; # all reported are installed

  # set it to "0" to avoid caching of list during upgrade
  package_list_update_command => "/bin/true";  # Fixme later
  package_list_update_ifelapsed => "240";

  package_add_command => "/usr/bin/pkg install --accept ";
  package_list_update_command => "/usr/bin/pkg refresh --full";
  package_delete_command => "/usr/bin/pkg uninstall";
  package_update_command =>  "/usr/bin/pkg install --accept";
  package_patch_command =>  "/usr/bin/pkg install --accept";
  package_verify_command =>  "/usr/bin/pkg list -a -v --no-refresh";
  package_noverify_regex => "(.*---|pkg list: no packages matching .* installed)";
}


# SmartOS (solaris 10 fork by Joyent) uses pkgin
```

1.96 body package_method msi_explicit(repo)

```
body package_method msi_explicit(repo)
# use software name as promiser, e.g. "7-Zip", and explicitly
# specify any package_version and package_arch
{
package_changes => "individual";
package_file_repositories => { "$(repo)" };

package_installed_regex => ".*";

package_name_convention => "$(name)-$(version)-$(arch).msi";
package_delete_convention => "$(firstrepo)$(name)-$(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";
}
```

1.97 body package_method msi_implicit(repo)

```
body package_method msi_implicit(repo)
# Use whole file name as promiser, e.g. "7-Zip-4.50-x86_64.msi",
# the name, version and arch is then deduced from the promiser
{
package_changes => "individual";
package_file_repositories => { "$(repo)" };

package_installed_regex => ".*";

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

package_name_regex => "^(\S+)-(\d+\.?)+";
package_version_regex => "^\S+-((\d+\.?)+)";
package_arch_regex => "^\S+-[\d\.]+-(.*).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";
}
```

1.98 body package_method pacman

```
body package_method pacman

{
package_changes => "bulk";

package_list_command => "/usr/bin/pacman -Q";

# set it to "0" to avoid caching of list during upgrade
package_list_update_ifelapsed => "240";

package_list_name_regex    => "(.*)\s+.*";
package_list_version_regex => ".*\s+(.*)";
package_installed_regex => ".*";

package_name_convention => "$(name)";
package_add_command => "/usr/bin/pacman -S --noconfirm --noprogressbar --needed";
package_delete_command => "/usr/bin/pacman -Rs --noconfirm";
package_update_command => "/usr/bin/pacman -S --noconfirm --noprogressbar --needed";
}
```

```
 # Single bundle for all the similar managers simplifies promises
```

1.99 body package_method rpm_filebased(path)

```
body package_method rpm_filebased(path)

# Contributed by Aleksey Tsalolikhin. Written on 29-Feb-2012.
# Based on yum_rpm body in COPBL by Trond Hasle Amundsen.
# Purpose: install packages from local filesystem-based package repository.
# Note: Specify the path to the local package repository in the argument.

# Example of how to use it:
#
# {{{
# packages:
# "epel-release"
# package_policy => "add",
# package_version => "5-4",
# package_architectures => { "noarch" },
# package_method => rpm_filebased("/repo/RPMs");
# }}}

{
  package_file_repositories => { "$(path)" };
  # the above is an addition to Trond's yum_rpm body

  package_add_command => "/bin/rpm -ihv ";
  # The above is a change from Trond's yum_rpm body, this makes the commands rpm only.
  # The reason I changed the install command from yum to rpm is yum will be default
  # refuse to install the epel-release RPM as it does not have the EPEL GPG key,
  # but rpm goes ahead and installs the epel-release RPM and the EPEL GPG key.

  package_name_convention => "$(name)-$(version).$(arch).rpm";
  # The above is a change from Tron's yum_rpm body. When package_file_repositories is in play,
  # package_name_convention has to match the file name, not the package name, per the
  # CFEngine 3 Reference Manual

  # set it to "0" to avoid caching of list during upgrade
  package_list_update_command => "/usr/bin/yum --quiet check-update";
  package_list_update_ifelapsed => "240";

  # The rest is unchanged from Trond's yum_rpm body
  package_changes => "bulk";
  package_list_command => "/bin/rpm -qa --qf '%{name} %{version}-%{release} %{arch}\n'";

  package_list_name_regex => "^(\S+?)\s\S+?\s\S+$";
```

```
  package_list_version_regex => "^\S+?\s(\S+?)\s\S+$";
  package_list_arch_regex => "^\S+?\s\S+?\s(\S+)$";

  package_installed_regex => ".*";


  package_delete_command => "/bin/rpm -e --allmatches";
  package_verify_command => "/bin/rpm -V";
}


# OpenSolaris based systems (Solaris 11, Illumos, etc) use the much better
# Image Package System.
```

1.100 body package_method rpm_version(repo)

```
body package_method rpm_version(repo)
{
package_changes => "individual";

package_list_command => "/bin/rpm -qa --queryformat \"i | repos | %{name} | %{version}-%{release} |

# set it to "0" to avoid caching of list during upgrade
package_list_update_command => "/usr/bin/yum --quiet check-update";
package_list_update_ifelapsed => "240";

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

package_installed_regex => "i.*";

package_file_repositories => { "$(repo)" };

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

package_add_command => "/bin/rpm -ivh ";
package_update_command => "/bin/rpm -Uvh ";
package_patch_command => "/bin/rpm -Uvh ";
package_delete_command => "/bin/rpm -e --nodeps";
package_verify_command => "/bin/rpm -V";
package_noverify_regex => ".*[^\s].*";
}
```

1.101 body package_method smartos

```
body package_method smartos
{
  package_changes => "bulk";
  package_list_command => "/opt/local/bin/pkgin list";
  package_list_name_regex    => "(.*)\-[0-9]+.*";
  package_list_version_regex => ".*\-([0-9][^\s]+).*";

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

  package_list_update_command => "/opt/local/bin/pkgin -y update";
  package_list_update_ifelapsed => "240";

  package_add_command => "/opt/local/bin/pkgin -y install";

  package_delete_command => "/opt/local/bin/pkgin -y remove";
  package_update_command =>  "/opt/local/bin/pkgin upgrade";
}


# The older solaris package system is poorly designed, with too many different
# names to track. See the example in tests/units/unit_package_solaris.cf
# to see how to use this
```

1.102 body package_method solaris (pkgname, spoolfile, adminfile)

```
body package_method solaris (pkgname, spoolfile, adminfile)
{
package_changes => "individual";
package_list_command => "/usr/bin/pkginfo -l";
package_multiline_start    =>  "\s*PKGINST:\s+[^\s]+.*";
package_list_name_regex    => "\s*PKGINST:\s+([^\s]+).*";
package_list_version_regex => "\s*VERSION:\s+([^\s]+).*";
package_list_arch_regex    => "\s*ARCH:\s+([^\s]+)";
package_installed_regex => "\s*STATUS:\s*(completely|partially)\s+installed.*";
package_name_convention => "$(name)";
package_add_command => "/usr/sbin/pkgadd -n -a /tmp/$(adminfile) -d /tmp/$(spoolfile)";
package_delete_command => "/usr/sbin/pkgrm -n -a /tmp/$(adminfile)";
}


#
# The following bundle is part of a package setup for solaris, see unit examples
#
```

1.103 body package_method windows_feature

```
body package_method windows_feature
{
package_changes => "individual";

package_name_convention   => "$(name)";
package_delete_convention => "$(name)";

package_installed_regex => ".*";
package_list_name_regex => "(.*)";
package_list_version_regex => "(.*)";  # FIXME: the listing does not give version, so takes name for

package_add_command    => "$(sys.winsysdir)\\WindowsPowerShell\\v1.0\\powershell.exe -Command \"Impo
package_delete_command => "$(sys.winsysdir)\\WindowsPowerShell\\v1.0\\powershell.exe -Command \"Impo
package_list_command   => "$(sys.winsysdir)\\WindowsPowerShell\\v1.0\\powershell.exe -Command \"Impo
}
```

1.104 body package_method yum

```
body package_method yum
{
package_changes => "bulk";
package_list_command => "/usr/bin/yum --quiet list installed";
package_patch_list_command => "/usr/bin/yum --quiet check-update";

# Remember to escape special characters like |

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

package_installed_regex => ".*(installed|\s+@).*";
package_name_convention => "$(name)-$(version).$(arch)";

# set it to "0" to avoid caching of list during upgrade
package_list_update_command => "/usr/bin/yum --quiet check-update";
package_list_update_ifelapsed => "240";

package_patch_installed_regex => "^\s.*";
package_patch_name_regex    => "([^.]+).*";
package_patch_version_regex => "[^\s]\s+([^\s]+).*";
package_patch_arch_regex    => "[^.]+\.([^\s]+).*";

package_add_command => "/usr/bin/yum -y install";
package_update_command => "/usr/bin/yum -y update";
```

```
package_patch_command => "/usr/bin/yum -y update";
package_delete_command => "/bin/rpm -e --nodeps";
package_verify_command => "/bin/rpm -V";
}
```

1.105 body package_method yum_rpm

```
body package_method yum_rpm

# Contributed by Trond Hasle Amundsen

# More efficient package method for RedHat - uses rpm to list instead of yum
# Notes:
# - using $(name).$(arch) instead of $(name) for package_name_convention
#   causes uninstallation to fail.
# - using allmatches to remove for all architectures
#

{
  package_changes => "bulk";
  package_list_command => "/bin/rpm -qa --qf '%{name} %{version}-%{release} %{arch}\n'";
  package_patch_list_command => "/usr/bin/yum --quiet check-update";

  package_list_name_regex    => "^(\S+?)\s\S+?\s\S+$";
  package_list_version_regex => "^\S+?\s(\S+?)\s\S+$";
  package_list_arch_regex    => "^\S+?\s\S+?\s(\S+)$";

  package_installed_regex => ".*";
  package_name_convention => "$(name)";

  # set it to "0" to avoid caching of list during upgrade
  package_list_update_command => "/usr/bin/yum --quiet check-update";
  package_list_update_ifelapsed => "240";

  package_patch_installed_regex => "^\s.*";
  package_patch_name_regex    => "([^.]+).*";
  package_patch_version_regex => "[^\s]\s+([^\s]+).*";
  package_patch_arch_regex    => "[^.]+\.([^\s]+).*";

  package_add_command    => "/usr/bin/yum -y install";
  package_update_command => "/usr/bin/yum -y update";
  package_patch_command => "/usr/bin/yum -y update";
  package_delete_command => "/bin/rpm -e --nodeps --allmatches";
  package_verify_command => "/bin/rpm -V";
}
```

1.106 body package_method zypper

```
body package_method zypper

{
package_changes => "bulk";

package_list_command => "/bin/rpm -qa --queryformat \"i | repos | %{name} | %{version}-%{release} |

# set it to "0" to avoid caching of list during upgrade
package_list_update_command => "/usr/bin/zypper list-updates";
package_list_update_ifelapsed => "240";

package_patch_list_command => "/usr/bin/zypper patches";
package_installed_regex => "i.*";
package_list_name_regex    => "[^|]+\|[^|]+\|\s+([^\s]+).*";
package_list_version_regex => "[^|]+\|[^|]+\|[^|]+\|\s+([^\s]+).*";
package_list_arch_regex    => "[^|]+\|[^|]+\|[^|]+\|[^|]+\|\s+([^\s]+).*";

package_patch_installed_regex => ".*Installed.*|.*Not Applicable.*";
package_patch_name_regex    => "[^|]+\|\s+([^\s]+).*";
package_patch_version_regex => "[^|]+\|[^|]+\|\s+([^\s]+).*";

package_name_convention => "$(name)";
package_add_command => "/usr/bin/zypper --non-interactive install";
package_delete_command => "/usr/bin/zypper --non-interactive remove --force-resolution";
package_update_command => "/usr/bin/zypper --non-interactive update";
package_patch_command => "/usr/bin/zypper --non-interactive patch$"; # $ means no args
package_verify_command => "/usr/bin/zypper --non-interactive verify$";
}
```

1.107 body perms m(mode)

```
body perms m(mode)
{
mode   => "$(mode)";
}
```

1.108 body perms mo(mode,user)

```
body perms mo(mode,user)
```

```
{
owners => { "$(user)" };
mode   => "$(mode)";
}
```

1.109 body perms mog(mode,user,group)

```
body perms mog(mode,user,group)
{
owners => { "$(user)" };
groups => { "$(group)" };
mode   => "$(mode)";
}
```

1.110 body perms og(u,g)

```
body perms og(u,g)
{
owners => { "$(u)" };
groups => { "$(g)" };
}
```

1.111 body perms owner(user)

```
body perms owner(user)
{
owners => { "$(user)" };
}
```

1.112 body process_count any_count(cl)

```
body process_count any_count(cl)

{
match_range => "0,0";
out_of_range_define => { "$(cl)" };
}
```

1.113 body process_count check_range(name,lower,upper)

```
body process_count check_range(name,lower,upper)
{
match_range => irange("$(lower)","$(upper)");
out_of_range_define => { "$(name)_out_of_range" };
}
```

1.114 body process_select days_older_than(d)

```
body process_select days_older_than(d)
{
stime_range    => irange(ago(0,0,"$(d)",0,0,0),now);
process_result => "stime";
}
```

1.115 body process_select exclude_procs(x)

```
body process_select exclude_procs(x)
{
command => "$(x)";
process_result => "!command";
}
```

1.116 body rename disable

```
body rename disable
{
disable => "true";
}
```

1.117 body rename rotate(level)

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

1.118 body rename to(file)

```
body rename to(file)
{
newname => "$(file)";
}
```

1.119 body replace_with comment(c)

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

1.120 body replace_with uncomment

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

1.121 body replace_with value(x)

```
body replace_with value(x)
{
replace_value => "$(x)";
occurrences => "all";
}
```

1.122 body select_region INI_section(x)

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

1.123 body service_method bootstart

```
body service_method bootstart
{
  service_autostart_policy => "boot_time";
  service_dependence_chain => "start_parent_services";
windows::
  service_type => "windows";
}
```

1.124 body service_method force_deps

```
body service_method force_deps
{
  service_dependence_chain => "all_related";
windows::
  service_type => "windows";
}
```

1.125 body volume min_free_space(free)

```
body volume min_free_space(free)
{
check_foreign  => "false";
freespace      => "$(free)";
sensible_size  => "10000";
sensible_count => "2";
}
```

1.126 bundle agent cronjob(commands,user,hours,mins)

```
bundle agent cronjob(commands,user,hours,mins)

 # For adding lines to crontab for a user
 # methods:
 #  "cron" usebundle => cronjob("/bin/ls","mark","*","5,10");

{
vars:
  SuSE::
   "crontab" string => "/var/spool/cron/tabs";
  redhat|fedora::
   "crontab" string => "/var/spool/cron";
```

```
 !(SuSE|redhat|fedora)::
    "crontab" string => "/var/spool/cron/crontabs";

files:

!windows::
  "$(crontab)/$(user)"

    comment => "A user's regular batch jobs are added to this file",
     create => "true",
  edit_line => append_if_no_line("$(mins) $(hours) * * * $(commands)"),
      perms => mo("644","$(user)"),
    classes => if_repaired("changed_crontab");

processes:

changed_crontab::
   "cron"
         comment => "Most crons need to be huped after file changes",
         signals => { "hup" };

}
```

1.127 bundle agent standard_services(service,state)

```
bundle agent standard_services(service,state)
{
 # DATA,

vars:

 any::

  "stakeholders[cfengine3]" slist => { "cfengine_in" };
  "stakeholders[acpid]" slist => { "cpu", "cpu0", "cpu1", "cpu2", "cpu3" };
  "stakeholders[mongod]" slist => { "mongo_in" };
  "stakeholders[postfix]" slist => { "smtp_in" };
  "stakeholders[sendmail]" slist => { "smtp_in" };
  "stakeholders[www]" slist => { "www_in", "wwws_in", "www_alt_in" };
  "stakeholders[ssh]" slist => { "ssh_in" };
  "stakeholders[mysql]" slist => { "mysql_in" };
  "stakeholders[nfs]" slist => { "nfsd_in" };
  "stakeholders[syslog]" slist => { "syslog" };
```

```
 "stakeholders[rsyslog]" slist => { "syslog" };
 "stakeholders[tomcat5]" slist => { "www_alt_in" };
 "stakeholders[tomcat6]" slist => { "www_alt_in" };

linux::

 "startcommand[acpid]" string => "/etc/init.d/acpid start";
 "stopcommand[acpid]"  string => "/etc/init.d/acpid stop";
 "pattern[acpid]"      string => ".*acpid.*";

 "startcommand[cfengine3]" string => "/etc/init.d/cfengine3 start";
 "stopcommand[cfengine3]"  string => "/etc/init.d/cfengine3 stop";
 "pattern[cfengine3]"      string => ".*cf-execd.*";

 "startcommand[fancontrol]" string => "/etc/init.d/fancontrol start";
 "stopcommand[fancontrol]"  string => "/etc/init.d/fancontrol stop";
 "pattern[fancontrol]"      string => ".*fancontrol.*";

 "startcommand[hddtemp]" string => "/etc/init.d/hddtemp start";
 "stopcommand[hddtemp]"  string => "/etc/init.d/hddtemp stop";
 "pattern[hddtemp]"      string => ".*hddtemp.*";

 "startcommand[irqbalance]" string => "/etc/init.d/irqbalance start";
 "stopcommand[irqbalance]"  string => "/etc/init.d/irqbalance stop";
 "pattern[irqbalance]"      string => ".*irqbalance.*";

 "startcommand[lm-sensor]" string => "/etc/init.d/lm-sensor start";
 "stopcommand[lm-sensor]"  string => "/etc/init.d/lm-sensor stop";
 "pattern[lm-sensor]"      string => ".*psensor.*";

 "startcommand[mongod]" string => "/etc/init.d/mongod start";
 "stopcommand[mongod]"  string => "/etc/init.d/mongod stop";
 "pattern[mongod]"      string => ".*mongod.*";

 "startcommand[openvpn]" string => "/etc/init.d/openvpn start";
 "stopcommand[openvpn]"  string => "/etc/init.d/openvpn stop";
 "pattern[openvpn]"      string => ".*openvpn.*";

 "startcommand[postfix]" string => "/etc/init.d/postfix start";
 "stopcommand[postfix]"  string => "/etc/init.d/postfix stop";
 "pattern[postfix]"      string => ".*postfix.*";

 "startcommand[rsync]" string => "/etc/init.d/rsync start";
 "stopcommand[rsync]"  string => "/etc/init.d/rsync stop";
 "pattern[rsync]"      string => ".*rsync.*";

 "startcommand[rsyslog]" string => "/etc/init.d/rsyslog start";
```

```
 "stopcommand[rsyslog]"  string => "/etc/init.d/rsyslog stop";
 "pattern[rsyslog]"      string => ".*rsyslogd.*";

 "startcommand[sendmail]" string => "/etc/init.d/sendmail start";
 "stopcommand[sendmail]"  string => "/etc/init.d/sendmail stop";
 "pattern[sendmail]"      string => ".*sendmail.*";

 "startcommand[ssh]" string => "/etc/init.d/sshd start";
 "stopcommand[ssh]"  string => "/etc/init.d/sshd stop";
 "pattern[ssh]"      string => ".*sshd.*";

 "startcommand[tomcat5]" string => "/etc/init.d/tomcat5 start";
 "stopcommand[tomcat5]"  string => "/etc/init.d/tomcat5 stop";
 "pattern[tomcat5]"      string => ".*tomcat5.*";

 "startcommand[tomcat6]" string => "/etc/init.d/tomcat6 start";
 "stopcommand[tomcat6]"  string => "/etc/init.d/tomcat6 stop";
 "pattern[tomcat6]"      string => ".*tomcat6.*";

 "startcommand[varnish]" string => "/etc/init.d/varnish start";
 "stopcommand[varnish]"  string => "/etc/init.d/varnish stop";
 "pattern[varnish]"      string => ".*varnish.*";

 "startcommand[wpa_supplicant]" string => "/etc/init.d/wpa_supplicant start";
 "stopcommand[wpa_supplicant]"  string => "/etc/init.d/wpa_supplicant stop";
 "pattern[wpa_supplicant]"      string => ".*wpa_supplicant.*";

SuSE|suse::

 "startcommand[mysql]" string => "/etc/init.d/mysqld start";
 "stopcommand[mysql]"  string => "/etc/init.d/mysqld stop";
 "pattern[mysql]"      string => ".*mysqld.*";

 "startcommand[www]" string => "/etc/init.d/apache2 start";
 "stopcommand[www]"  string => "/etc/init.d/apache2 stop";
 "pattern[www]"      string => ".*apache2.*";


redhat::

 "startcommand[anacron]" string => "/etc/init.d/anacron start";
 "stopcommand[anacron]"  string => "/etc/init.d/anacron stop";
 "pattern[anacron]"      string => ".*anacron.*";

 "startcommand[atd]" string => "/etc/init.d/atd start";
 "stopcommand[atd]"  string => "/etc/init.d/atd stop";
 "pattern[atd]"      string => ".*sbin/atd.*";
```

```
"startcommand[auditd]" string => "/etc/init.d/auditd start";
"stopcommand[auditd]"  string => "/etc/init.d/auditd stop";
"pattern[auditd]"      string => ".*auditd.*";

"startcommand[autofs]" string => "/etc/init.d/autofs start";
"stopcommand[autofs]"  string => "/etc/init.d/autofs stop";
"pattern[autofs]"      string => ".*automount.*";

"startcommand[bluetoothd]" string => "/etc/init.d/bluetooth start";
"stopcommand[bluetoothd]"  string => "/etc/init.d/bluetooth stop";
"pattern[bluetoothd]"      string => ".*hcid.*";

"startcommand[capi]" string => "/etc/init.d/capi start";
"stopcommand[capi]"  string => "/etc/init.d/capi stop";
"pattern[capi]"      string => ".*capiinit.*";

"startcommand[conman]" string => "/etc/init.d/conman start";
"stopcommand[conman]"  string => "/etc/init.d/conman stop";
"pattern[conman]"      string => ".*conmand.*";

"startcommand[cpuspeed]" string => "/etc/init.d/cpuspeed start";
"stopcommand[cpuspeed]"  string => "/etc/init.d/cpuspeed stop";
"pattern[cpuspeed]"      string => ".*cpuspeed.*";

"startcommand[crond]" string => "/etc/init.d/crond start";
"stopcommand[crond]"  string => "/etc/init.d/crond stop";
"pattern[crond]"      string => ".*crond.*";

"startcommand[dc_client]" string => "/etc/init.d/dc_client start";
"stopcommand[dc_client]"  string => "/etc/init.d/dc_client stop";
"pattern[dc_client]"      string => ".*dc_client.*";

"startcommand[dc_server]" string => "/etc/init.d/dc_server start";
"stopcommand[dc_server]"  string => "/etc/init.d/dc_server stop";
"pattern[dc_server]"      string => ".*dc_server.*";

"startcommand[dnsmasq]" string => "/etc/init.d/dnsmasq start";
"stopcommand[dnsmasq]"  string => "/etc/init.d/dnsmasq stop";
"pattern[dnsmasq]"      string => ".*dnsmasq.*";

"startcommand[dund]" string => "/etc/init.d/dund start";
"stopcommand[dund]"  string => "/etc/init.d/dund stop";
"pattern[dund]"      string => ".*dund.*";

"startcommand[gpm]" string => "/etc/init.d/gpm start";
"stopcommand[gpm]"  string => "/etc/init.d/gpm stop";
```

```
  "pattern[gpm]"      string => ".*gpm.*";

  "startcommand[haldaemon]" string => "/etc/init.d/haldaemon start";
  "stopcommand[haldaemon]"  string => "/etc/init.d/haldaemon stop";
  "pattern[haldaemon]"      string => ".*hald.*";

  "startcommand[hidd]" string => "/etc/init.d/hidd start";
  "stopcommand[hidd]"  string => "/etc/init.d/hidd stop";
  "pattern[hidd]"      string => ".*hidd.*";

#  "startcommand[ip6tables]" string => "/etc/init.d/ip6tables start";
#  "stopcommand[ip6tables]"  string => "/etc/init.d/ip6tables stop";
#  "pattern[ip6tables]"      string => ".*ip6tables.*";

#  "startcommand[iptables]" string => "/etc/init.d/iptables start";
#  "stopcommand[iptables]"  string => "/etc/init.d/iptables stop";
#  "pattern[iptables]"      string => ".*iptables.*";

  "startcommand[irda]" string => "/etc/init.d/irda start";
  "stopcommand[irda]"  string => "/etc/init.d/irda stop";
  "pattern[irda]"      string => ".*irattach.*";

  "startcommand[iscsid]" string => "/etc/init.d/iscsid start";
  "stopcommand[iscsid]"  string => "/etc/init.d/iscsid stop";
  "pattern[iscsid]"      string => ".*iscsid.*";

  "startcommand[isdn]" string => "/etc/init.d/isdn start";
  "stopcommand[isdn]"  string => "/etc/init.d/isdn stop";
  "pattern[isdn]"      string => ".*isdnlog.*";

  "startcommand[lvm2-monitor]" string => "/etc/init.d/lvm2-monitor start";
  "stopcommand[lvm2-monitor]"  string => "/etc/init.d/lvm2-monitor stop";
  "pattern[lvm2-monitor]"      string => ".*vgchange.*";

  "startcommand[mcstrans]" string => "/etc/init.d/mcstrans start";
  "stopcommand[mcstrans]"  string => "/etc/init.d/mcstrans stop";
  "pattern[mcstrans]"      string => ".*mcstransd.*";

  "startcommand[mdmonitor]" string => "/etc/init.d/mdmonitor start";
  "stopcommand[mdmonitor]"  string => "/etc/init.d/mdmonitor stop";
  "pattern[mdmonitor]"      string => ".*mdadm.*";

  "startcommand[mdmpd]" string => "/etc/init.d/mdmpd start";
  "stopcommand[mdmpd]"  string => "/etc/init.d/mdmpd stop";
  "pattern[mdmpd]"      string => ".*mdmpd.*";

  "startcommand[messagebus]" string => "/etc/init.d/messagebus start";
```

```
"stopcommand[messagebus]"  string => "/etc/init.d/messagebus stop";
"pattern[messagebus]"      string => ".*dbus-daemon.*";

"startcommand[microcode_ctl]" string => "/etc/init.d/microcode_ctl start";
"stopcommand[microcode_ctl]"  string => "/etc/init.d/microcode_ctl stop";
"pattern[microcode_ctl]"      string => ".*microcode_ctl.*";

"startcommand[multipathd]" string => "/etc/init.d/multipathd start";
"stopcommand[multipathd]"  string => "/etc/init.d/multipathd stop";
"pattern[multipathd]"      string => ".*multipathd.*";

"startcommand[mysql]" string => "/etc/init.d/mysqld start";
"stopcommand[mysql]"  string => "/etc/init.d/mysqld stop";
"pattern[mysql]"      string => ".*mysqld.*";

"startcommand[netplugd]" string => "/etc/init.d/netplugd start";
"stopcommand[netplugd]"  string => "/etc/init.d/netplugd stop";
"pattern[netplugd]"      string => ".*netplugd.*";

"startcommand[NetworkManager]" string => "/etc/init.d/NetworkManager start";
"stopcommand[NetworkManager]"  string => "/etc/init.d/NetworkManager stop";
"pattern[NetworkManager]"      string => ".*NetworkManager.*";

"startcommand[nfs]" string => "/etc/init.d/nfs start";
"stopcommand[nfs]"  string => "/etc/init.d/nfs stop";
"pattern[nfs]"      string => ".*nfsd.*";

"startcommand[nfslock]" string => "/etc/init.d/nfslock start";
"stopcommand[nfslock]"  string => "/etc/init.d/nfslock stop";
"pattern[nfslock]"      string => ".*rpc.statd.*";

"startcommand[nscd]" string => "/etc/init.d/nscd start";
"stopcommand[nscd]"  string => "/etc/init.d/nscd stop";
"pattern[nscd]"      string => ".*nscd.*";

"startcommand[oddjobd]" string => "/etc/init.d/oddjobd start";
"stopcommand[oddjobd]"  string => "/etc/init.d/oddjobd stop";
"pattern[oddjobd]"      string => ".*oddjobd.*";

"startcommand[pand]" string => "/etc/init.d/pand start";
"stopcommand[pand]"  string => "/etc/init.d/pand stop";
"pattern[pand]"      string => ".*pand.*";

"startcommand[pand]" string => "/etc/init.d/pand start";
"stopcommand[pcscd]"  string => "/etc/init.d/pand stop";
"pattern[pcscd]"      string => ".*pcscd.*";
```

```
"startcommand[portmap]" string => "/etc/init.d/portmap start";
"stopcommand[portmap]"  string => "/etc/init.d/portmap stop";
"pattern[portmap]"      string => ".*portmap.*";

"startcommand[postgresql]" string => "/etc/init.d/postgresql start";
"stopcommand[postgresql]"  string => "/etc/init.d/postgresql stop";
"pattern[postgresql]"      string => ".*postmaster.*";

"startcommand[rdisc]" string => "/etc/init.d/rdisc start";
"stopcommand[rdisc]"  string => "/etc/init.d/rdisc stop";
"pattern[rdisc]"      string => ".*rdisc.*";

"startcommand[rdisc]" string => "/etc/init.d/rdisc start";
"stopcommand[rdisc]"  string => "/etc/init.d/rdisc stop";
"pattern[rdisc]"      string => ".*rdisc.*";

"startcommand[readahead_early]" string => "/etc/init.d/readahead_early start";
"stopcommand[readahead_early]"  string => "/etc/init.d/readahead_early stop";
"pattern[readahead_early]"      string => ".*readahead.*early.*";

"startcommand[readahead_later]" string => "/etc/init.d/readahead_later start";
"stopcommand[readahead_later]"  string => "/etc/init.d/readahead_later stop";
"pattern[readahead_later]"      string => ".*readahead.*later.*";

"startcommand[restorecond]" string => "/etc/init.d/restorecond start";
"stopcommand[restorecond]"  string => "/etc/init.d/restorecond stop";
"pattern[restorecond]"      string => ".*restorecond.*";

"startcommand[rpcgssd]" string => "/etc/init.d/rpcgssd start";
"stopcommand[rpcgssd]"  string => "/etc/init.d/rpcgssd stop";
"pattern[rpcgssd]"      string => ".*rpc.gssd.*";

"startcommand[rpcidmapd]" string => "/etc/init.d/rpcidmapd start";
"stopcommand[rpcidmapd]"  string => "/etc/init.d/rpcidmapd stop";
"pattern[rpcidmapd]"      string => ".*rpc.idmapd.*";

"startcommand[rpcsvcgssd]" string => "/etc/init.d/rpcsvcgssd start";
"stopcommand[rpcsvcgssd]"  string => "/etc/init.d/rpcsvcgssd stop";
"pattern[rpcsvcgssd]"      string => ".*rpc.svcgssd.*";

"startcommand[saslauthd]" string => "/etc/init.d/saslauthd start";
"stopcommand[saslauthd]"  string => "/etc/init.d/saslauthd stop";
"pattern[saslauthd]"      string => ".*saslauthd.*";

"startcommand[smartd]" string => "/etc/init.d/smartd start";
"stopcommand[smartd]"  string => "/etc/init.d/smartd stop";
"pattern[smartd]"      string => ".*smartd.*";
```

```
 "startcommand[svnserve]" string => "/etc/init.d/svnserve start";
 "stopcommand[svnserve]"  string => "/etc/init.d/svnserve stop";
 "pattern[svnserve]"      string => ".*svnserve.*";

 "startcommand[syslog]" string => "/etc/init.d/syslog start";
 "stopcommand[syslog]"  string => "/etc/init.d/syslog stop";
 "pattern[syslog]"      string => ".*syslogd.*";

 "startcommand[tcsd]" string => "/etc/init.d/tcsd start";
 "stopcommand[tcsd]"  string => "/etc/init.d/tcsd stop";
 "pattern[tcsd]"      string => ".*tcsd.*";

 "startcommand[www]" string => "/etc/init.d/httpd start";
 "stopcommand[www]"  string => "/etc/init.d/httpd stop";
 "pattern[www]"      string => ".*httpd.*";

 "startcommand[xfs]" string => "/etc/init.d/xfs start";
 "stopcommand[xfs]"  string => "/etc/init.d/xfs stop";
 "pattern[xfs]"      string => ".*xfs.*";

 "startcommand[ypbind]" string => "/etc/init.d/ypbind start";
 "stopcommand[ypbind]"  string => "/etc/init.d/ypbind stop";
 "pattern[ypbind]"      string => ".*ypbind.*";

 "startcommand[yum-updatesd]" string => "/etc/init.d/yum-updatesd start";
 "stopcommand[yum-updatesd]"  string => "/etc/init.d/yum-updatesd stop";
 "pattern[yum-updatesd]"      string => ".*yum-updatesd.*";

debian|ubuntu::

 "startcommand[atd]" string => "/etc/init.d/atd start";
 "stopcommand[atd]"  string => "/etc/init.d/atd stop";
 "pattern[atd]"      string => "atd.*";

 "startcommand[bluetoothd]" string => "/etc/init.d/bluetoothd start";
 "stopcommand[bluetoothd]"  string => "/etc/init.d/bluetoothd stop";
 "pattern[bluetoothd]"      string => ".*bluetoothd.*";

 "startcommand[bootlogd]" string => "/etc/init.d/bootlogd start";
 "stopcommand[bootlogd]"  string => "/etc/init.d/bootlogd stop";
 "pattern[bootlogd]"      string => ".*bootlogd.*";

 "startcommand[crond]" string => "/etc/init.d/cron start";
 "stopcommand[crond]"  string => "/etc/init.d/cron stop";
 "pattern[crond]"      string => ".*cron.*";
```

```
 "startcommand[kerneloops]" string => "/etc/init.d/kerneloops start";
 "stopcommand[kerneloops]"  string => "/etc/init.d/kerneloops stop";
 "pattern[kerneloops]"      string => ".*kerneloops.*";

 "startcommand[mysql]" string => "/etc/init.d/mysql start";
 "stopcommand[mysql]"  string => "/etc/init.d/mysql stop";
 "pattern[mysql]"      string => ".*mysqld.*";

 "startcommand[NetworkManager]" string => "/etc/init.d/network-manager start";
 "stopcommand[NetworkManager]"  string => "/etc/init.d/network-manager stop";
 "pattern[NetworkManager]"      string => ".*NetworkManager.*";

 "startcommand[ondemand]" string => "/etc/init.d/ondemand start";
 "stopcommand[ondemand]"  string => "/etc/init.d/ondemand stop";
 "pattern[ondemand]"      string => ".*ondemand.*";

 "startcommand[plymouth]" string => "/etc/init.d/plymouthd start";
 "stopcommand[plymouth]"  string => "/etc/init.d/plymouthd stop";
 "pattern[plymouth]"      string => ".*plymouthd.*";

 "startcommand[postgresql84]" string => "/etc/init.d/postgresql-8.4 start";
 "stopcommand[postgresql84]"  string => "/etc/init.d/postgresql-8.4 stop";
 "pattern[postgresql84]"      string => ".*postgresql.*";

 "startcommand[postgresql91]" string => "/etc/init.d/postgresql-9.1 start";
 "stopcommand[postgresql91]"  string => "/etc/init.d/postgresql-9.1 stop";
 "pattern[postgresql91]"      string => ".*postgresql.*";

 "startcommand[saned]" string => "/etc/init.d/saned start";
 "stopcommand[saned]"  string => "/etc/init.d/saned stop";
 "pattern[saned]"      string => ".*saned.*";

 "startcommand[udev]" string => "/etc/init.d/udev start";
 "stopcommand[udev]"  string => "/etc/init.d/udev stop";
 "pattern[udev]"      string => ".*udev.*";

 "startcommand[udevmonitor]" string => "/etc/init.d/udevmonitor start";
 "stopcommand[udevmonitor]"  string => "/etc/init.d/udevmonitor stop";
 "pattern[udevmonitor]"      string => ".*udevadm.*monitor.*";

 "startcommand[www]"     string => "/etc/init.d/httpd stop";
 "stopcommand[www]"      string => "/etc/init.d/httpd stop";
 "pattern[www]"          string => ".*apache2.*";


# METHODS that implement these .....................................
```

```
classes:

  "start" expression => strcmp("start","$(state)"),
            comment => "Check if to start a service";
  "stop"  expression => strcmp("stop","$(state)"),
            comment => "Check if to stop a service";

# Do we want to include the packages here too?

processes:

  start::

    "$(pattern[$(service)])" ->  { "@(stakeholders[$(service)])" } ,

             comment => "Verify that the service appears in the process table",
       restart_class => "restart_$(service)";

  stop::

    "$(pattern[$(service)])" -> { "@(stakeholders[$(service)])" },

           comment => "Verify that the service does not appear in the process",
      process_stop => "$(stopcommand[$(service)])",
           signals => { "term", "kill"};

  commands:

   "$(startcommand[$(service)])" -> { "@(stakeholders[$(service)])" },

            comment => "Execute command to restart the $(service) service",
         ifvarclass => "restart_$(service)";
}
```

1.128 bundle common paths

```
bundle common paths
{
  vars:

      #
      # Common full pathname of commands for OS
      #
    aix::
```

```
    "awk"      string => "/usr/bin/awk";
    "crontabs" string => "/var/spool/cron/crontabs";
    "df"       string => "/usr/bin/df";
    "dig"      string => "/usr/bin/dig";
    "egrep"    string => "/usr/bin/egrep";
    "grep"     string => "/usr/bin/grep";
    "ls"       string => "/usr/bin/ls";
    "netstat"  string => "/usr/bin/netstat";
    "cksum"    string => "/usr/bin/cksum";
    "sort"     string => "/usr/bin/sort";
    "find"     string => "/usr/bin/find";
    "cat"      string => "/bin/cat";
    "sed"      string => "/usr/bin/sed";
    "diff"     string => "/usr/bin/diff";
    "cut"      string => "/usr/bin/cut";
    "printf"   string => "/usr/bin/printf";
    "tr"       string => "/usr/bin/tr";
    "echo"     string => "/usr/bin/echo";
    "bc"       string => "/usr/bin/bc";
    "dc"       string => "/usr/bin/dc";

  solaris::

    "awk"      string => "/usr/bin/awk";
    "crontabs" string => "/var/spool/cron/crontabs";
    "df"       string => "/usr/bin/df";
    "dig"      string => "/usr/sbin/dig";
    "egrep"    string => "/usr/bin/egrep";
    "grep"     string => "/usr/bin/grep";
    "ls"       string => "/usr/bin/ls";
    "netstat"  string => "/usr/bin/netstat";
    "cksum"    string => "/usr/bin/cksum";
    "sort"     string => "/usr/bin/sort";
    "find"     string => "/usr/bin/find";
    "cat"      string => "/usr/bin/cat";
    "sed"      string => "/usr/bin/sed";
    "diff"     string => "/usr/bin/diff";
    "cut"      string => "/usr/bin/cut";
    "printf"   string => "/usr/bin/printf";
    "tr"       string => "/usr/bin/tr";
    "echo"     string => "/usr/bin/echo";
    "bc"       string => "/usr/bin/bc";
    "dc"       string => "/usr/bin/dc";

    #
    "svcs"     string => "/usr/bin/svcs";
```

```
      "svcadm"   string => "/usr/sbin/svcadm";

    redhat::

      "awk"      string => "/bin/awk";
      "crontabs" string => "/var/spool/cron";
      "df"       string => "/bin/df";
      "dig"      string => "/usr/bin/dig";
      "egrep"    string => "/bin/egrep";
      "grep"     string => "/bin/grep";
      "ls"       string => "/bin/ls";
      "netstat"  string => "/bin/netstat";
      "cksum"    string => "/usr/bin/cksum";
      "sort"     string => "/bin/sort";
      "find"     string => "/usr/bin/find";
      "cat"      string => "/bin/cat";
      "sed"      string => "/bin/sed";
      "diff"     string => "/usr/bin/diff";
      "cut"      string => "/bin/cut";
      "printf"   string => "/usr/bin/printf";
      "tr"       string => "/usr/bin/tr";
      "echo"     string => "/bin/echo";
      "bc"       string => "/usr/bin/bc";
      "dc"       string => "/usr/bin/dc";

      #
      "svc"      string => "/sbin/service";
      "useradd"  string => "/usr/sbin/useradd";
      "groupadd" string => "/usr/sbin/groupadd";

    all::
      "ping" string => "/usr/bin/ping";

}
```

1.129 bundle edit_line append_groups_starting(v)

```
bundle edit_line append_groups_starting(v)

 # For adding groups to /etc/group, needs
 # an array v[groupname] string => "line..."

{
vars:

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

```
classes:

  "add_$(index)"     not => groupexists("$(index)"),
                 comment => "Class created if group does not exist";

insert_lines:

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

         comment => "Append users into a group file format",
      ifvarclass => "add_$(index)";

}
```

1.130 bundle edit_line append_if_no_line(str)

```
bundle edit_line append_if_no_line(str)
{
insert_lines:

 "$(str)"

     comment => "Append a line to the file if it doesn't already exist";
}
```

1.131 bundle edit_line append_if_no_lines(list)

```
bundle edit_line append_if_no_lines(list)
{
insert_lines:

 "$(list)"

   comment => "Append lines to the file if they don't already exist";
}
```

1.132 bundle edit_line append_to_line_end(start,end)

```
bundle edit_line append_to_line_end(start,end)
#
# Lines starting with "$(start)" and not ending with "$(end)"
# will get appended with "$(end)", whitespaces will be left unmodified.
```

```
# For example, append_to_line_end("kernel", "vga=791") would replace
# "kernel /boot/vmlinuz root=/dev/sda7"
# with
# "kernel /boot/vmlinuz root=/dev/sda7 resume=/dev/sda9 vga=791"
#
# WARNING: Be careful not to have multiple promises matching the same line,
#          which would result in the line growing indefinetively.
{
field_edits:

   "\s*$(start)\s.*"
         comment => "Append lines with $(this.start) and $(this.end)",
      edit_field => line("(^|\s)$(start)\s*", "2", "$(end)","append");
}
```

1.133 bundle edit_line append_user_field(group,field,allusers)

```
bundle edit_line append_user_field(group,field,allusers)

 # For adding users to to a file like /etc/group
 # at field position "field", comma separated subfields

{
vars:

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

field_edits:

 "$(group):.*"

       comment => "Append users into a password file format",
    edit_field => col(":","$(field)","$(val)","alphanum");
}
```

1.134 bundle edit_line append_users_starting(v)

```
bundle edit_line append_users_starting(v)

 # For adding to /etc/passwd or etc/shadow, needs
 # an array v[username] string => "line..."

{
vars:
```

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

classes:

  "add_$(index)"     not => userexists("$(index)"),
                 comment => "Class created if user does not exist";

insert_lines:

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

         comment => "Append users into a password file format",
      ifvarclass => "add_$(index)";
}
```

1.135 bundle edit_line comment_lines_containing(regex,comment)

```
bundle edit_line comment_lines_containing(regex,comment)

 # Comment lines of a file containing a regex

{
replace_patterns:

 "^(.*$(regex).*)$"

     replace_with => comment("$(comment)"),
          comment => "Comment out lines in a file";
}
```

1.136 bundle edit_line comment_lines_matching(regex,comment)

```
bundle edit_line comment_lines_matching(regex,comment)

 # Comment lines of a file matching a regex

{
replace_patterns:

 "^($(regex))$"

     replace_with => comment("$(comment)"),
          comment => "Search and replace string";
```

```
}
```

1.137 bundle edit_line create_solaris_admin_file

```
bundle edit_line create_solaris_admin_file
{
insert_lines:

  "mail=
instance=unique
partial=nocheck
runlevel=nocheck
idepend=nocheck
rdepend=nocheck
space=nocheck
setuid=nocheck
conflict=nocheck
action=nocheck
networktimeout=60
networkretries=3
authentication=quit
keystore=/var/sadm/security
proxy=
basedir=default",
      comment => "Insert contents of Solaris admin file (automatically install packages)";
}
```

1.138 bundle edit_line delete_lines_matching(regex)

```
bundle edit_line delete_lines_matching(regex)
{
delete_lines:

  "$(regex)"

     comment => "Delete lines matching regular expressions";
}
```

1.139 bundle edit_line expand_template(templatefile)

```
bundle edit_line expand_template(templatefile)
```

```
 # Read in the named text file and expand $(var)
 # inside the file

{
insert_lines:

   "$(templatefile)"

        insert_type => "file",
            comment => "Expand variables in the template file",
     expand_scalars => "true";
}
```

1.140 bundle edit_line insert_file(templatefile)

```
bundle edit_line insert_file(templatefile)
{
insert_lines:

   "$(templatefile)"
            comment => "Insert the template file into the file being edited",
        insert_type => "file";
}
```

1.141 bundle edit_line insert_lines(lines)

```
bundle edit_line insert_lines(lines)
{
insert_lines:

  "$(lines)"
    comment => "Append lines if they don't exist";
}
```

1.142 bundle edit_line maintain_key_values(v,sep)

```
bundle edit_line maintain_key_values(v,sep)

 # Contributed by David Lee
 # Purpose: Sets the RHS of configuration items with an giving separator

{
  vars:
```

```
    "index" slist => getindices("$(v)");
    # Be careful if the index string contains funny chars
    "cindex[$(index)]" string => canonify("$(index)");
    # Matching pattern for line (basically key-and-separator)
    "keypat[$(index)]" string => "\s*$(index)\s*$(sep)\s*";

    # Values may contain regexps. Escape them for replace_pattern matching.
    "ve[$(index)]" string => escape("$($(v)[$(index)])");

  classes:
    "$(cindex[$(index)])_key_in_file"
       comment => "Dynamic Class created if patterns matching",
       expression => regline("^$(keypat[$(index)]).*", "$(edit.filename)");

  replace_patterns:
    # For convergence need to use negative lookahead on value:
    # "key sep (?!value).*"
    "^($(keypat[$(index)]))(?!$(ve[$(index)])).*"
      comment => "Replace definition of $(index)",
      replace_with => value("$(match.1)$($(v)[$(index)])");

  insert_lines:
    "$(index)$(sep)$($(v)[$(index)])",
      comment => "Insert definition of $(index)",
      ifvarclass => "!$(cindex[$(index)])_key_in_file";
}
```

1.143 bundle edit_line manage_variable_values_ini(tab, sectionName)

```
bundle edit_line manage_variable_values_ini(tab, sectionName)

 # Sets the RHS of configuration items in the file of the form
 # LHS=RHS
 # If the line is commented out with #, it gets uncommented first.
 # Adds a new line if none exists.
 # Removes any variable value pairs not defined for the ini section
 # The argument is an associative array containing tab[SectionName][LHS]="RHS"
 # don't change value when the RHS is dontchange

 # Based on set_variable_values_ini
 # Added delete lines section to empty out undefined variable values for INI section

 # CAUTION : for it to work nicely, you should use Cfengine with the commit n3229
 # otherwise you may risk a segfault
```

```
 #
 # If you are running 3.2.1 or earlier or more specifically git commit # or
 # earlier you can use this to work around the segfault bug.
 # vars:
 #     "$(file)"
 #         edit_line => append_if_no_line("[#EOF#]"),
 #         create => "true",
 #         comment => "Work around bug<bug#here> where eof did not mean end
 #                     of section and thus cannot select a region. This promise
 #                     should be placed before your call to this bundle";
 #      "$(file)"
 #         edit_line   => manage_variable_values_ini("context.array", "SectionName"),
 #         create => "true",
 #         comment     => "Set the variale values only in the specified ini region";
 #

{
    vars:
        "index" slist => getindices("$(tab)[$(sectionName)]");

    # Be careful if the index string contains funny chars
        "cindex[$(index)]" string => canonify("$(index)");

    classes:
        "edit_$(cindex[$(index)])"     not => strcmp("$($(tab)[$(sectionName)][$(index)])","dontchan
                                  comment => "Create conditions to make changes";

    field_edits:

    # If the line is there, but commented out, first uncomment it
        "#+$(index)\s*=.*"
            select_region => INI_section("$(sectionName)"),
            edit_field => col("=","1","$(index)","set"),
            ifvarclass => "edit_$(cindex[$(index)])";

    # match a line starting like the key something
        "$(index)\s*=.*"
            edit_field => col("=","2","$($(tab)[$(sectionName)][$(index)])","set"),
            select_region => INI_section("$(sectionName)"),
            classes => if_ok("manage_variable_values_ini_not_$(cindex[$(index)])"),
            ifvarclass => "edit_$(cindex[$(index)])";

    delete_lines:
        ".*"
            select_region => INI_section("$(sectionName)"),
            comment       => "Remove all entries in the region so there are no extra entries";
```

```
    insert_lines:
        "[$(sectionName)]"
                location => start,
                 comment => "Insert lines";

        "$(index)=$($(tab)[$(sectionName)][$(index)])",
                select_region => INI_section("$(sectionName)"),
                ifvarclass => "!manage_variable_values_ini_not_$(cindex[$(index)]).edit_$(cindex[$(i

}
```

1.144 bundle edit_line replace_line_end(start,end)

```
bundle edit_line replace_line_end(start,end)
#
# Lines starting with "$(start)" will get the ending given in "$(end)",
# whitespaces will be left unmodified.
# For example, replace_line_end("ftp", "2121/tcp") would replace
# "ftp             21/tcp"
# with
# "ftp             2121/tcp"
{
field_edits:

   "\s*$(start)\s.*"
         comment => "Replace lines with $(this.start) and $(this.end)",
      edit_field => line("(^|\s)$(start)\s*", "2", "$(end)","set");
}
```

1.145 bundle edit_line replace_or_add(pattern,line)

```
bundle edit_line replace_or_add(pattern,line)

 # Replace a pattern in a file with a single line.
 # If the pattern is not found, add the line to the file.
 # The pattern must match the whole line (it is automatically
 # anchored to the start and end of the line) to avoid
 # ambiguity.

{
vars:
  "cline" string => canonify("$(line)");

replace_patterns:
```

```
  "^(?!$(line)$)$(pattern)$"
         comment => "Replace a pattern here",
     replace_with => value("$(line)"),
         classes => always("replace_done_$(cline)");

insert_lines:
  "$(line)"
    ifvarclass => "replace_done_$(cline)";
}
```

1.146 bundle edit_line resolvconf(search,list)

```
bundle edit_line resolvconf(search,list)

 # search is the search domains with space
 # list is an slist of nameserver addresses

{
delete_lines:

  "search.*"     comment => "Reset search lines from resolver";
  "nameserver.*" comment => "Reset nameservers in resolver";

insert_lines:

  "search $(search)"    comment => "Add search domains to resolver";
  "nameserver $(list)"  comment => "Add name servers to resolver";
}
```

1.147 bundle edit_line set_colon_field(key,field,val)

```
bundle edit_line set_colon_field(key,field,val)

 # Set the value of field number "field" of the
 # line whose first field is "key", in a colon-separated file.

{
field_edits:

  "$(key):.*"

       comment => "Edit a colon-separated file, using the first field as a key",
     edit_field => col(":","$(field)","$(val)","set");
}
```

1.148 bundle edit_line set_config_values(v)

```
bundle edit_line set_config_values(v)

 # Sets the RHS of configuration items in the file of the form
 #   LHS RHS
 # If the line is commented out with #, it gets uncommented first.
 # Adds a new line if none exists.
 # The argument is the fully-qualified name of an associative array containing v[LHS]="rhs"

{
vars:
  "index" slist => getindices("$(v)");

  # Be careful if the index string contains funny chars
  "cindex[$(index)]" string => canonify("$(index)");

replace_patterns:
  # If the line is there, maybe commented out, uncomment and replace with
  # the correct value
  "^\s*($(index)\s+(?!$($(v)[$(index)])$).*|# ?$(index)\s+.*)$"
         comment => "Correct the value",
    replace_with => value("$(index) $($(v)[$(index)])"),
         classes => always("replace_attempted_$(cindex[$(index)])");

insert_lines:
  "$(index) $($(v)[$(index)])"
    ifvarclass => "replace_attempted_$(cindex[$(index)])";

}
```

1.149 bundle edit_line set_config_values_matching(v,pat)

```
bundle edit_line set_config_values_matching(v,pat)

 # Sets the RHS of configuration items in the file of the form
 #   LHS RHS
 # If the line is commented out with #, it gets uncommented first.
 # Adds a new line if none exists.
 # Only elements of "v" that match the regex "pat" are used
 # The argument is the fully-qualified name of an associative array containing v[LHS]="rhs"

{
vars:
```

```
  "allparams" slist => getindices("$(v)");
  "index"     slist => grep("$(pat)", "allparams");

  # Be careful if the index string contains funny chars
  "cindex[$(index)]" string => canonify("$(index)");

replace_patterns:
    # If the line is there, maybe commented out, uncomment and replace with
    # the correct value
    "^\s*($(index)\s+(?!$($(v)[$(index)])).*|# ?$(index)\s+.*)$"
           comment => "Correct the value",
      replace_with => value("$(index) $($(v)[$(index)])"),
           classes => always("replace_attempted_$(cindex[$(index)])");

insert_lines:
  "$(index) $($(v)[$(index)])"
    ifvarclass => "replace_attempted_$(cindex[$(index)])";

}
```

1.150 bundle edit_line set_user_field(user,field,val)

```
bundle edit_line set_user_field(user,field,val)

 # Set the value of field number "field" in
 # a :-field formatted file like /etc/passwd

{
field_edits:

 "$(user):.*"

        comment => "Edit a user attribute in the password file",
     edit_field => col(":","$(field)","$(val)","set");
}
```

1.151 bundle edit_line set_variable_values(v)

```
bundle edit_line set_variable_values(v)

 # Sets the RHS of variables in the file of the form
 #   LHS = RHS
 # Adds a new line if no LHS exists, repairs RHS values if one does exist
 #
```

```
 # To use:
 #   1) Define an array, where the keys are the LHS and the values are the RHS
 #        "stuff[lhs-1]" string => "rhs1";
 #        "stuff[lhs-2]" string => "rhs2";
 #   2) The parameter passed to the edit_line promise is the fully qualified
 #      name of the array (i.e., "bundlename.stuff") WITHOUT any "$" or "@"

{
vars:

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

  # Be careful if the index string contains funny chars

  "cindex[$(index)]" string => canonify("$(index)");

field_edits:

  # match a line starting like the key = something

  "\s*$(index)\s*=.*"

     edit_field => col("=","2","$($(v)[$(index)])","set"),
        classes => if_ok("$(cindex[$(index)])_in_file"),
        comment => "Match a line starting like key = something";

insert_lines:

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

         comment => "Insert a variable definition",
      ifvarclass => "!$(cindex[$(index)])_in_file";
}
```

1.152 bundle edit_line set_variable_values_ini(tab, sectionName)

```
bundle edit_line set_variable_values_ini(tab, sectionName)

 # Sets the RHS of configuration items in the file of the form
 # LHS=RHS
 # If the line is commented out with #, it gets uncommented first.
 # Adds a new line if none exists.
 # The argument is an associative array containing tab[SectionName][LHS]="RHS"
 # don't change value when the RHS is dontchange

 # Based on set_variable_values from cfengine_stdlib.cf, modified to
```

```
 # use section to define were to write, and to handle commented-out lines.

 # CAUTION : for it to work nicely, you should use Cfengine with the commit n3229
 # otherwise you may risk a segfault

 #
 # If you are running 3.2.1 or earlier or more specifically git commit # or
 # earlier you can use this to work around the segfault bug.
 # vars:
 #     "$(file)"
 #         edit_line => append_if_no_line("[#EOF#]"),
 #         create => "true",
 #         comment => "Work around bug<bug#here> where eof did not mean end
 #                     of section and thus cannot select a region. This promise
 #                     should be placed before your call to this bundle";
 #      "$(file)"
 #         edit_line   => set_variable_values_ini("context.array", "SectionName"),
 #         create => "true",
 #         comment     => "Set the variale values only in the specified ini region";
 #

{
    vars:
        "index" slist => getindices("$(tab)[$(sectionName)]");

    # Be careful if the index string contains funny chars
        "cindex[$(index)]" string => canonify("$(index)");

    classes:
        "edit_$(cindex[$(index)])"     not => strcmp("$($(tab)[$(sectionName)][$(index)])","dontchan
                                   comment => "Create conditions to make changes";

    field_edits:

    # If the line is there, but commented out, first uncomment it
        "#+$(index)\s*=.*"
            select_region => INI_section("$(sectionName)"),
            edit_field => col("=","1","$(index)","set"),
            ifvarclass => "edit_$(cindex[$(index)])";

    # match a line starting like the key something
        "$(index)\s*=.*"
            edit_field => col("=","2","$($(tab)[$(sectionName)][$(index)])","set"),
            select_region => INI_section("$(sectionName)"),
            classes => if_ok("set_variable_values_ini_not_$(cindex[$(index)])"),
            ifvarclass => "edit_$(cindex[$(index)])";
```

```
    insert_lines:
        "[$(sectionName)]"
                location => start,
                 comment => "Insert lines";

        "$(index)=$($(tab)[$(sectionName)][$(index)])",
                select_region => INI_section("$(sectionName)"),
                ifvarclass => "!set_variable_values_ini_not_$(cindex[$(index)]).edit_$(cindex[$(inde

}
```

1.153 bundle edit_line uncomment_lines_containing(regex,comment)

```
bundle edit_line uncomment_lines_containing(regex,comment)

 # Uncomment lines of a file where the regex matches
 # the text after the comment string

{
replace_patterns:

 "^$(comment)\s?(.*$(regex).*)$"

    replace_with => uncomment,
         comment => "Uncomment a line containing a fragment";
}
```

1.154 bundle edit_line uncomment_lines_matching(regex,comment)

```
bundle edit_line uncomment_lines_matching(regex,comment)

 # Uncomment lines of a file where the regex matches
 # the text after the comment string

{
replace_patterns:

 "^$(comment)\s?($(regex))$"

       replace_with => uncomment,
            comment => "Uncomment lines matching a regular expression";
}
```

1.155 bundle edit_line warn_lines_matching(regex)

```
bundle edit_line warn_lines_matching(regex)
{
delete_lines:

  "$(regex)"

   comment => "Warn about lines in a file",
    action => warn_only;
}
```