

# Config::Model and configuration upgrades during package upgrade

Dominique Dumont

Debian Perl Group

FOSDEM 2010

# Outline

- 1 Why
  - Configuration upgrade problems
  - Objectives
- 2 Config::Model
  - Overview
  - Configuration model creation
  - User point of view
- 3 Package upgrades
- 4 Status

## Configuration is often painful !

Configuration upgrade is often difficult for a user:

- Surprise question during upgrade
- Edit a text file outside of /home
- Read man pages
- Ensure consistency
- Leave spurious files

Basic configuration may also be difficult...

## Objective 1: Make configuration easier for users

Handle configuration upgrade smoothly (mostly no interaction)

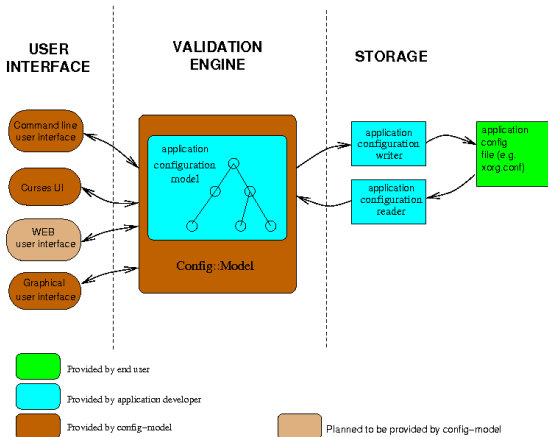
Provide a graphical interface with:

- Integrated help
- Default values
- Validation of configuration data
- Several levels of skills (*from beginner to master*)
- Search

## Objectif 2: Make maintenance easy for developers

- Configuration tool and upgrader must be easy to maintain:
  - Avoid ad-hoc validation code (e.g. don't rewrite Webmin)
  - Base validation on "meta-data": the *configuration model*
  - Generate interfaces (graphicals or not) from the model
  - Model contains properties to upgrade configuration
  - GUI to create and maintain models
- Minimise code required to read or write configuration files:
  - Use existing libraries (Config::Ini, Config::Augeas – and all Augeas lenses...)
  - Provide basic classes to help configuration reads and writes

# Config::Model design



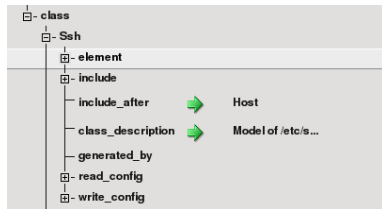
## What is a model?

Config is represented in a tree.  
The model defines its  
structure:

- A class is represented by a node
- A parameter is represented by a leaf

Each class contains :

- a set of elements (parameters)
- optional: a specification to access configuration file (backend)

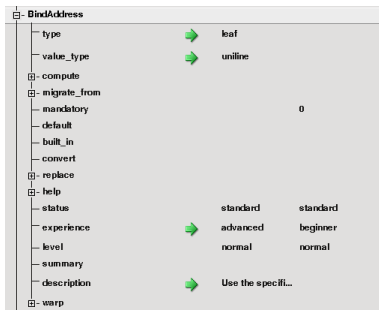


model GUI

## Simple elements

Each element has:

- a type (leaf, hash, list, node)
- constraints (integer, max, mini ... )
- a default value
- a description and a summary (for integrated help)
- an experience level (beginner, advanced, master)
- a status (normal or obsolete)



Model GUI

# Model analysis

- Read the application man pages:
  - Find the structure of the tree
  - Identify configuration parameters, their constraints and relations
- Find several valid examples:
  - To verify that the documentation was understood
  - For the non-regression tests

## Model declaration

In summary, configuration documentation is translated in a format usable by Config::Model:

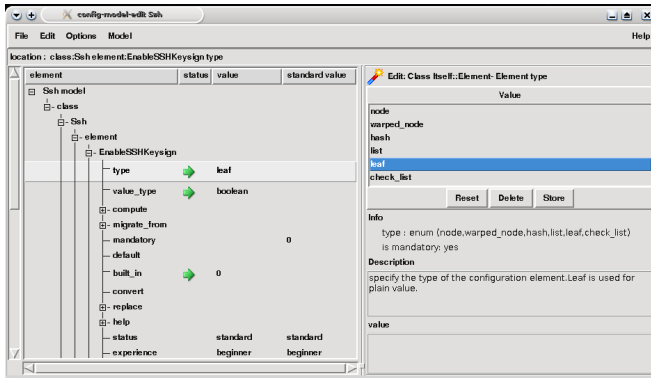
- The structure is translated into configuration classes
- Configuration parameters into elements
- Constraints into element attributes

```
name => 'Ssh',          # class name
element => [
  EnableSSHKeysign => { # element name
    type => 'leaf',
    value_type => 'boolean',
    built_in => '0',    # default value
    description => 'Setting ...',
  },
]
```

See [http://sourceforge.net/apps/mediawiki/config-model/index.php?title=Creating\\_a\\_model](http://sourceforge.net/apps/mediawiki/config-model/index.php?title=Creating_a_model)

## Declaration (easier mode)

Since writing a data structure is not fun (even with Perl), a model can be create with a GUI:



From time to time, do a Menu → Model → test

# Prepare configuration updates

For smooth upgrades

For application designers:

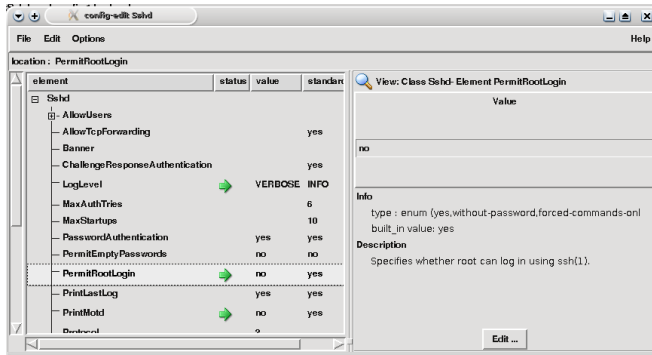
- 1 No new parameters  $\leftrightarrow$  no new problems
- 2 Picking parameter name and value : A good name is better than 3 pages of doc
- 3 Default values : Application can work with an empty config file

But, if needed, model and backend can specify:

- How to replace a value (replace)
- Obsolete parameters (status)
- How to migrate a value (migrate\_from + formula)
- Migration from one syntax with another (backends)

For more information on migration applied to software packages, see <http://wiki.debian.org/PackageConfigUpgrade>

# Configuration GUI



Note: Menu Option → experience to show more parameters

## Configuration and package upgrades

Package upgrade:

- RedHat: Configuration evolutions leave rpm.new or rpm.save file
- Debian: Configuration evolution either:
  - trigger questions (often cryptic)
  - expose details to user with a diff
  - leave spurious files (dpkg-new or dpkg-old)

In all cases

Merging configuration requires good knowledge from user.

## Configuration and package upgrades

### Proposal

Use Config::Model to merge :

- user data from config file
- package/upstream evolutions from model

Models with merge capability can be implemented by:

- Upstream projects
- Distributions (Debian, RedHat ...)
- Derived distribution (Knoppix, SkoleLinux ...)

Each can improve model coming from upstream

See proposal for Debian:

<http://wiki.debian.org/PackageConfigUpgrade>

# Configuration upgrade example

sshd\_config: TCPKeepAlive option was formerly called KeepAlive.

```
KeepAlive => { value_type => 'enum',
               status    => 'deprecated',
               type      => 'leaf',
               choice    => [ 'no', 'yes' ]
             },

TCPKeepAlive => { value_type => 'enum',
                  type      => 'leaf',
                  choice    => [ 'no', 'yes' ],
                  migrate_from
=> { formula    => '$keep_alive',
    variables => { keep_alive => '- KeepAlive' },
    },
```

## Package upgrade howto

### Debian

In package build instructions (*debian/rules* file):

```
dh_config_model_upgrade --model_name Sshd \  
--model_package libconfig-model-sshd-perl
```

### RedHat

In postinst:

```
config-edit --model Sshd -ui none -save
```

## Project status

### Available Models

- OpenSsh
- Approx
- Config::Model
- Krb5
- Xorg

### Backend

- INI syntax
- Perl data structure
- YAML (on-going)
- Augeas

### Community

- Debian packages
- Rpm packages (under constructions)
- Mandriva packages
- Proposal and patches for dh\_config (package upgrades)
- Article in GNU/Linux Mag France

## Future projects

### Interfaces

- Search parameters, values and help
- Annotations (e.g. comments)

### backend

- JSON
- XML
- Other ?

### We need you !

Config::Model needs your help:

- Integration in distros
- Multi-level configuration
- Plug-in mechanism for models (Xorg drivers)
- Define mechanism for configuration injection (e.g. mercurial viewer in Apache)

## Links

- Config::Model site  
<http://config-model.wiki.sourceforge.net>
- Config::Model on CPAN  
<http://search.cpan.org/dist/Config-Model/>
- Config::Model user mailing list <https://lists.sourceforge.net/lists/listinfo/config-model-users>
- GNU/Linux Mag France n°117 and n°120 "Config::Model - Créer un éditeur graphique de configuration avec Perl" (2 parts)
- Proposal to use Config::Model to upgrade configuration during Debian package upgrade  
<http://wiki.debian.org/PackageConfigUpgrade>
- Augeas project <http://augeas.net>