

OpenWrt: UCI and beyond

Felix Fietkau, John Crispin

February 18, 2009

What is UCI?

- *Unified Configuration Interface*
- Simple configuration format
- Human readable/writable
- Based on typed sections, containing option/value pairs, lists
- Designed for 90% of typical routers' configuration needs

Example

```
config interface 'loopback'
    option ifname    'lo'
    option proto     'static'
    option ipaddr    '127.0.0.1'
    option netmask   '255.0.0.0'

config interface 'lan'
    option ifname    'eth0'
    option type      'bridge'
    option proto     'static'
    option ipaddr    '192.168.1.1'
    option netmask   '255.255.255.0'
```

History of UCI

- Project initiated at Fraunhofer FOKUS by Alexander Morlang
- Initial code written by
 - Carsten Tittel (FOKUS)
 - Felix Fietkau (OpenWrt)
- Simple reference implementation
 - shell/awk functions for read/write access
 - only sections/options supported (no lists)
 - limited validation support based on awk
- Reference implementation for remote SNMP config access
 - TODO: port it to the new system

Timeline

- October 2006: Integration in OpenWrt
- January 2008: Rewrite in C
 - Input sanitization for shell API
- February 2008: Switch to the new UCI in OpenWrt
- April 2008: Lua binding
 - LuCI integration
- November 2008: libucimap
 - Map UCI sections to/from C data structures
 - Easier integration for third party C apps

Current state of UCI

- API for C, Shell, Lua
- Used in almost all OpenWrt base packages
- LuCI (Lua Configuration Interface)
 - MVC web application framework
 - OpenWrt configuration interface
- Validation framework (Lua)
 - contributed by the LuCI team

Data storage

- Config split by packages: `/etc/config/<package>`
- Example: `/etc/config/network`

```
config interface 'lan'  
    option ipaddr '192.168.1.1'
```

- Access through UCI pointers:

```
# uci get network.lan.ipaddr  
192.168.1.1
```

- Changes are staged, then committed/reverted
- Multiple backend support
- Multiple overlays possible
 - Useful for storing config state

API types

- Simple init scripts: shell API
 - config data parsed as shell code
 - very fast on embedded hardware
- Simple C applications: ucix wrappers
 - simplified wrapper API with limited functionality
- Regular C API
 - direct access to data structures
 - full functionality
- libucimap
 - automatic conversion to/from C data structures
- Lua binding
 - efficient scripting access (e.g. for web applications)

What's next?

- UCI support in more applications/packages
- Automatically generated config interfaces (from validation)
- Transparent config access from remote sources (backends)
- ...

- Questions?