DNS Servers and RESTCONF

Secure Remote Configuration and Management

Ladislav Lhotka • lhotka@nic.cz • 16 October 2017
Agenda

• motivation
• standards and technologies
• JetConf: RESTCONF server implementation
• backend for Knot DNS
• data model for server management
• data model for zone data
• access control
• clients
• related work
Motivation

Effective management of DNS servers:

- automation, scripting
- reliable retrieval of state data
- remote execution of commands
- access control with fine granularity

Existing alternatives:

- Ansible, Puppet, Expect, … - work but have difficulties with unexpected errors, require screenscraping.
- SNMP - used commonly for data acquisition but rarely for configuration.
Standards and Technologies

RESTCONF protocol [RFC 8040]

• management protocol utilizing HTTP methods and features
• TLS is mandatory
• supports configuration and state data, RPC operations and asynchronous notifications
• multiple representations (JSON and XML)

YANG data modelling language [RFC 7950]

• inspired by W3C XML Schema, RELAX NG and SMI
• models data hierarchy, scalar types and semantic constraints
• formal validation against the data model
• detailed documentation of data
JetConf: RESTCONF Server Implementation

GitHub project: https://github.com/CZ-NIC/jetconf

Main features:

• written in Python 3, uses HTTP/2 over TLS
• certificate-based authentication of clients
• all mandatory functionality specified in RFC 8040
• Per-user staging datastores with transactions
• API for device- or server-specific backends

Planned:

• asynchronous notifications (probably via websockets)
Knot DNS Backend

GitHub project: https://github.com/CZ-NIC/knot-jetconf

Works with a specific data model, implements configuration commit, callbacks for selected state data nodes and operations.

Easy to write: Knot DNS provides a Unix socket for inspecting and manipulating configuration, reading and updating zones etc.

Backends for other DNS server implementations would be less straightforward.
Server Data Model

GitHub project:  https://github.com/CZ-NIC/dns-server-yang

YANG modularity is used for separating data that are common to major DNS server implementations:

- BIND 9
- NSD
- Knot DNS
- PowerDNS

Each implementation then adds its specific data or operations (only available for Knot DNS so far).
Fragment of the Schema Tree

```plaintext
---rw resources
  ---rw max-tcp-clients? uint16
  ---rw max-udp-size? uint16
  ---rw knot:tcp-workers? uint8
  ---rw knot:udp-workers? uint8
  ---rw knot:background-workers? uint8
  ---rw knot:tcp-idle-timeout? uint32
  ---rw knot:tcp-handshake-timeout? uint32
  ---rw knot:tcp-reply-timeout? uint32
```
Model for Zone Data

GitHub project:  https://github.com/CZ-NIC/zone-data-yang

JetConf keeps complete server configuration data in datastores – running and at most one staging per user.

In contrast, zone data may get very big, so they are edited only via RPC operations, e.g. zone-set adds a specific resource record to a zone.
Access Control

- RFC 6536: Network Configuration Protocol (NETCONF) Access Control Model
- draft-ietf-netconf-rfc6536bis (updates for RESTCONF)

NACM rules may be assigned to specific data nodes or operations. They define CRUDX access rights for certain groups of users.

In JetConf, NACM rules can be configured only by the superuser.
JetConf Clients

JetScreen: generic data browser

Preferrable approach: use the REST API and integrate it into existing websites.

Scripting is also easy through tools like `curl`.
Related Work

By Sara and John Dickinson:

- DNSCCM project - uses NETCONF
  https://portal.sinodun.com/wiki/display/DNSCCM/DNSCCM+Home

- draft-dickinson-dnsop-nameserver-control (expired) - an attempt to standardize the DNS server data mode.