Release of getdns-0.1.0

APPAREA, IETF 89

getdns core team

Allison Mankin, Duane Wessels (Verisign Labs)

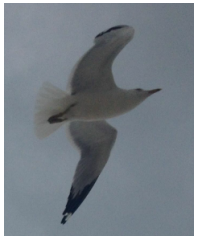
Craig Despeaux, Neel Goyal, Glen Wiley (Verisign)

Olaf Kolkman, Willem Toorop, Wouter Wijngaards (NLnet Labs)

Melinda Shore, No Mountain Software

Outline

- Background: getdns-api spec
- Open source implementation
- Major features of this release
- Coming soon



Background of getdns-api

- Paul Hoffman edited as an app-oriented DNS API, first publication April 2013. His slide from APPAREA, IETF 86:
  - "Fully asynchronous,* has multiple ways of using DNSSEC, supports new DNS types"
  - Expanded points
    - Default async
    - Eased leveraging of DANE, DNSSEC, SRV, etc
    - Extensible
- Updated getdns-api February 2014
    - Extensive discussions during the implementation

Acknowledgements

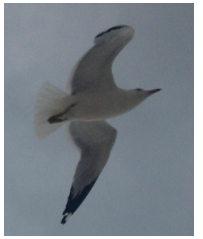
- Paul Hoffman, along with the original getdns-api design group, and the denizens of the getdns-api mailing list
- Matt Larson, who first envisioned this open source project

Open Source Implementation

- Two research labs, Verisign Labs and NLnet Labs, with long-standing interest in enabling DNS innovation and DNS-supported security
  - Also on team: OSS development and QA engineers
- Open source implementation in C with BSD-New license
  - https://github.com/getdnsapi/getdns
- Overview site
  - https://getdnsapi.net
  - Downloads and documentation available
  - https note – best with DANE TLSA

Dependencies

- Are linked outside the build tree, with configure finding them
- We strive to minimize them
- Current set
  - libldns and libunbound from Nlnet Labs (libldns requires openssl headers and libraries)
  - libexpat
  - libidn from FSF, version 1
- Packagers are at work - as of IETF 89
  - brew – formula exists
  - RHEL – in review
    - https://bugzilla.redhat.com/show_bug.cgi?id=1070510



Major features of this release

- Works with a variety of event loops, each built as a separate shared library
  - Details in wiki of the github repo
  - libevent
  - libev
  - libuv
- DNSSEC support fully implemented with well-tested Unbound at base
- Platforms as of IETF 89
  - RHEL/CentOS, MacOS
  - Soon to drop: FreeBSD, iOS (now rough but usable)
  - Windows, Android in view

DNSSEC in the API and implementation

- DNSSEC validation is off by default for stub mode (by design group consensus), but easy to turn on – use of extensions defined in API
  - dnssec_return_status
  - dnssec_return_only_secure
  - dnssec_return_validation_chain

- The API spec allows enabling DNSSEC on a per-request basis via setting the dnssec_return_status extension. For convenience, the implementation provides a means to enable this extension for every request in a given context
  - Documented in getdnsapi repo community wiki

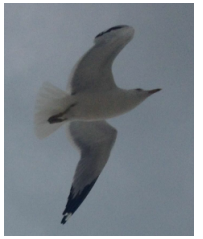
Coming soon

- Planned series of updates (0.1.1, 0.1.2, …) along with more platforms
  - Several fixes ready, plus a patch was contributed by community the day after the release.
- Language bindings
  - Soon after IETF 89: github/getdnsapi/getdns-python
  - TBA: Node.js
  - TBA: Java
  - Join in!
- Release 0.1.0 hasn't implemented all of spec yet
  - MDNS and NetBIOS namespaces – included in spec
  - DNS search suffixes – getdns_context_set_append_name, getdns_context_set_suffix – following DNSOP discussions…
  - GETDNS_TRANSPORT_TCP_ONLY_KEEP_CONNECTIONS_OPEN
  - Full set of EDNS(0) and OPT extensions
  - Full list in README

API examples – getdns_general()

- Some API examples are included for Extra Reading
- getdns_general is typical of public entry points
- Handle arbitrary resource record types

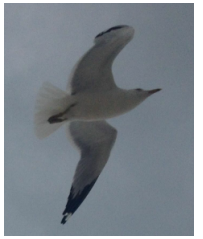
```
getdns_return_t
getdns_general(
  getdns_context_t       context,
  const char             *name,
  uint16_t               request_type,
  struct getdns_dict     *extensions,
  void                   *userarg,
  getdns_transaction_t   *transaction_id,
  getdns_callback_t      callbackfn
);
```



API examples - getdns_address()

- Handles requests by host name
- Always returns both IPv4 and IPv6 addresses
- Uses all name spaces from the context

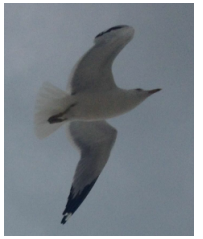
```
getdns_return_t
getdns_address(
  getdns_context_t        context,
  const char              *name,
  struct getdns_dict      *extensions,
  void                    *userarg,
  getdns_transaction_t    *transaction_id,
  getdns_callback_t       callbackfn
);
```



API examples - getdns_hostname()

- Accepts either IPv4 or IPv6 address

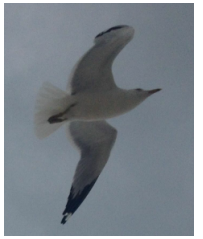
```
getdns_return_t
getdns_hostname(
  getdns_context_t        context,
  struct getdns_dict      *address,
  struct getdns_dict      *extensions,
  void                    *userarg,
  getdns_transaction_t    *transaction_id,
  getdns_callback_t       callbackfn
);
```



API examples - getdns_service()

- Returns the relevant SRV information

```
getdns_return_t
getdns_service(
  getdns_context_t        context,
  const char              *name,
  struct getdns_dict      *extensions,
  void                    *userarg,
  getdns_transaction_t    *transaction_id,
  getdns_callback_t       callbackfn
);
```



Python Bindings
Repo open before April, 2014

getdns-python(in progress)

- getdns already uses very Python-friendly data structures
- Goals
  - Provide easy-to-use interface to advanced DNS features for developers who are not necessarily DNS experts
  - remain as Pythonic as possible

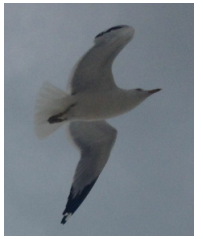
Basic use - getdns-python

- Currently:

```
import getdns
c = getdns.context_create()
ext = { "return_both_v4_and_v6" : \
   getdns.GETDNS_EXTENSION_TRUE, "dnssec_return_status" : \
   getdns.GETDNS_EXTENSION_TRUE }
getdns.address(c, "www.google.com", getdns.GETDNS_RRTYPE_A, ext)
```

- Soon:

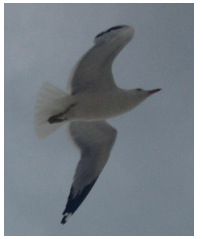
```
import getdns
c = getdns.Context()
ext = { "return_both_v4_and_v6" : \
   getdns.GETDNS_EXTENSION_TRUE, "dnssec_return_status" : \
   getdns.GETDNS_EXTENSION_TRUE }
c.address("www.google.com", getdns.GETDNS_RRTYPE_A, ext)
```



sync vs. async

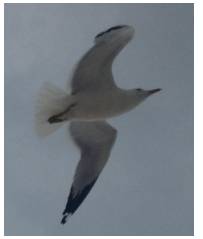
- Same basic interface, can make async calls by providing a callback function and optionally a transaction_id to identify the particular query

```
c.address("www.google.com", getdns.GETDNS_RRTYPE_A, ext, \
                            reply_processor)
```



Data structures

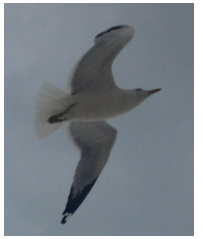
- Inputs to queries are
  - strings
  - Dictionaries
  - lists
- Query replies are basically JSON documents (we haven't yet run the format through a JSON validator yet, though …)
- We expose standard and complete RDATA returned by the API
- Complete set of getdns constants and return types
- Throw Python exceptions on errors
  - Currently generic exception
  - Will be adding getdns exception class



What replies look like

- `"replies_tree"` dictionary element from query response

```
"replies_tree": [
{
# This is the first reply
"header": { "id": 23456, "qr": 1, "opcode": 0, ... },
"question": { "qname": <bindata for "www.google.com">,
      "qtype": 1, "qclass": 1 },
"answer": [ {
     "name": <bindata for "www.google.com">,
     "type": 1,
     "class": 1,
     "ttl": 300,
     "rdata":
         { "ipv4_address": <bindata of 0x0a0b0c01>
           "rdata_raw": <bindata of 0x0a0b0c01>
         }
     }
],
```



Replies_tree queried from Python(cont'd)

- Usage from Python

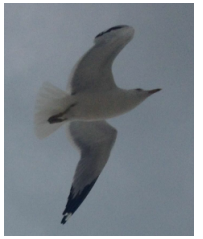
```
import getdns
c = getdns.context_create()
ext = { "return_both_v4_and_v6" : \
        getdns.GETDNS_EXTENSION_TRUE, "dnssec_return_status" : \
        getdns.GETDNS_EXTENSION_TRUE }
ret = getdns.address(c, "www.google.com", getdns.GETDNS_RRTYPE_A, ext)

print ret['replies_tree'][0]

 {'answer':
 {'rdata': {'ipv4_address': '74.125.131.104'}, 'type': 1,
 'class': 1, 'name': '\www.google.com', 'ttl': 300}}

print ret['replies_tree'][0]['answer']['rdata']['ipv4_address']

 74.125.131.104
```



Questions?

Most answers will be found at

getdnsapi.net

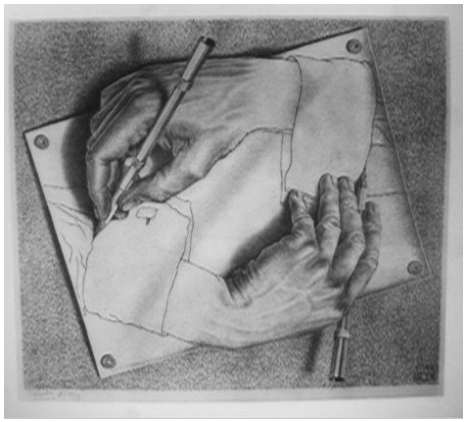
and

github.com/getdnsapi

Backup Material

One API, two modes

- Stub resolver
  - Often implemented via local library (e.g. libresolv)
  - Provides entry points for applications (e.g. gethostbyname)
  - Relies on a recursive name server
  - May not cache, but may implement e.g. single local cache
- Recursive Resolver
  - Typically receives DNS requests via wire protocol
  - Iterates on behalf of clients
  - Typically leverages caching
- getdns-api context controls which of these (2 modes)
- When DNSSEC is enabled for stub mode, the stub can iterate just DNSSEC validation on its own behalf



Stub resolver inDNS ecosystem