Discovering Provisioning Domain Names and Data

draft-ietf-intarea-provisioning-domains-00

P. Pfister, **E. Vyncke,** T. Pauly, D. Schinazi, M. Keane

The purpose of this draft is to:

1. Identify Provisioning Domains (PvDs).

[RFC7556] *Provisioning Domains (PvDs) are consistent sets of network properties that can be implicit, or advertised explicitly.*

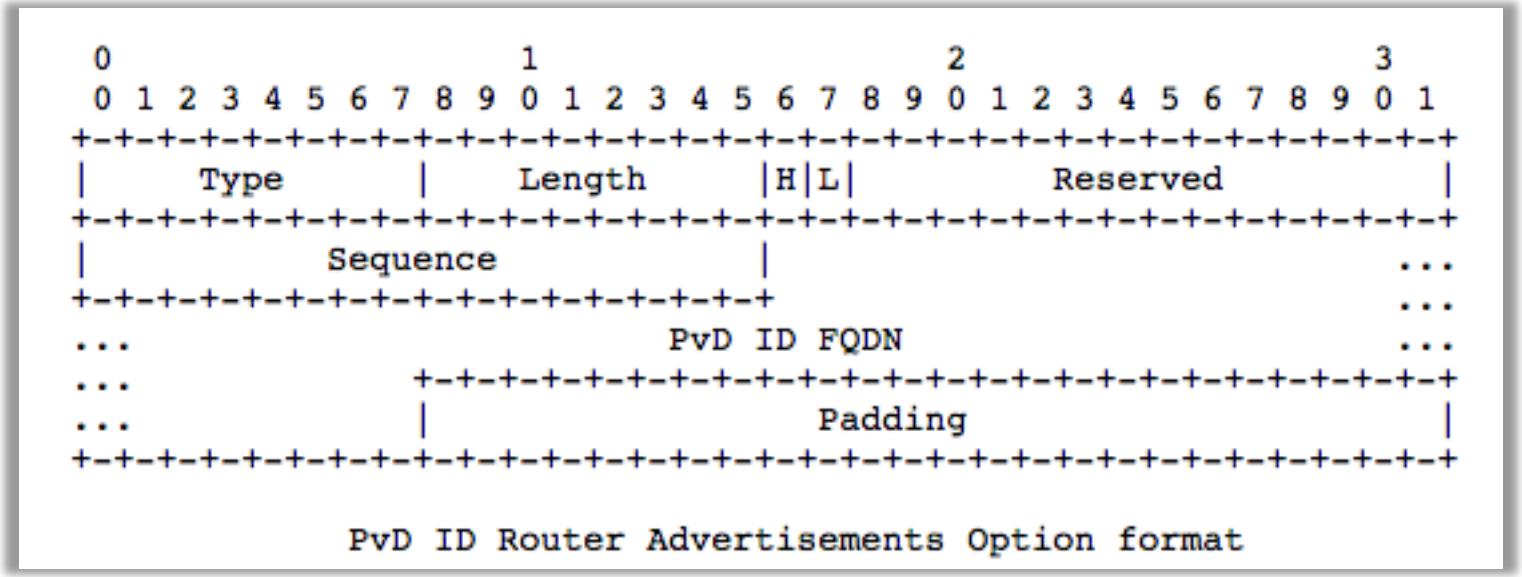
Differentiate provisioning domains by using FQDN identifiers.

2. Give PvD Additional Information.

Name, characteristics, captive portal, etc…

Step 1: Identify PvDs

With the PvD ID Router Advertisement Option

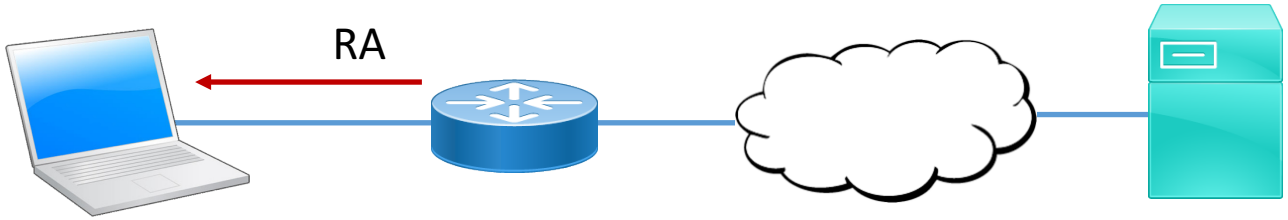


- At most **one occurrence in each RA**.
- **PvD ID is an FQDN** associated with options in the RA**.**
- **H bit** to indicate **Additional Information is available with HTTPS**.
- **L bit** to indicate the **PvD has DHCPv4 on the link**.
- Seq. number used for **push-based refresh**.

Step 1b: Identifying PvD (Cont.)

- Information in a RA without PvD ID is linked to an implicit PvD (identified by interface & link-local address of router)
- Option in RA can change of PvD when they are received in a RA with a different PvD ID
- DHCPv6 information MUST be associated to a PvD ID received on the same interface from the same link-local address

Step 2: Get the PvD Additional Data

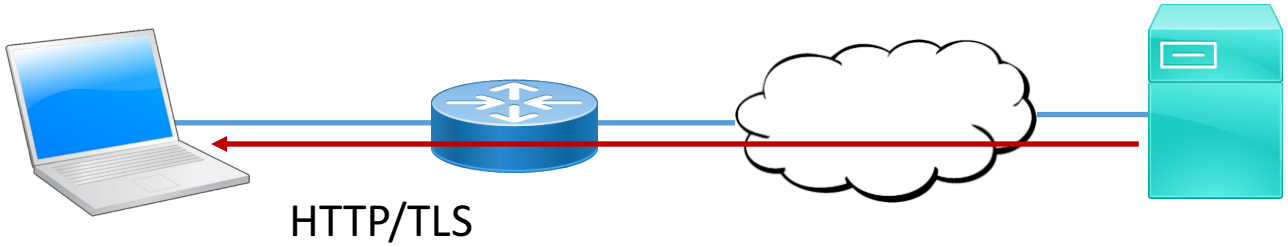


When the H bit is set:

GET https://<pvd-id>/.well-known/pvd

Using network configuration (source address, default route, DNS, etc…)
associated with the received PvD.

Step 2: Get the PvD Additional Data



When the H bit is set:

GET https://<pvd-id>/.well-known/pvd

Using network configuration (source address, default route, DNS, etc…)
associated with the received PvD.

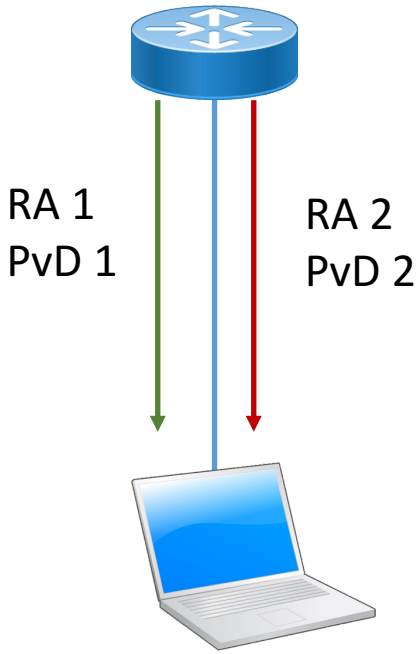
Step 2: Get the PvD Additional Data

```
{
  "name": "Foo Wireless",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes" : ["2001:db8:1::/48", "2001:db8:4::/48"],
  "localizedName": "Foo-Hôtel à Paris Wifi",
  "dnsZones": ["example.com","sub.example.com"];
  "characteristics": {
        "maxThroughput": { "down":200000, "up": 50000 },
        "minLatency": { "down": 0.1, "up": 1 }
  }
}
```

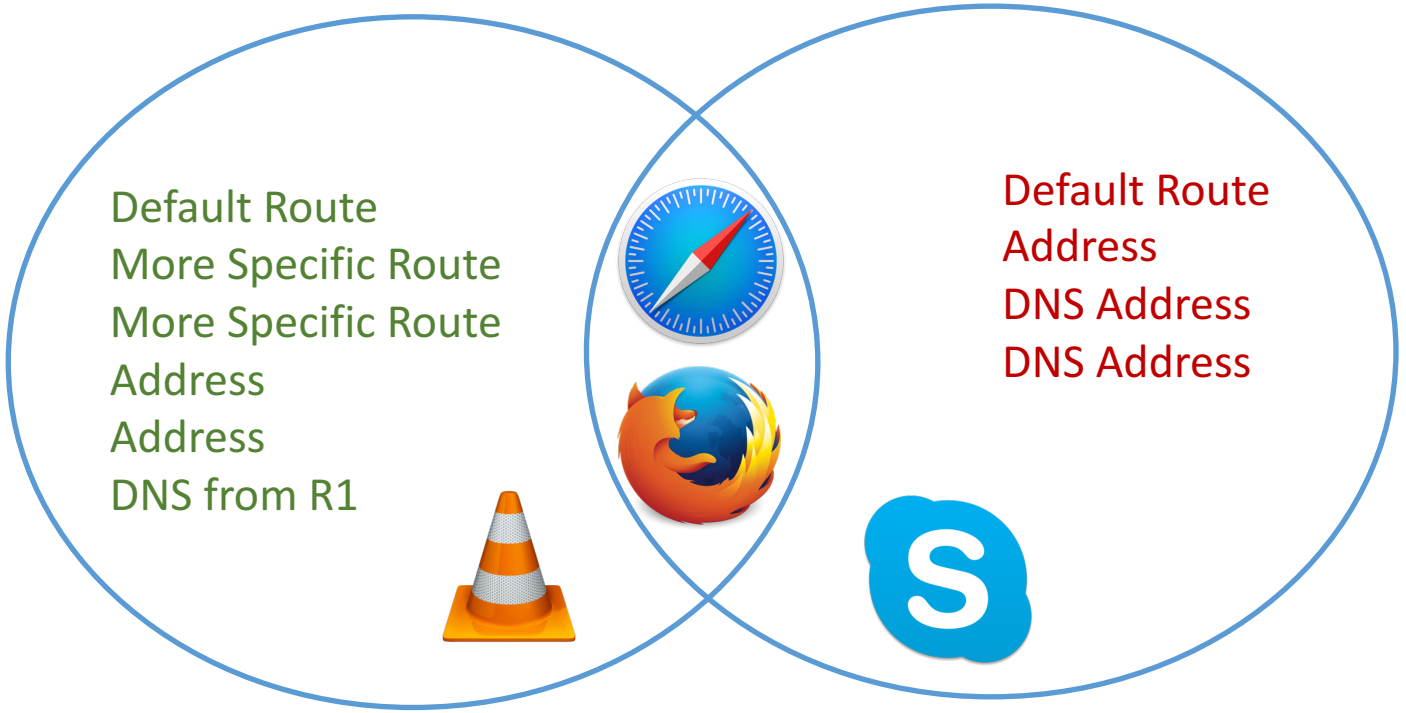
Some other examples (see also https://smart.mpvd.io/.well-known/pvd) :

```
noInternet : true,
metered : true,
captivePortalURL : "https://captive.org/foo.html"
```

Step 3: Host behavior



Hosts and applications **behave according to existing specs** on one or more PvDs.



Changes in Document

- IANA consideration follows RFC 8126
- **PvD ID FQDN encoded per RFC 1035**
- “dnsZones” added back
- **Privacy section added**
- Aiming “standard track”

PvD ID FQDN encoded per RFC 1035

- Before the PvD was plain ASCII
  - Could have different representation

- Now, RFC 1035 encoding is used

  The FQDN used as PvD ID encoded as described in Section 3.1 of RFC1035. Note that for simple decoding, the domain names MUST NOT be encoded in the compressed form described in Section 4.1.4 of RFC1035. This encoding is the same as the one used in RFC8106. The encoding MUST end with a null (zero-length) label.

What about Privacy ?

- The main concern is the OPTIONAL fetch over HTTPS of additional information from a server
- But,
  - This server is in the same management domain as DHCP, AAA or DNS
  - Not over the “wild” Internet
  - Better to discover captive portal via PvD additional information than a blind request to http://captive.example.com/hotspot-detect.html (default behavior of most current OS)
- Section 6 is now the privacy consideration

Implementation status

Linux - https://github.com/IPv6-mPvD

- pvdd: A Daemon to manage PvD IDs and Additional Data
- Linux Kernel patch for RA processing
- iproute tool patch to display PvD IDs
- Wireshark dissector
- RADVD and ODHCPD sending PvD ID

Implemented in one commercial vendor router

Next steps

- Feedback, suggestions, … are welcome

- If WG consensus, request IANA for an Neighbor Discovery option
  - Section 13 of RFC 4861 “reclaimable in future”
  - Currently using 253 “experimental”
  - **Several implementations => need to have a code ASAP to ensure interop**