DNS-SD Data Block

An encoding for non-IP transports: DNS-SD service information over Bluetooth LE, NFC and similar channels

`draft-kennedy-dnssd-data-block-02`

Smith Kennedy · 16 November 2026

THE VALUE

DNS-SD is widely used

- Widely deployed for service advertisement on IP networks
  - Printers, media servers, file sharing and many other types of service
- DNS-SD is two things
  - **A set of protocols** that advertise and discover services
  - **A large and easily extensible suite of service names:** many registered and tied to IETF application protocols (e.g. **_ipp._tcp**), others defined privately or ad hoc
- Its value comes as much from the size and flexibility of the suite of service names as from the protocols that carry them
- **This proposal extends the reach of those service names to non-IP transports**

THE PROBLEM

Where DNS-SD doesn’t reach

- DNS-SD works well when client and service share a segment (mDNS), or the network has infrastructure DNS-SD
- Two common cases fall outside that
  - **Segmented LANs:** routable, but mDNS doesn’t span the segments, or multicast is disabled or unreliable, and there’s no infrastructure DNS-SD
  - **Peer-to-peer links:** DNS-SD works only after connecting, so the user must connect before knowing what’s offered
- **DDB complements infrastructure DNS-SD such as ULD**
  - Carries the same DNS-SD information over a non-IP channel such as Bluetooth LE or NFC
  - Where DNS-SD over IP is available, it remains the preferred path

THE PROBLEM

The missing payload

- Discovery technologies reserve a payload slot keyed by an organization identifier; that organization defines its format
- **Bluetooth SIG:** Transport Discovery Service
  - AD Type **0x26**; each Organization Block has a Transport Data field
- **NFC Forum:** Verb RTD
  - Context Identifier **0x02** is assigned to “IETF IAB / IANA”, intended for DNS-SD
  - The Descriptor Data layout was left for the IETF
- For DNS-SD, the payload is still undefined. **DDB is intended to fill this gap.**

THE PROBLEM

Scope

- Example scenarios
  - Office printer on a segmented network advertises IPP over Bluetooth LE (TDS)
  - Hotel-room TV answers an NFC tap with its streaming services and carriers
- In scope
  - One encoding of DNS-SD service information
  - Rules to map it to and from PTR, SRV and TXT
- Not in scope
  - Encoding of the encapsulating technology
  - Relaying DDBs over IP, or any change to DNS-SD or mDNS
  - Trust: a DDB is a discovery aid only

CONSTRAINTS

Size constraints

SPACE A DDB CARRIER LEAVES FOR THE DDB, IN OCTETS



- The encoding has to fit the smallest DDB Carriers
  - A minimal service description needs to fit in about 26 octets
  - Richer descriptions need extended advertising or a separate record, such as a media-type NDEF record

IN USE

Deployment contexts

- **Peer-to-peer:** browse, connect, resolve
  - A direct link comes up and mDNS works on it; the DDB helps the user decide before connecting
  - Service Name and Instance Name are enough; TXT Data helps selection
- **LAN:** no resolution afterwards
  - mDNS doesn’t span the segments, so the DDB is the client’s only DNS-SD information
  - Include a unicast-resolvable Hostname, the Port if not the well-known one assigned to that service name, the TXT keys if needed to use the service, and the Domain if in unicast DNS-SD
- A sender can’t tell which context applies; a unicast-resolvable hostname serves both

IN USE

Mapping back to DNS-SD

- The Service Instance Name is the identity anchor
  - The PTR target a DNS-SD browse would return
- The other fields carry SRV and TXT information directly
  - **PTR:** Instance Name + Service Name + Domain (“local” if absent); each subtype is a subtype PTR
  - **SRV:** Port (or the assigned port) + Hostname; one SRV per DDB, priority and weight 0
  - **TXT:** TXT Data verbatim, plus a **UUID=…** key/value pair rebuilt from the binary UUID field
- Where DNS-SD is reachable, the client resolves that name as usual: **DNS-SD stays authoritative**

FOR DISCUSSION

External dependencies

Both encapsulating technologies already have a slot for the DDB

- **Bluetooth SIG:** DDB in the TDS Transport Data field
  - Would need an IETF Organization ID from the Bluetooth SIG
- **NFC Forum:** DDB in the Verb Descriptor Data field
  - Context ID 0x02 is already assigned to “IETF IAB / IANA”

AFTER ADOPTION

To discuss after adoption

-02 proposes answers to these; they are for the WG to debate once the work is adopted.

- The encoding: TLV (as in -02), CBOR or DNS wire records
- Compact forms for Service Name and Instance Name
- The field set, and which fields are required
- Registry policy for new field types
- Security and privacy details

FOR DISCUSSION

Questions and next step

1. Is DNSSD the right home for this work?
2. Offer only, or should a DDB also express a query?
3. Which fields are missing for real deployments?
4. Who would review the draft? **DNS-SD knowledge is enough; no Bluetooth or NFC expertise is needed**
5. Who would implement or test it?

NEXT STEP

Request a call for adoption

CONTACT

Smith Kennedy
smitty.standards@gmail.com
dnssd@ietf.org

`datatracker.ietf.org/doc/draft-kennedy-dnssd-data-block`

Thank you.

BACKUP

Wire format

One Version octet, then a sequence of TLV fields. Receivers skip any Type they do not recognize.

Version 1 octet, 0x01	**Type** 1 octet	**Length** 1 octet, or 0xFF + 2	**Value** Length octets	**Type** …	… more TLVs

SMALLEST CONFORMANT DDB: IPP, NAME INHERITED FROM THE DDB CARRIER

01 Version	**01 09** Service Name	**5F 69 70 70 2E 5F 74 63 70** “_ipp._tcp”	**02 00** Instance, inherited	**14 octets** total

0x00 is a one-octet pad, only where a transport’s framing needs it.

Version changes only for incompatible changes; new fields get new Types.

Strict: a repeated Type or an invalid value discards the whole DDB.

BACKUP

Field types

Type	Field	Status	Notes
0x01	Service Name	Required	e.g. “_ipp._tcp”; sent first
0x02	Instance Name	Required	Length 0: inherited from the DDB Carrier
0x03	TXT Data	Optional	TXT RDATA, verbatim
0x04	UUID	Optional	Binary form of the TXT “UUID” key
0x05	Domain	Optional	Absent means “local”
0x06	Port	Optional	Absent means the IANA-assigned port
0x07	Subtype List	Optional	Length-prefixed subtype labels
0x08	Hostname	Recommended	SRV target, as in the DNS-SD record

0x09 to 0xEF: Specification Required, optional fields only · 0xF0 to 0xFE: Private Use · 0xFF: Reserved

BACKUP

Encoding alternatives

	TLV (the draft)	DNS wire records	CBOR
Per-field overhead	2 octets, fixed	RR headers, names	Key + header, varies
Parser	Byte arithmetic	DNS message parsing	CBOR decoder plus a deterministic profile
TXT data	Carried verbatim	Native	Re-encoded
Today’s flat field set	Good fit	Heavier than needed	No clear size gain
Type codes	Own 1-octet registry	RR TYPEs, 16-bit	Map keys
Position	v1 baseline	Not pursued	Future candidate

CBOR is a real option, not a strawman. The question is timing: a simple interoperable baseline first, then revisit with implementation data.

BACKUP

Compact encoding

- Raised on the list: very compact DNS-SD descriptions for length-limited frames
  - Numeric codes (as in draft-ietf-core-uri-path-abbrev) or name prefixes
- The test is whether a field is user visible
  - **Service Name:** rarely shown; a code, prefix or Bloom filter could work
  - **Instance Name, human-readable:** shown in browse UIs; truncated, every printer of a model looks the same, e.g. “HP Color LaserJ”
  - **Instance Name, machine-generated:** IDs like **1BCAF22BFDBEABDF-0000000078994672** aren’t meant to be read by users; could they shrink?
- -02 already avoids sending the Instance Name twice when the DDB Carrier has it

BACKUP

Changes in -02

- Retitled “An Encoding for Non-IP Transports”, with a scope statement and its relationship to RFC 7558
- Fields
  - Instance Name required; Length 0 inherits the name from the DDB Carrier
  - Hostname recommended; no Port field means the IANA-assigned port
  - UUID is a compact form of the TXT “UUID” key
- Decoding
  - Strict: duplicates, invalid values or non-canonical lengths discard the DDB
  - Domain and Hostname encoded exactly as in DNS-SD records
- Also new: media type **application/dnssd-ddb**, a deployment contexts appendix, an expanded security section

BACKUP

Security and privacy

- **Spoofing:** anyone in range can send any DDB; trust comes later, over the application protocol
- **TLS names:** a valid certificate proves the name, not that the server is the device nearby
- **Relay:** receiving a DDB doesn’t prove the sender is close; it can be relayed or replayed
- **Privacy:** radio reaches people mDNS never would; RFC 8882 applies, so minimize what is sent
  - A stable UUID or a personal name lets the device be tracked; omit or rotate them
- **Duplicates:** a repeated Type discards the DDB, so no two parsers read it differently