IDN in RPP

Maarten Wullink (SIDN Labs), Pawel Kowalik (DENIC)

IETF 127, San Francisco

IDN in RPP

- The registry needs both forms: the A-label for the DNS, the U-label for validation
- Domain Create based on abandoned **draft-ietf-eppext-idnmap-02**
- Which labels are valid is registry policy, expressed as a **Label Generation Ruleset (LGR)**
- Each draft covers one layer, so model, transport and format stay separate:
  - data-objects: what an IDN is in the data model
  - core: how an IDN appears in HTTP and in URLs
  - json: how an IDN is encoded and validated in JSON
  - idn-variants* extension: variant handling beyond the base model

*= I-D not yet published

Data model (data-objects)

Element	Card.	Mutability	Rule
name	1	create-only	Fully qualified domain name; an IDN is carried as its A-label (ACE encoded)
uName	0-1	create-only	Unicode, NFC; must convert to exactly name
lgr	0-1	create-only	IANA published LGR [1]; required when uName is present

- On create the server validates the name (RFC 5891, Section 4.2) and returns an error for a code point not in the LGR, or for a uName that does not map to name
- The Availability operation takes uName and lgr as additional parameters
- Discovery lists the supported LGRs per TLD and the default LGR

[1] https://www.iana.org/domains/idn-tables

Discovery representation (json)

- List of support IDN Label Generating Rules (LGR)
  - At most 1 default LGR per TLD

```
{
    "@type": "discovery",
    "tlds": ["example"],
    "idnLgr": [
        {
            "@type": "discoveryIdnLgr",
            "tld": ["example"],
            "name": "example-latn-1.0",
            "url": "https://www.iana.org/domains/idn-tables/tables/example_latn_1.0.txt",
            "default": true
        }
    ]
}
```

HTTP mapping (core)

- An IDN used as the `{id}` path segment **MUST** be its A-label
- Why: the URL stays a valid URI without percent-encoding, and it identifies the DNS name, not a Unicode rendering of it

Example using domain name bücher.example

```
GET /domainNames/xn--bcher-kva.example HTTP/1.1
```

IDN on create

```
POST /domainNames HTTP/1.1
Content-Type: application/rpp+json

{
  "@type": "domainName",
  "name": "xn--bcher-kva.example",
  "uName": "bücher.example",
  "lgr": "latn-1.0",
  "registrant": { "@type": "contact", "id": "JD-1234" },
  ...
}
```

- The URL carries no name: create is `POST /domainNames`, the name is in the body
- `name` is the A-label, `uName` the U-label, `lgr` the LGR used for validation
- The server **MUST** reject a `uName` that is
  - Not NFC,
  - Does not convert to `name`
  - `Includes` a code point outside the LGR

IDN in responses that contain the domain name

- Any response that returns the Domain Name Data Object (for example Create, Read, Update) carries `name`, and for an IDN also `uName` and `lgr`
- `uName` and `lgr` are read-only; clients use `name` for identity and comparison
- The `Location` header is a URL, so it uses the A-label

```
HTTP/1.1 201 Created
Location: https://rpp.example/rpp/domainNames/xn--bcher-kva.example

{
  "@type": "domainName",
  "name": "xn--bcher-kva.example",
  "uName": "bücher.example",
  "lgr": "latn-1.0",
  "status": [ { "@type": "domainStatus", "label": "ok" } ],
  ...
}
```

Other requests use the A-label only

Request	Where the name appears
Read	`GET /domainNames/{id}`
Update	`PUT` or `PATCH /domainNames/{id}`; the body has no name elements, they are create-only
Delete	`DELETE /domainNames/{id}`
Transfer, renew, restore	`/domainNames/{id}/processes/...`
Availability	`GET` or `HEAD /domainNames/{id}/availability`; optional uName and `lgr` query parameters
Headers	`Location` and `Link` (`rel="rpp-process"`) carry URLs

The A-label identifies the resource. The U-label travels only as data: in the create request and in responses, plus the optional availability parameter.

IDN variants extension (WIP*)

- Base specifications know only the A-label (`name`), the U-label (`uName`) and the LGR (`lgr`); no variant concepts
- On create the server checks, using the LGR, whether the name is a variant of an existing name; allowed only for the same registrant
- Variants form a **bundle**; it is not a data object and each name stays its own Domain Name Data Object
- Client may request list of possible variants for IDN and their availability status?

```
{ "@type": "domainVariants", "lgr": "latn-1.0", "variants": [
    { "name": "buecher.example", "uName": "buecher.example", "status": "registered" },
    { "name": "bucher.example",  "uName": "bucher.example",  "status": "available" } ] }
```

*=This is not yet a published ID

Name forms in URLs

Option	Example path	For	Against
A. A-label only (current core)	`/domainNames/xn--bcher-kva.example`	One identifier per resource; valid URI; matches the DNS; simple caching, logging and access control	Unreadable URLs; clients must convert U-label to A-label
B. U-label only	`/domainNames/b%C3%BCcher.example?lgr=example-latn-1.0`	Readable; no client conversion	Needs normalisation and mapping rules; LGR as request parameter; percent-encoding
C. Both accepted, A-label canonical	both paths address the same resource	Convenient; fits the idea of reaching a domain by either form	Two URLs per resource; server must map U to A HTTP caches may treat these as different URI strings Which URL should the server return in Location headers?

Whatever is chosen must be consistent across URLs, query parameters, Header fields, JSON

Open points and feedback wanted

- URLs: A-label only, U-label only, or both with the A-label as canonical form?
- Do we need Variant support?
  - If yes:
    - do we also include support for bundles?
    - How handle availability check for variants?
- Should the Host Data Object support uName as well?

Questions and comments welcome, on the RPP list and on GitHub


SIDN.nl


@SIDN


SIDN

Thank You

www.sidnlabs.nl | stats.sidnlabs.nl

SIDN LABS