
ICANN
COMMUNITY FORUM
64
KOBE
9–14 March 2019

How it Works: RDAP

Eduardo Alvarez, ICANN
Gustavo Lozano, ICANN

ICANN 64

09,10 March 2019

Agenda

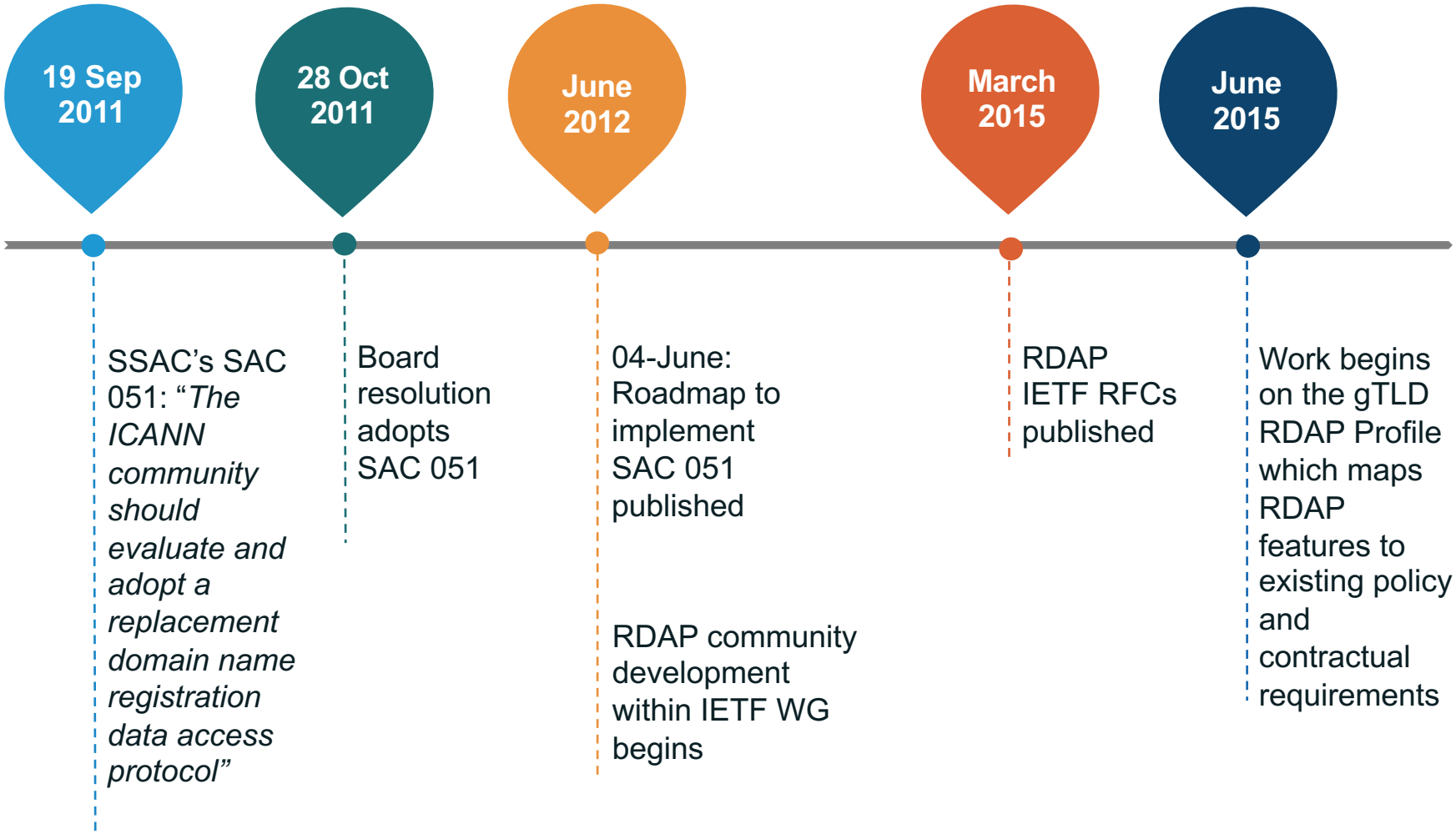
- Introduction
- RDAP Basics
- RDAP Queries and Responses
- RDAP Features and Concepts
- Differentiated Access
- Future and Ongoing Work

Introduction

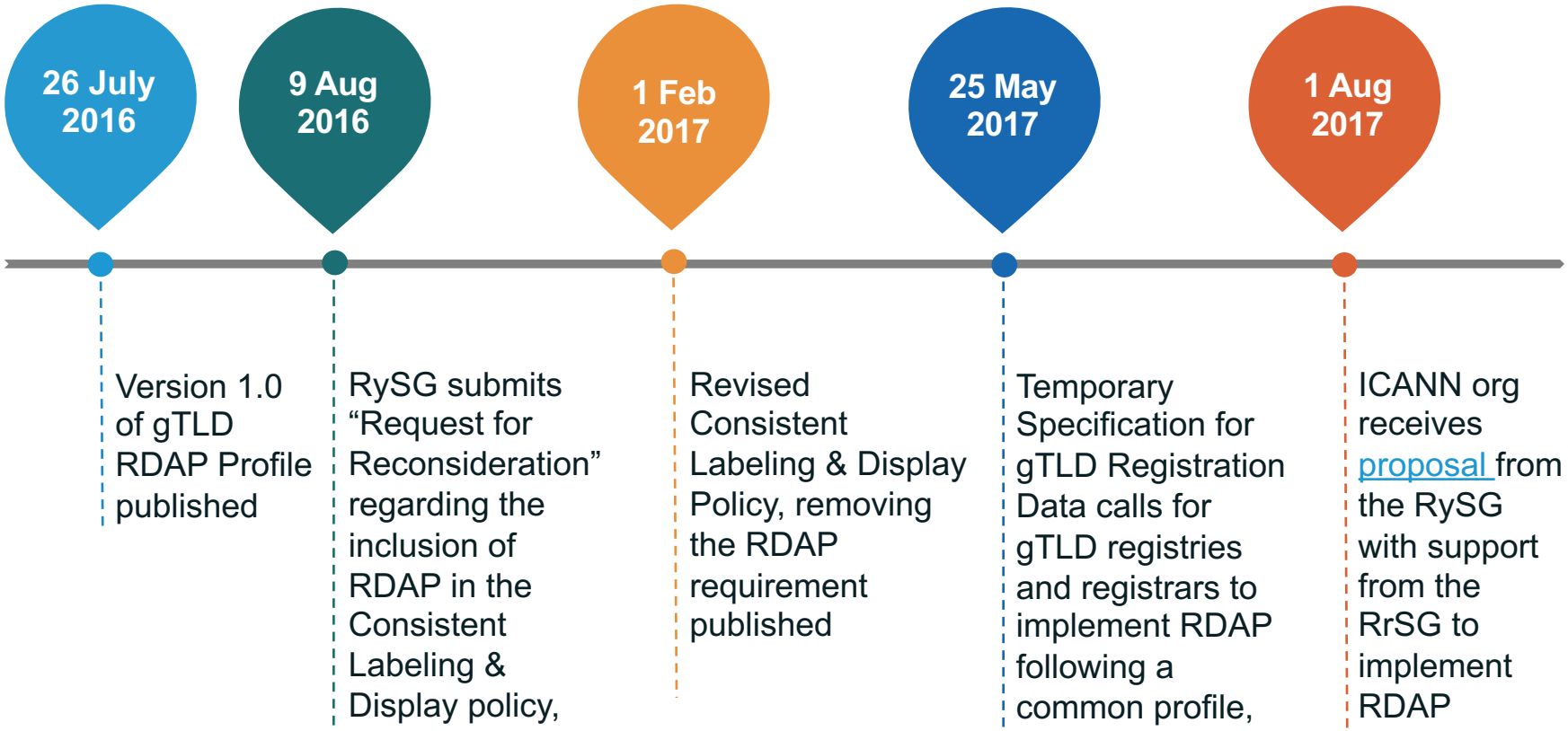
Issues with WHOIS (Port-43)

- No standardized format
- Lack of Support for Internationalization
- Unable to authenticate and thus provide different outputs depending on the user
- Lookup only; no search support
- Lack of standardized redirection/reference
- No standardized way of knowing what server to query
- Insecure
  - Cannot authenticate the server
  - Cannot encrypt data between server and client

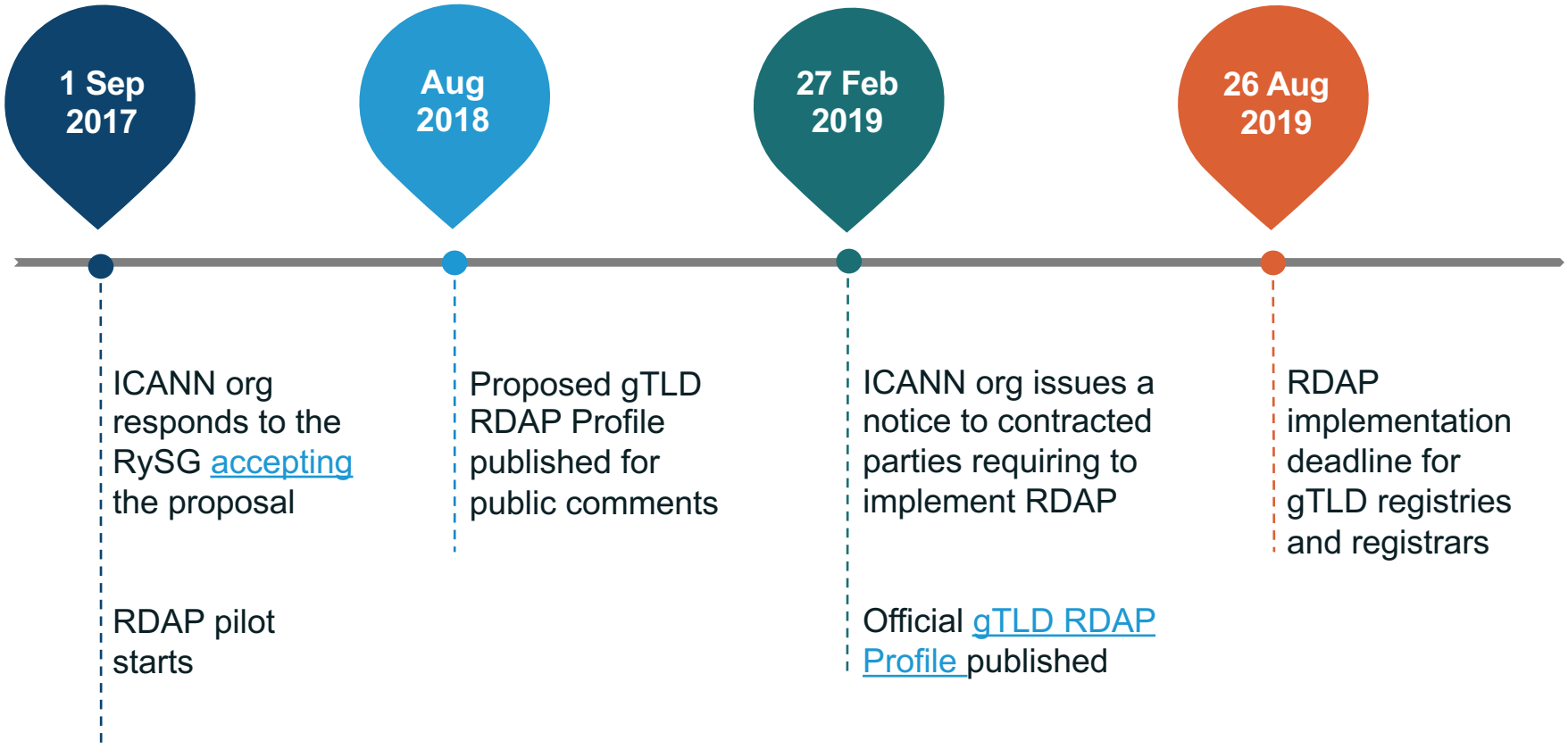
Chronology of RDAP Implementation [1/3]



Chronology of RDAP Implementation [2/3]



Chronology of RDAP Implementation [3/3]



RDAP Features [1/2]

RDAP is a protocol designed in the IETF (RFCs 7480 - 7484) to replace the existing WHOIS protocol and provides the following benefits:

- Standardized query, response and error messages
- Secure access to data
  - Over HTTPS
- Extensibility
  - Easy to add output elements
- Enables differentiated access
  - Limited access for anonymous users
  - Full access for authenticated users

RDAP Features [2/2]

- Bootstrapping mechanism to easily find the authoritative server for a given query
- Standardized redirection/reference mechanism
  - From a registry to a registrar
- Builds on top of HTTP, the well-known web protocol
- Internationalization support for registration data
- Enables searches for objects
  - Domain Names

Implementation Status

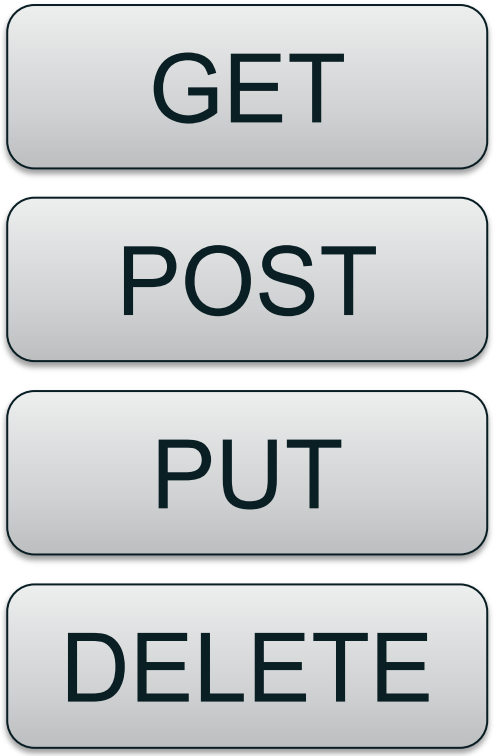
- Temporary Specification for gTLD Registration Data calls for gTLD registries and registrars to implement RDAP:
  - Following a common gTLD RDAP profile
  - SLA, and
  - Registry reporting requirements

- ICANN org and the contracted parties continue to work on finalizing the SLA, and registry reporting requirements.

RDAP Basics

RESTful Architecture and RDAP

- The RESTful architecture is a common architecture to define APIs for web services.
- RDAP is an API defined using the RESTful architecture.
- RESTful usually takes advantage of HTTP, leveraging libraries to easily interact with services defined under this architecture.



HTTP Basics

- The RDAP service is read-only. The core RDAP protocol uses the HTTP GET and HEAD methods only.
- HTTP contains mechanisms for servers and clients to authenticate each other.
- HTTP responses indicate the status code in the response to signal the request result. Some examples:

HTTP Status	Description
200	Ok
302	Found
400	Bad Request
404	Not Found
500	Internal Server Error
501	Not implemented

Restful Architecture and RDAP

- RDAP support two types of queries:
  - Lookup
  - Search

- Lookups find exact-match information of one object, e.g., a query for the domain name ”icann.org”.

- Searches are used to find information of one or multiple objects using search patterns that may contain an asterisk ('*') character to match zero or more trailing characters, e.g., domain names that start with “icann”.

Restful Architecture and RDAP

- The following lookup paths are defined in the specification:

domain/<domain name>

nameserver/<nameserver name>

entity/<handle>

ip/<IP address> or ip/<CIDR prefix>/<CIDR length>

autnum/<autonomous system number>

help

Restful Architecture and RDAP

- The following search paths are defined in the specification:

```
domains?name=<domain search pattern>
domains?nsLdhName=<domain search pattern>
domains?nsIp=<domain search pattern>
```

```
nameservers?name=<nameserver search pattern>
nameservers?ip=<nameserver search pattern>
```

```
entities?fn=<entity name search pattern>
entities?handle=<entity handle search pattern>
```

JSON – JavaScript Object Notation

- RDAP responses are provided using JSON (RFC 7159)
- JSON objects are unordered sets of name/value pairs.
  - E.g. { “name” : “value” }
- JSON defines the following data types for the values:
  - Number
  - String
  - Boolean (true or false)
  - Array
  - Object
  - Null

Example

```
{
  "title"     : "Introduction to JSON",
  "isbn".     : "123456-ABC",
  "author"    : "John Doe",
  "pages"     : 80,
  "published" : true,
  "tableOfContents" :
  [
    { "section" : "Introduction", "page" : 2  },
    { "section" : "Chapter 1",    "page" : 8  },
    { "section" : "Chapter 2",    "page" : 31 },
    { "section" : "Epilogue",     "page" : 57 },
    { "section" : "Appendix A",   "page" : 70 }
  ]
}
```

RDAP Common Data Structures

- The **Object Class Name** identifies the type of object being processed
  - Required in all RDAP response objects

```
"objectClassName" : "domain"
```

```
"objectClassName" : "nameserver"
```

```
"objectClassName" : "entity"
```

```
"objectClassName" : "ip network"
```

```
"objectClassName" : "autnum"
```

RDAP Common Data Structures

- The **RDAP Conformance** member indicates the list of specifications or definitions that the RDAP response is based on.
  - An array of strings
  - String values are defined in the IANA registry for RFC 7480
  - Always appears at the topmost JSON object of the response

```
"rdapConformance" :
[
  "rdap_level_0",
  "itNic_level_0"
]
```

RDAP Common Data Structures

- The **Links** array includes a list of links to related resources on the Internet.
  - An array of "link" objects with values as defined in RFC 5988.
  - Only the "href" value is mandatory

```
"links" : [
  { "value" : "https://rdap.example/rdap/domain/test.example",
    "rel"   : "self",
    "href"  : "https://rdap.example/rdap/domain/test.example",
    "type"  : "application/rdap+json"
  },
  { "rel"   : "related",
    "href"  : "http://nic.example/",
    "title" : "Dot Example Registry"
  }
]
```

RDAP Common Data Structures

- **Notices** and **Remarks -** used to provide general information about the RDAP service
  - Both take the form of an array of objects
  - "type" values per the IANA registry
  - “notices” may appear only once in main object of the response
  - "remarks" may appear multiple times across different objects

```
"notices" : [
  { "title"       : "Data has been truncated",
    "type"        : "object truncated due to authorization",
    "description" : [ "Private data has been omitted from the response.",
                      "Contact mail@registry.example for more details." ],
    "links"       : [ { "href"  : "http://www.rdap.example/notice.html",
                        "type"  : "text/html"  }
                    ]
  }
]
```

RDAP Common Data Structures

- The **Language Identifier**, can be used to inform the language used for the object class.
  - A single name/value pair with the language identifier as described in RFC 5646
  - May appear anywhere in an object class or data structure
  - The following example indicates the use of Mongolian language in Cyrillic script, as used in the region of Mongolia.

```
"lang" : "mn-Cyrl-MN"
```

RDAP Common Data Structures

- The **Events** member is used to represent events that have occurred on an instance of an object class
  - An array of objects
  - The "eventAction" and "eventDate" members are mandatory in every event
  - May include a "eventActor" and "links" members.

```
"events" : [
  { "eventAction" : "registration",
    "eventActor"  : "sample_registrar",
    "eventDate"   : "1999-12-31T23:59:59Z"
  },
  { "eventAction" : "expiration",
    "eventDate"   : "2019-12-31T23:59:59Z"
  }
]
```

RDAP Common Data Structures

- The **Status** data structure is used to indicate the state of a registered object.
  - An array of strings
  - List of allowed values is available in the "RDAP JSON Values" IANA registry (RFC 7483)

```
"status" : [
   "active",
   "client transfer prohibited"
]
```

RDAP Common Data Structures

- The **Port 43 WHOIS Server** data structure is the fully qualified host name or IP address of the WHOIS server where the containing object may be found.

```
"port43" : "whois.nic.example"
```

RDAP Common Data Structures

- The **Public IDs** shows the public identifiers of an object class.
  - An array of objects
  - Each object contains the following members:
    - type
    - identifier

```
"publicIds" : [
   { "type"       : "IANA Registrar ID",
     "identifier" : "9999"
   }
]
```

RDAP Queries and Responses

RDAP Lookup Segment Specification

- The resource type path segments for exact match lookup are

Domain	domain/<domain name>
Nameserver	nameserver/<nameserver name>
Entity	entity/<handle>
IP Network	ip/<IP address> ip/<CIDR prefix>/<CIDR length>
ASN	autnum/<autonomous system number>

Domain Queries

Domain query <baseUrl>/domain/<domain name>

- Used to identify a domain name and associated data referenced.
- The <domain name> is a fully qualified domain name.
  - A-label and U-label format are both supported
- Examples:
  - domain.example
  - 网站.域名
  - xn--fo-5ja.example

Domain Response

Member	Type	Description
handle	String	The unique identifier of the domain object response
ldhName	String	The domain name in LDH form
unicodeName	String	The domain with U-labels
variants	Object array	An array of objects, each with the following values: • relation • idnTable • variantNames
nameservers	Object array	Name
entities	Object array	Entity objects related to this domain object
network	Object array	The IP network for which a revers DNS domain is referenced

Domain Response

Member	Type	Description

secureDNS	Object	DNSSEC related info with the following members:
		• zoneSigned
		• delegationsigned
		• maxSigLife
		• dsData
		  • flags
		  • keyTag
		  • algorithm
		  • digest
		  • digestType
		  • events
		  • links
		• keyData
		  • flags
		  • protocol
		  • publicKey
		  • algorithm
		  • events
		  • links

Nameserver Queries

Nameserver query <baseUrl>/nameserver/<nameserver name>

- Used to identify a nameserver information query using a host name.
- The <nameserver name> is a fully qualified host name.
  - A-label and U-label format are both supported
  - Some examples:
    - ns1.example.com
    - ns1.xn--fo-5ja.example
    - ns1.网站.域名

Nameserver Response

Member	Type	Description
handle	String	The unique identifier of the nameserver object response
ldhName	String	The nameserver in LDH form
unicodeName	String	The DNS Unicode name of the nameserver
ipAddresses	Object array	An object array with the following members: • v6: as an array of strings with IPv6 addresses of the nameserver • v4: as an array of strings with IPv4 addresses of the nameserver
entities	Object array	Entity objects related to this nameserver object

Entity Queries

Entity query <baseUrl>/entity/<handle>

- Used to identify an entity (e.g., contact, registrar) information query using a string identifier.
- The <handle> represents an entity identifier.
  - Syntax is specific to the registration provider
- Some examples:
  - REGISTRAR1
  - 123456
  - 123456-EXAMPLE

Entity Response

Member	Type	Description
handle	String	The unique identifier of the entity
vcardArray	Object	A jCard (RFC 7095) with the entity's contact information
roles	String array	The relationship an object would have with it's closes containing object. • Role values defined in the IANA registry for RFC 7483
entities	Object array	Entity objects related to this entity object
asEventActor	Object array	Same as the "events" common data structure, denoting this entity as the event actor for the given events.
networks	Object array	ipNetwork objects related to this entity
autnums	Object array	autnum objects related to this entity

vCard and jCard

- A vCard is the digital representation of a business card.
- The vCard format is defined in RFC6350.
- The vCard format is widely supported in software.

```
BEGIN:VCARD
VERSION:4.0
FN:John Doe
N:Doe;John;;;Mr.
GENDER:M
ORG;TYPE=work:Example Corp
ADR;TYPE=work:;Suite 202;2562 Shobe
Lane;Greeley;CO;86311;United States
TEL;VALUE=uri;TYPE="work,voice";PREF=1:tel:+1-555-555-
555;ext=555
EMAIL;TYPE=work:john.doe@domain.example
END:VCARD
```

vCard and jCard

- Contact information is defined using jCard in RDAP.
- jCard is a JSON format for vCard data.

```
[ ["version", {}, "text", "4.0"],
  ["fn", {}, "text", "Joe User"],
  ["n", {}, "text",
    ["User", "Joe", "", "", ["Mr."]]
  ],
  ["kind", {}, "text", "individual"],
  ["adr",
    { "type":"work" },
    "text",
    [
      "","Suite 1234","2562 Shobe
Lane","Greeley", "CO","86311",
"United States"
    ]
  ],
  ["tel",
    { "type":["work", "cell"] },
    "uri", "tel:+1-555-555-4321"
  ],
  ["email",
    { "type":"work" },
    "text", "joe.user@example.com"
  ]]
```

Entity Response – jcard Example

Jcard Properties

- version
- **fn**
- kind
- lang
- **org**
- title
- role
- **adr**
- **tel**
- **email**
- lang
- geo
- gender
- …

"adr" Member Order

1. Post office box
2. Extended address
3. Street address
4. Locality (e.g. city)
5. Region (e.g. state or province)
6. Postal code
7. Country name

```
"vcardArray": [
  "vcard", [
     [ "version", {}, "text", "4.0" ],
     [ "fn", {}, "text", "John Doe" ],
     [ "org", {}, "text", "ICANN" ],
     [ "tel", {}, "uri", "tel:+1-123-456-7890" ],
     [ "adr", {}, "text", [
                "",
                "",
                "6 Rond-Point Schuman",
                "Brussels",
                "",
                "B-1040",
                "Belgium"
                ]
     ]
  ]
]
```

IP Network Queries

IP Network query

<baseUrl>/ip/<IP address>
<baseUrl>/ip/<CIDR prefix>/<CIDR length>

- Used to identify IP networks and associated data referenced using either an IPv4 or IPv6 address.
- Query targets the smallest IP network that encompasses the provided address in a hierarchy of IP networks
- The <IP address> may be IPv4 dotted decimal or IPv6 address (RFC 4291)
  - IPv6 recommended text representation per RFC 5952.
  - IPv6 zone_id not supported and must not be used (RFC 6874)
- CIDR notation address blocks prefix and length are defined in RFC 4632
- Some examples
  - 192.0.2.0
  - 192.0.2.0/24
  - 2001:db8::0

IP Network Response

Member	Type	Description
handle	String	The RIR-unique identifier of the network registration
startAddress	String	Starting v4 or v6 IP address of the network
endAddress	String	Ending v4 or v6 IP address of the network
ipVersion	String	the IP protocol version ("v4" or "v6")
name	String	identifier assigned to the network registration by the registration holder
type	String	RIR-specific classification of the network
country	String	2-character country code of the network
parentHandle	String	RIR-unique identifier of the parent network of this registration
entities	Object array	Entity objects related to this network

Autonomous System Number Queries

ASN query <baseUrl>/autnum/<autonomous system number>

- Used to identify Autonomous System number registrations and associated data referenced.

- The <autonomous system number> is an autonomous system number.
  - Format as specified in RFC 5396
  - Target of the query is the ASN block registration that includes the queried number.

Autonomous System Number Response

Member	Type	Description
handle	String	RIR-unique identifier of the ASN registration
startAutnum	Number	A number representing the starting number in the block of ASNs
endAutnum	Number	A number representing the ending number in the block of ASNs
name	String	Identifier assigned to the ASN registration by the registration holder
type	String	RIR-specific classification of the ASN
country	String	The 2-character country code of the ASN
entities	Object array	Entity objects related to this ASN

Other responses

Help query `<baseUrl>/help`

- Used to request information from the RDAP server, such as command syntax, terms of service, policies, etc.
  - Uses the ”notices” common structure

```
"notices" : [
  {
    "title"         : "Terms of use",
    "description"   : [
                      "This service is for informational purposes only.",
                      "Data is provided as is."
    ],
    "links"         : [
                      {
                        "href"  : "http://www.rdap.example/terms.html",
                        "type"  : "text/html"
                      }
    ]
  }
]
```

Other responses

- **Error responses** have a basic structure that allows to include information to describe the error.
  - *errorCode* value is based in the HTTP response code

```
{
   "errorCode": 400,
   "title": "Incorrect path segment",
   "description":
   [
      "Error processing this request.",
      "Please try again."
   ]
}
```

RDAP Features and Concepts

Extensibility in RDAP

- RDAP was defined with extensibility in mind.
- Several IANA Registries are used to easily extend the values defined in RDAP.
- Expert Review process:
  - Adds value to the RDAP related IANA Registries
  - Lightweight process that involves sending an email to IANA. Next an expert does a quick review and approves or rejects the addition.

Extensibility in RDAP

- The following types can be extended via the Expert Review process:
  - notice and remark type
  - status
  - event action
  - role
  - domain variant relation
  - event action
- The RDAP JSON Values registry is here:

  https://www.iana.org/assignments/rdap-json-values/rdap-json-values.xhtml

RDAP Extensions

- An RDAP extension augments the features of the RDAP protocol.
- An IANA Registry exists to document RDAP extensions. The purpose of the Registry is to allow implementers to know and understand about the extensions being implemented in the market place.
- The RDAP extensions registry is here:

  https://www.iana.org/assignments/rdap-extensions/rdap-extensions.xhtml

Bootstrapping

- RDAP includes a standard mechanism that allows a client to find the authoritative server for a domain name query
- RDAP specification explains how to form direct queries and basic search queries
- http://data.iana.org/rdap/dns.json

References from Registry to Registrar

- A thick registry holds all of the contact information related to domain names.
- In a thin registry the contact data is held by the registrar.
- With RDAP, a registry is capable of providing to the user a URL to the registrar's RDAP in order to obtain authoritative information maintained by the Registrar.

Registrar Bootstrapping

- The RDAP technical implementation guide has the following requirement:

> A registry server RDAP response to a domain query MUST contain a *links* object as defined in [RFC7483] section 4.2., in the topmost JSON object of the response. The *links* object MUST contain the elements *rel:related* and *href* containing the Registrar's RDAP URL of the queried domain object if the Registrar's RDAP URL has been defined.

- The RDAP pilot group asked ICANN to setup a (temporary) central repository for RDAP base URLs of gTLD Registrars

RDAP Object Tagging

- RDAP includes a method that can be used to identify the authoritative server for processing domain name, IP address, and autonomous system number queries.
- There is no such method for entities objects (e.g. contacts, registrars).
- This limitation exists because the identifiers bound to these queries are typically not structured in a way that makes it easy to associate an identifier with a specific service provider.

RDAP Object Tagging

- RFC 8521 defines a method for tagging objects allowing a client to identify the source of the information.
- A service provider tag is constructed by prepending a hyphen "-" to an IANA-registered string that identifies the RDAP authoritative source.
- The service provider registry is populated using the "First Come First Served" policy.
- https://www.iana.org/assignments/rdap-provider-object-tags/rdap-provider-object-tags.xhtml

Internationalization

- Internationalized domain names are supported in both the query and the response
- Internationalized contact information is also supported
- Contact information supports language tags in order to identify the language/script of data fields
- Replies are JSON formatted, which supports and by default requires UTF-8
- UTF-8 is a Unicode encoding. Unicode supports most of the world's writing systems.

Rate Limiting

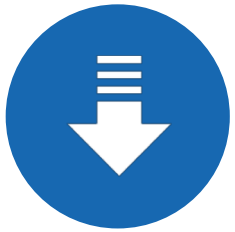
- Rate Limiting: Limiting the rate of traffic (usually through limiting the rate of requests) received by a server for a given client to mitigate the risk of denial of service attacks and data mining.
- Servers usually rate-limit the WHOIS service, and it's likely that rate limiting will be used in the RDAP service.
- Rate limit may have the effect of restricting access to registration data for legitimate users.
- RDAP through HTTP provides an status code to signal that the server is rate-limiting the connection.

HTTP 429 Too Many Requests

Differentiated Access

Authentication

- **Authentication:** verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a system.



For example:

When logging into a website, after the user provides their username and password, the website authenticates the user before providing access.

Authorization

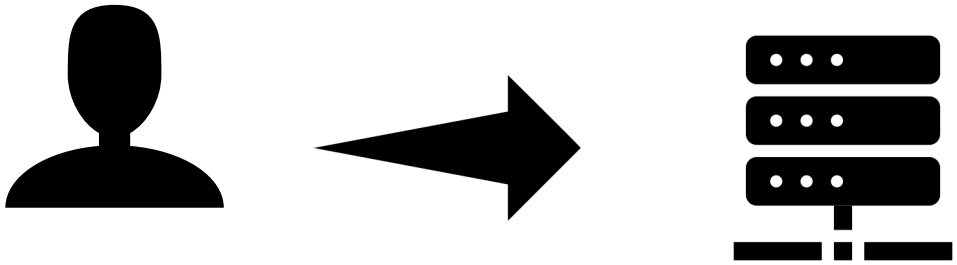
- **Authorization:** the granting or denying of access rights to a user, program, or process.

For example:

An authenticated user tries to access a file, the system authorize the user to access the file based on the predefined permissions.

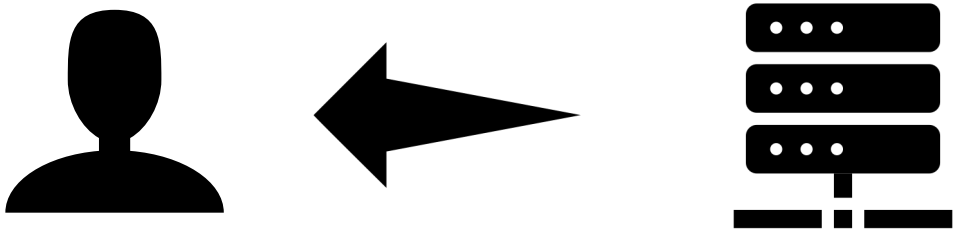
Authentication and Authorization

Authentication



Are you really who you say you are?

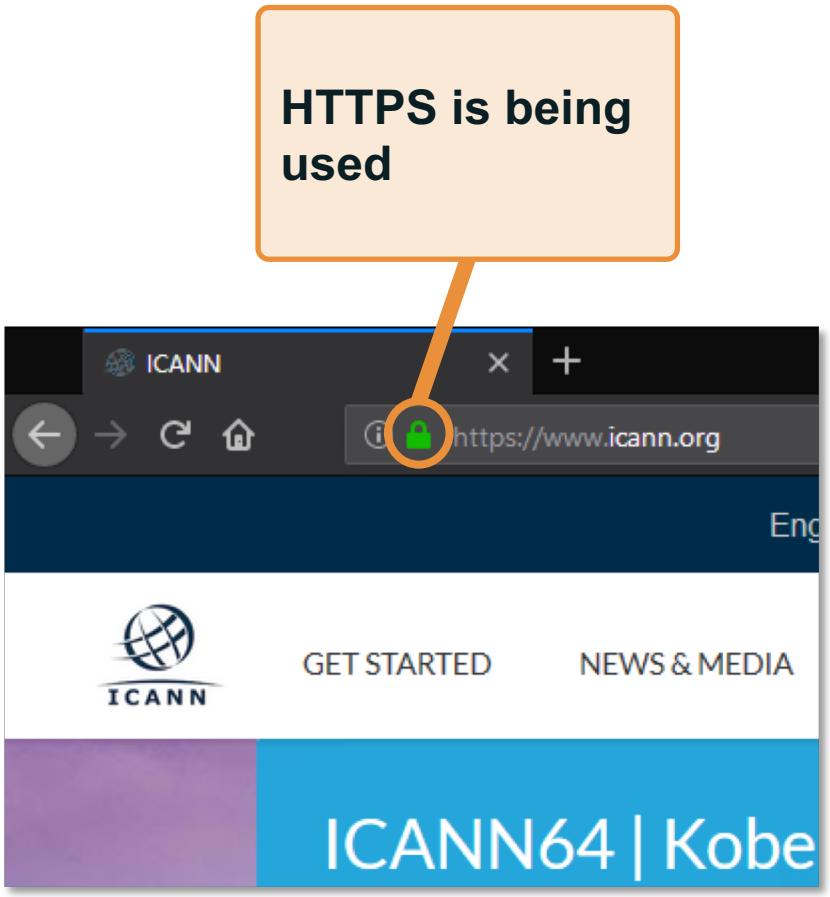
Authorization



Do you have permissions to access the resource that you are trying to access?

Data Confidentiality and Integrity

- Hypertext Transfer Protocol Secure (HTTPS) is the security protocol of choice on the Internet.
- HTTPS create an encrypted channel between the client and service provider.
- HTTPS uses Transport Layer Security (TLS) as the basis for providing the security services.



Data Confidentiality and Integrity

- TLS rely on digital certificates to authenticate the server and (optionally the) client.
- Certification authorities act as trusted third parties that certify the authenticity of a digital certificate.
- Digital certificates uses the standard x.509.

Data Confidentiality and Integrity

- Typically in a client / server interaction on the Internet, digital certificates signed by a CA are used to authenticate the server side.

- Client certificates can be used to authenticate the client. In this scenario, the user of username and password for authentication is not required.

Differentiated Access

- Differentiated access refers to showing different subsets of data fields based on the permissions of who is asking

- The Temporary Specification for gTLD Registration Data defines a minimum output and also requires providing access to further data on the basis of legitimate interest

- Further policy work/requirements have to be developed in order to have a Unified Access Model that would provide for this access in a consistent way in gTLDs

ICANN Org's RDAP Web Client

- ICANN org implemented an RDAP web client
  - Supports domain object lookups
  - Besides the standard bootstrapping, the client temporarily supports participants of the RDAP pilot
  - Supports authentication using:
    - X.509 digital certificates
    - OpenID Connect

https://rdap-client.viagenie.ca/

Future and Ongoing Work

gTLD RDAP Profile

- RDAP is a flexible protocol that allows implementers to choose from different features.

- The gTLD RDAP Profile is intended to achieve interoperability by defining the features to be implemented by gTLD registries and registrars on how to implement the RDAP service.

- The profile maps current policy requirements to the RDAP implementation with flexibility to incorporate future policy changes with minimal reengineering.

gTLD RDAP Profile

- A discussion group of gTLD registries and registrars (contracted parties) developed a proposal for a gTLD-RDAP Profile consisting of two documents:

  1) RDAP Technical Implementation Guide

  2) RDAP Response Profile

gTLD RDAP SLA and Registry Reporting

- ICANN is working with a discussion group of gTLD registries and registrars to finalize a gTLD-RDAP service-level agreement (SLA) and reporting requirements for registries (e.g., how many RDAP queries were received in a month)

EPDP [1/2]

- In May 2018, the ICANN Board adopted the proposed Temporary Specification for gTLD Registration Data.
- This is an interim measure to bring existing RDDS obligations in line with requirements of the European Union's General Data Protection Regulation (GDPR).
- This also triggered the GNSO Council to undertake a policy development process to confirm it, or not, as a Consensus Policy within 12 months.

EPDP [2/2]

- This policy development process is known as the Expedited Policy Development Process (EPDP) on the Temporary Specification for gTLD Registration Data.
- The EPDP team started their work in August 2018.
- On 20 February 2019, the EPDP Team submitted to the GNSO Council its Final Report.
- On 4 March 2019, the GNSO council approved the EPDP Final Report & Recommendations.

ICANN's TSG

- On December 2018 ICANN launched a Technical Study Group on access to non-public registration data.
- The TSG is exploring technical solutions built on RDAP for authenticating, authorizing, and providing access to non-public registration data for third parties with legitimate interests.
- Ultimate goal is building a proposal that can be considered for a Unified Access Model implementing the policy agreed by the community (e.g., EPDP).

Internet Engineering Task Force (IETF)

- *The mission of the IETF is to produce high quality, relevant technical and engineering documents that influence the way people design, use, and manage the Internet in such a way as to make the Internet work better.*

- *Any interested person can participate in the work, know what is being decided, and make his or her voice heard on the issue.*

- *The IETF make standards based on the combined engineering judgement of our participants and our real-world experience in implementing and deploying our specifications.*

Note: text from RFC 3935.

IETF's REGEXT Working Group

- The technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.).
- The REGEXT WG is the home of the coordination effort for defining standards to be used in the EPP and RDAP protocols.
- Registries and registrars actively participate in the REGEXT WG, and the continuous development of RDAP takes place in this working group.

Current Proposed RDAP Extensions

- Federated Authentication for the RDAP using OpenID Connect (draft-hollenbeck-regext-rdap-openid)
- RDAP Partial Response (draft-loffredo-regext-rdap-partial-response)
- RDAP Reverse Search Capabilities (draft-loffredo-regext-rdap-reverse-search)
- RDAP Query Parameters for Result Sorting and Paging (draft-loffredo-regext-rdap-sorting-and-paging)
- RDAP Search using POSIX Regular Expressions (draft-fregly-regext-rdap-search-regex)

Server and Client Implementations Available

- RDAP server open-source projects:
  - DNSBelgium
  - Red Dog
- RDAP client projects:
  - ICANN's prototype RDAP web client
  - CentralNIC
  - DNSBelgium
  - NicInfo
  - OpenRDAP
  - ARIN’s RDAP web client

Engage with ICANN

Thank You and Questions

Visit us at **icann.org**
Email: globalSupport@icann.org

@icann

linkedin/company/icann

facebook.com/icannorg

slideshare/icannpresentations

youtube.com/icannnews

soundcloud/icann

flickr.com/icann

instagram.com/icannorg