MARRAKECH – How It Works: Domain Name Registry Protocols
Sunday, March 06, 2016 – 13:30 to 15:00 WET
ICANN55 | Marrakech, Morocco

UNIDENTIFIED MALE: We're going to get started in just a second. We're just waiting to see if any more people come in. In the meantime, I'm just going to awkwardly stare at you all, so just relax and it's good. Bob?

STEVE CONTE: All right, we're going to go ahead and get started .Thank you all for coming in. It's halfway through day zero or day one, so I applaud you all. And how many of you all are now counting down until the meeting's over? Oh, just me. Oh, David too.

Thanks for coming back. This session is on domain registration protocols and I'm not going to speak too much. I'm going to hand it over to David Conrad, our CTO here at ICANN. So without further ado –

DAVID CONRAD: Hi. Yeah, it is working. Okay. I'm David Conrad. I'm ICANN's CTO, as Steve mentioned. Been with ICANN this time for about 18 months. Originally joined ICANN in 2005, ran IANA for a number of years and then escaped in 2010 with my life. But ICANN is so much fun, I had to come back for a second round and rejoined ICANN in August of 2014.

I'm not usually the person who does this presentation. Usually, Ed Lewis is the expert presenter on this session. Ed was not able to come to Marrakech this time so you all get stuck with me. I will try to do Ed justice but if there is any confusion, it's my fault, not Ed's.

This How It Works session is going to be on domain name registration protocols. This is a slight revision to a similar session that we gave in Dublin and in Buenos Aires. This one is to be more generalized towards other protocols, not just the DNS.

And actually, out of curiosity, how many people here would consider themselves with the DNS? Could you raise your hand? How about with the other registration protocols like RDAP and WHOIS and EPP? Anybody? Okay, so about half. Okay. How about things like data escrow and TMCH? Anybody got that? Oh, okay. Okay. So moving right along.

So we're going to be talking mostly about the registration ecosystem for domain names. There are a whole bunch of protocols that are involved and they are interacting at different places, be it the registry, the registrar, the registrant, end users, resellers, TMCH, the Trademark Clearinghouse, data escrow agents.

All of those, of course, operate over Internet protocols, IPv4 or IPv6. Almost all of the protocols that we'll be talking about are also over TCP. Not important. The only one that really isn't is UDP, is the DNS protocol itself which goes mostly over UDP.

We won't be talking about the protocols in-depth in this session. We're talking more about sort of the architectural levels and not going into bit formats or anything like that. So if that's interesting to you, we can talk about it later. That's not what this session's about. And if you are interested in that, then you're probably a nerd like me.

So ecosystem players, these are the folks that are involved and this is mostly just to sort of set the stage. You have the registrant which is the holder of the domain name registration for a domain name.

The registry, it's actually the database of the domain names and the registrants for those domain names. The registrar is the agent that acts primarily between the registrant and the registry.

It's not necessarily the only agent. There are re-sellers who re-sell registrar services. TMCH, the Trademark Clearinghouse, an institution that was created through the new gTLD program to deal with intellectual property rights. Data escrow agents, which provide third party backup of registry and registrar data.

And there are a whole bunch of other potential protocols that we could talk about, things like the privacy and proxy services and how those work. But we probably won't get into that in this session unless you really want to get into it.

And I should say if anyone has any questions at any time, feel free to interrupt me. I actually much prefer sort of an interactive session. But I'm happy to ramble on until people fall asleep.

The relationships associated with the top level domain registries is sort of depicted on that slide. You have the TLD registry that is interacted with by the registrars. Registrants interact with the registrars or through re-sellers. Pretty straight-forward, everyone is presumably used to these sets of relationships now.

On the other side, we have data escrow that interacts with the TLD registry and the Trademark Clearinghouse. Both of these are less familiar to folks because they're sort of back in the infrastructure of the domain name registration ecosystem itself. But it is something that is commonly used within the registry systems. In fact, I believe it's mandated by ICANN.

So the protocols that we'll be talking about here, we have EPP, the Extensible Provisioning Protocol. We have DNS and a modification to DNS called DNSSEC. WHOIS, RDAP, the Registration Data Access Protocol, data escrow and the Trademark Clearinghouse.

So first, starting at the bottom, talking about the DNS protocol itself. What is the DNS protocol? It's a query response protocol that translates a domain name into something. What that something is, is a data object. The most common data object that a domain name is translated into are IP addresses, what are called A records for IPv4 and quad-A records for IPv6. It's four A's because an IPv6 address is four times bigger than an IPv4 address.

There are other common types, [somethings]. These are known as DNS types. They include mail server records, MX, which actually stands for Mail Exchange.

Address-to-name mappings, if you have an IPv4 or an IPv6 address, you can sometimes translate that back into a domain name by using a PTR record and reversing the IP address, and in the case of IPv4, adding ".inat" or ".arpa". Or in the case of IPv6, adding ".ip6.arpa".

There is also where to ask for more information about a name or more specific information about a name. And those are the Name Server records or NS records.

The query asks for the information and includes a tuple of a domain name, a type, a class, but today on the Internet, pretty much the only class that's used is class IN for Internet and those are the 3-tuple that you will send to a name server to get information back.

The response contains the information or it contains something that says "what you asked for does not exist."

The significance of the DNS? Well, it's one of the earliest protocols. The original specifications for the DNS came out around 1983 which, purely by coincidence, is actually when I started doing Internet stuff. That age impacts its design and attempts to improve, which is actually also true about me.

It has proven to be resistant to replacement. The DNS has been around for quite a while. There have been numerous attempts to come up with a better protocol because, frankly, the DNS protocol itself is sort of not a lot of fun to deal with.

Conceptually, it's very simple, but when you actually try to implement it, it turns out to be quite difficult. A lot of corner cases, a lot of weirdnesses that we've learned a tremendous amount about how to write protocols since 1983 that is not reflected in the DNS because it's both resistant to replacement and it's also very difficult to change the DNS. When you have an installed base of millions, tens of millions of machines, getting to all of those to upgrade has proven to be quite difficult.

Domain name registries exist because of the DNS, obviously. The registries are the means to enter and manage the data associated with the DNS query response transactions.

What does the DNS mean to the registry? Well, obviously, it's critical to registry operations because the DNS protocol is how a registry's data is made available to people who need that data. That operation of the DNS protocol itself is sometimes outsourced. Many registries today will outsource to a backend registry operation.

It's important to note that the registry database is not the DNS. The registry database is the information that is transferred over the DNS but it's not the DNS itself. In many cases, the registry database is maintained on Oracle or on MySQL or common database languages. Or in many cases, it's simply a text file and the DNS protocol is what transfers that information over the Internet.

There are a number of folks who depend on the DNS data. They're called relying parties. And because there are so many of them – there are on the order of about 25,000 queries per second that hit the root servers today and hit the L-root server today; multiply that by 13 and you get the total aggregate account of queries across the entire root server system – that implies a need to scale the technology quite well.

That's why – one of the reasons that UDP is the primary protocol used for the DNS. UDP is stateless. It's a very simple protocol. It has no mechanisms to flow control or anything like that. You send a UDP datagram out, you get a response back hopefully. If not, you send it out again and wait for a response.

One other aspect of the DNS itself is that the queriers are essentially anonymous. In the vast majority of cases, queries are sent through what's called a resolver. The resolver aggregates queries from a whole bunch of clients so that it's very difficult to associate a DNS query with a specific end user, making the identity of who's issuing queries quite difficult to ascertain.

This graphic – hopefully it's readable – provides a high-level overview of how the DNS protocol actually works itself. So if you have an end user wanting to look something up with a browser, they type into their browser bar or an e-mail address or something like that and that issues a query to a recursive name server. The recursive name server, or resolver, takes the domain name that has been submitted to it and tries first to look to see if it's been asked that question before in recent memory. And if it has, it can respond back immediately. If not and if you assume that the resolver just started and its cache is completely empty, what it's going to do first is to talk to the root servers.

There are 13 root server IP addresses. Within those IP addresses, you'll have any number of machines. They're replicated out across the entire planet. And the query that is sent to the root servers is the full name and the type.

So if you're looking to look up www.icann.org, the full query, www.icann.org is sent to the root server. The root server only knows about top level domains. It will respond back with what's called a referral to the org name servers. The recursive resolver will receive that and look in its cache to see if it knows where the org servers are located.

If not, it looks in the referral information that's been provided and sends the exact same query, www.icann.org, into the .org name servers, asking the same question, will get back either an answer if the org servers know, which they almost certainly won't, or a referral down to ICANN's name servers. The query is then sent to ICANN's name servers and presumably at that point, ICANN's name servers will actually know the answer, return that back into the recursive resolver which then returns it back to the device that was issuing the initial request. And then the connection can actually be initiated for the web page or to send the e-mail.

All of this will happen even on a cold cache with no information in it. All of this will happen in less than a second, generally. If you're communicating to the root name servers, right now, the average latency to a name server – I just actually saw some statistics on this – across the globe, the average latency to a root name server is about 25 to 50 milliseconds. So this entire series of transactions where the queries are going out, coming back, going out, coming back multiple times, happens very quickly and generally with a high level of reliability.

Any questions on this at this stage? Am I going too fast? Too slow? Everything okay? Okay, good. Is everyone awake still? Recovering from lunch?

Okay, so components of the DNS. The authoritative servers. The authoritative servers are what the registry will typically operate, either directly themselves or through an outsourced arrangement. Historically, there are these terms, the primary or master server, and that's where the data is edited, and the secondary, or slave servers, where the data is copied to for redundancy, resiliency and performance.

Today, most places have what's called a stealth master. The place where the data is edited and made available via the DNS protocols is actually hidden behind a firewall or in a particularly safe area. And then the secondaries are actually the ones that are actually out on the Internet available for queries. So the primary, these days, generally doesn't get a whole lot of queries directly.

The other component that I mentioned, the recursive server, is what issues the queries to the authoritative servers. And what issues the queries to the recursive or caching servers? Well, that's the stubs or the client DNS library. So every application that interacts on the Internet will have an implementation of the DNS called a stub implementation. It doesn't have any caching generally. It's pretty much as stupid as you can possibly make it. And that's integrated into these applications and they send the queries off to the recursive resolver in order to do the actual work.

So how the DNS registry system works? Well, as I'm sure many of you are familiar, a registrant will talk to a registrar. The registrar interfaces via some registration interface to the registry database to allocate the name. Over time, those names are sort of aggregated together, dumped into the DNS server. That's then replicated out to the authoritative servers and the DNS protocol queries are sent to those servers.

IANA is involved with top-level domains. The delegation of the name servers for all of the top-level domains has to go through IANA. But there is no other interaction, generally, between ICAAN as the IANA function operator and the registry system itself.

Talking a bit about DNS security extensions. DNSSEC, which many of you may have heard of that. Has anyone familiarity with the DNSSEC? Raise your hand. Okay.

DNSSEC actually was designed to address a bug that was discovered in the DNS protocol suite itself. Initially, back when the DNS was created back in the mid-80s, there wasn't a whole lot of concern about security. There were not a whole lot of worries that people would corrupt data as it flew through the Internet.

And there turns out to be a poor choice in the length of a field in the DNS packet header itself that actually allows – didn't allow back in the mid-80s, because networks were much slower and computers were much slower – but it would allow you to remotely, without being on the same wire, be able to guess with sufficient accuracy to be able to insert whatever data you want into a response for a domain name.

So a bad guy would have the ability to insert a record for, say, www.bankofmorocco.ma to point to whatever destination they would actually want that domain name to resolve to. This makes it actually quite easy to do man-in-the-middle attacks, which if you're a bank is probably not a good idea.

DNSSEC was actually developed to address that particular bug. It also addresses problems associated with the fact that the data is stored in a potentially untrusted intermediary, the caching servers. Also that the data is transferred in the open. If I'm sitting on the same wire that you are in a Wi-Fi network, I can see exactly what you're querying for and if I can respond more quickly than the authoritative server, then I can actually provide, again, the malicious data that would allow for a man-in-the-middle attack.

So the point of DNSSEC was to try to come up with a system that would allow the end user to know that the data that they received matches what was sent, what was intended by the owner of the authoritative server. It was originally developed in – let's see, when was that? About 1998, '96 to '98.

The initial version turned out to not actually be deployable. The specification didn't take into account the needs to actually maintain the data very well. So it went through a number of iterations and I actually have a slide to talk about that.

The final iteration of the DNSSEC protocol was published in 2004. Even though the protocol was published, it didn't get a whole lot of traction. It turns out that when you turn on DNSSEC, it takes what, in general, even if it had a couple of errors, would still work. It would actually make those things, even though they had errors, it would still work, they would make them stop working.

It added a tremendous amount of complexity. You actually had to make sure the [clock4s] on your resolvers were accurate. You actually have to do work with resolvers. Resolvers, generally, you turn them on and you can forget about them after that, or it used to be. With DNSSEC, you actually have to ensure that there is configuration data that has kept up-to-date.

All of that made DNSSEC sort of painful to deal with and as a result, it didn't really get a whole lot of traction until about 2008 when a guy named Dan Kaminsky discovered a way of easily inserting bad information into caches that were not protected with DNSSEC (they relied on sort of a trick of issuing queries in a certain way) and was able to insert bad data into pretty much any cache remotely from anywhere on the Internet. He actually presented that at a Black Hat conference in 2008.

That caused a tremendous flurry of activity, predictably within ICANN and VeriSign as the root management partners. We, together, went and signed the DNS root in 2010. A bunch of TLDs, and in particular, .SE had signed themselves but because the root wasn't signed, there had to be specific configuration within the resolvers to be able to understand the fact that those top-level domains were signed.

When the root was signed, that meant that there was only a single piece of configuration that needed to be inserted into resolvers instead of the configuration of each top-level domain which made it much more tractable to actually deploy DNSSEC. Today, I believe about 70% of all top-level domains are signed largely because it's a contractual requirement to sign your zone if you're one of the new gTLDs, but a lot of ccTLDs are also signed and all of the legacy gTLDs are signed.

The approach to DNSSSEC, the way it works. DNS response data is accompanied with a digital signature. So what happens is you have your DNS database, a database of information for the DNS. You run it through a process called a signer. The DNSSEC signer creates a set of signatures, cryptographic signatures over the data, and makes that available. And when a query comes in, the answer is provided with the signature in essentially the same packet.

When the response is received, the client, the recursive resolver, has a new bit of code called the validator. The validator goes and verifies that the signature matches the data that has been sent. And that ensures that the data has not been corrupted. This works because it uses public key cryptography and that allows it to scale across the Internet.

From the registry's perspective, looking at DNSSEC, what the registry has to do is they have to accept the delegation signer records from the registrant. Delegation signers are a way of building a chain of trust that goes all the way up to the root, so that regardless of where you are in the DNS hierarchy, there is a way that you can validate the name all the way up to the root which ensures that you're dealing with the Internet as we know it. The registries have to sign and publish those DS records. They have to make that available over the Internet.

Registries also have to sign negative answers. So one of the quirks of the DNS is that you have a way of authoritatively proving something does not exist. That's actually quite important because it turns out, particularly today, a large percentage of the queries that are sent to name servers are actually for names that do not exist. Unfortunately, in many cases, that's caused by malware and domain generation algorithm names, but particularly at the root level, there are a huge number of queries for names that do not exist.

The registries need to manage the keys for their own TLDs. The registries that are signed will have the keys that are generated in that signing process. They maintain the private key and deal with the public key. And they interact with IANA to register the key material. The public key is published into the DNS to allow for the validators to actually pull that information down.

The registrar's portion of DNSSEC. So the registrars submit the registrant's DNSSEC information to the registry. And they do this by sending what's called the delegation signer or the DNS key record from the registrant to the registry. This actually introduces a little bit of a problem because it implies a level of trust in the registrar that wasn't really in the original design of DNSSEC. The registrar has to ensure that they don't do the bad thing, they don't corrupt, they don't translate that DS record or DNS key record as they transmit it up to the registry. In general, they all do that. But it does open up a vulnerability that many people have noted and have criticized DNSSEC for.

Registrars also can provide value-added services, hosting the DNS. They can actually do signing on behalf of the customer. This actually addresses some of the pain of dealing with DNSSEC. You actually have your registrar deal with the signing. I will admit that I don't sign my own DNS data these days. I actually rely on a registrar to do that for me because I like to live dangerously.

Let's see. This graphic tries to show sort of the flow of data. You have the registrar talking, providing DNS functions, acting as the agent providing the DNS functions into the registry database which then gets propagated down to the DNS server and therefore, allows the DNS queries themselves to be protected via DNSSEC.

Any questions on that? Have I completely lost you all with the glories that are DNSSEC? Nope. Okay.

Moving on to, if DNSSEC was complicated, WHOIS is blissfully simple and also incredibly annoying as I'm sure many of you in the ICANN community can well attest.

So the history of WHOIS. It has existed forever. It predated even the DNS. It predated the Internet. It predated the IP. The original WHOIS specification, if you can call it that, was to find back in the days when there was a protocol called NCP that was developed prior to TCP or IP. The purpose of WHOIS is to identify the registrant of the other end of the network, whether that's an IP address or a domain name. WHOIS is the technology that was designed to allow people to look up those identifiers to get back information about who to contact when something was broken. The original purpose of WHOIS was primarily that, to allow people to do network administration, to communicate with their peers on the network just when they had an IP address or domain name.

It is a stunningly simplistic question and answer protocol that was defined at a time when there were few concerns about minor details like privacy and security and accuracy.

The definition of the protocol? Well, you open a TCP connection on port 43. You send a question. You wait a little while. You get back an answer. You close the connection. It's sometimes useful to actually display what the answer was, but that's not required in the protocol.

And that's it. That is the entirety of the WHOIS protocol definition. Just in case you were curious, this is a pictorial view of that. Did I mention these aren't my slides? So you'll have to bear with me as I stumble through them.

So WHOIS is a painful protocol even though it's painfully simple. The problem is that what's in a question and what's in an answer are not defined. It's not specified within the protocol, and as a result, you have freeform responses and everybody seems to come up with something different. And that makes interoperability quite challenging. WHOIS assumes ASCII only. The only thing that, in theory, you are supposed to send over the WHOIS connection is what's called net ASCII. It's actually even a simplified form of ASCII.

It has no way of providing meta answers, things like "I don't know the answer, but look someplace else and here's where to look." There is no way for a user to provide their credentials which means that there's no way to have differentiated access on a user basis. You can do games based on IP address or time of day, phase of moon, those sorts of things. But there is no way to identify individual users to enable a limited set of data. So you either get nothing or you get everything. And, of course, there is no way of validating the information that's provided via a WHOIS query.

All of those challenges in the WHOIS protocol led to a number of attempts to replace WHOIS. There was RWhois, Referral WHOIS, to try to get the meta data question. There was WHOIS++ that attempted to do a whole bunch of things. There was IRIS, the protocol that was developed to completely replace WHOIS and provide registration data.

And all of those basically failed horribly. The only one that had any significant penetration was RWhois and that was only for the ARIN's, the American Registry for Internet Numbers, use because if you ran your own RWhois server then you wouldn't have to communicate with ARIN as much. But all of those ultimately failed.

There is a new protocol out. Many of you might have heard. It's called RDAP, the Registration Data Access Protocol. This one actually looks like it might someday allow us to bury WHOIS in the back yard and forget it ever existed. Someday, hopefully soon.

What is RDAP? It is a query response protocol to inspect a registration database. It's layered on top of HTTPS and reuses much of the web-developed technology, which means that the infrastructure to support RDAP already has a huge installed base.

The other protocols that I mentioned, in particular the IRIS protocol, all had the downside that it required everyone to deploy an entirely new set of protocols and infrastructure. And as anyone who runs a registry or registrar knows, WHOIS is not a profit center. It is a cost center and is nothing that doesn't generally provide much in the way of business benefit. It tends to be something that is mandated by evil folks like us here at ICANN. RDAP did away with the requirement to deploy an entirely new protocol infrastructure.

The components of RDAP. Well, there's obviously software that receives and parses the queries. There's software to access the database, software to prepare and send the responses back. All of that would sit at the registry or the registrar. And then there's a client with a web browser API with specific abilities to transform JSON, the response, and do a couple of other things.

The history of RDAP. Well, as I mentioned, no one's really happy with WHOIS, especially anyone that actually has to deal with it. Two of the IP address registries, I believe ARIN and RIPE – I know ARIN; I'm not sure if RIPE was the other one – but they started experimenting with new approaches of deploying registration data. They came up with a web-based approach and it was particularly successful. They then took that to the IETF and it became standardized into RDAP.

The result is a protocol that is very much tied to replacing the WHOIS protocol. The data schema associated with RDAP allows for different data types. So you can have names, you can have numbers, you can have autonomous system numbers, IP addresses, contact information. It has a fairly elaborate data protocol that allows for a commonality of fields. And it has security sort of built in because it sits on top of the HTTPS protocol. That's not actually a requirement. You don't have to deploy it on top of HTTPS, but today, pretty much everyone does because of an increased interest in privacy.

Basic description of RDAP. You send a query over HTTPS. It looks like a URL. This is just a structured and formalized version of the queries that were sent via WHOIS. Those are two examples. One's for IP addresses. One's for a domain name.

The response comes out over HTTPS using something called JSON. Again, like WHOIS, it's just sort of a data dump. But this time, it's structured and formalized.

In addition, RDAP provides a way of doing redirection. You can say, "I don't know the answer, but if you query that RDAP server, you can get an answer or another referral."

So this eye chart, which you're not intended to be able to read, is the response if you point your browser at that URL that's in the title. That's the first part of the response that you will get back. It's actually much longer than that and it actually has all of the information that you might expect in a WHOIS response. So here's the address that was being looked for. It's the network name, some meta information, a whole bunch of meta information. If you look at it on a browser, it actually has the contacts associated with it and it goes on for a couple of pages like this. JSON is actually a fairly ugly presentation format, but it is better than XML in my opinion.

Features of RDAP. Well, it actually has a very nicely-defined data model that's expansion friendly and allows for query and responses in sort of an obvious way. It allows for expansion beyond ASCII, internationalization. So you can put pretty much anything in the responses. I believe it's fully supportive of UTF-8. It supports IDNs, of course. You can distribute the data sources around. You can have these referrals that allow the clients to be automatically redirected to where the data is actually being stored.

RDAP has a way of allowing for differentiated access. The authorization model is still sort of up in the air, as some of you may know who are looking at the operational profile for RDAP being defined by ICANN. One of the outstanding issues is the authorization model. And it actually uses modern, up-to-date software engineering principles, web design and APIs and that sort of stuff. So it actually is a modern addition to the Internet.

A pictorial description of how things are supposed to work. RDAP client talks via RDAP to the handler which talks back to the registration database.

So that's a quick summary of our friend RDAP. Anyone have any questions? Yes?

[PEDRO]: Hi, everyone. My name is Pedro [inaudible]. I am the general manager of [Center]. Maybe I'm supposed to know most of this stuff already. But I was wondering why do you think RDAP will succeed where RWhois and IRIS and the others have failed?

DAVID CONRAD: So RDAP has a lot of advantages going for it. The biggest one is the fact that every computer already has a client that allows you to look up RDAP. You don't have to go out and get a special client, like a WHOIS client. Every computer has a web browser these days and that's all you need to be able to look up information using RDAP.

RDAP also has a bunch of other advantages. The internationalization is an advantage that will drive a lot of interest. The differentiated access. The big advantage of differentiated access, it allows registries and registrars to deal with data privacy laws. If you are an organization in a jurisdiction that disallows exposure of particular pieces of information, you can limit that information for classes of users. You cannot do that with the WHOIS protocol. So that's another one of the big drivers.

The referral ability within RDAP is actually very nice because it actually makes it easy to write a client if you don't want to use a web browser. You only have to know how to get to the RDAP root registry which exists at IANA at a well-known URL. And once you have access to that, then you can figure out where to go to find all the other information. So it has a number of advantages.

Also, ICANN has a contractual obligation to the contracted parties that forces them to move to RDAP within some period of time after RDAP is standardized. That's a minor detail.

[PEDRO]: So as you referred to, WHOIS is a cost issue for most registries. How does RDAP do that? Is it where you typically that it will reduce costs and number of queries, bandwidth and registries, or there is no relationship whatsoever?

DAVID CONRAD: Well, so yeah, the software that I'm aware of for RDAP is open-source. There is obviously an operational cost for deploying a new service. That varies on the registry. In terms of bandwidth consumption, it's not significantly different than what you get in WHOIS. I don't think there will be any significant cost differential that way as well. I think there was a question back there.

[MATTHEW]: I'm Matthew [inaudible] for the record. Is there any relation between this RDAP and the RDS which GNSO is working on currently in a working group?

DAVID CONRAD: It's all part of the same, dealing with registration data. So right now, there's the issues of WHOIS, thick WHOIS, thin WHOIS, the registration data models and the transition to RDAP. All of that is incorporated, as I understand it, within the panoply of the RDS stuff.

[MATTHEW]: So RDS is based on RDAP?

DAVID CONRAD:	Well, RDS is sort of an overarching aggregation of issues, right? There's the transition from thin to thick WHOIS. There's the transition from WHOIS to RDAP. All of it's sort of incorporated into this umbrella called RDS.
[MATTHEW]:	Thank you.
UNIDENTIFIED MALE:	David, I actually have a question. As we saw the presentation from Daniel this morning from the W3C and RDAP rides on top of HTTP, what was the decision factor to put this into the IETF versus W3C process?
DAVID CONRAD:	I'm not certain but I believe it was felt the W3C is providing the protocol for the underlying transport, but the data model and those sorts of things was [held] to be related more to objects that were associated with the IETF, the domain name system and IP addressing. That's my guess. Any other questions? Okay, moving right along. Talking a bit about the extensible provisioning protocol. So what is EPP? It's a business-to-business protocol between a registrar and a registry. It's used by registrars to essentially edit the

registration database that's held by the registry. You can add and delete registered domain names. You can add and delete modified contacts. You can implement transfers. And there's some other maintenance functions that are provided via EPP.

History. From 2000 to 2003, it was developed within the IETF. It was actually based on an earlier protocol called RRP, I believe, the Registry Registrar Protocol. Is that right? Yeah. I actually was involved in the panels that VeriSign put together. Were you NetSol or VeriSign in those days? Network Solutions actually put that together. That was a fun exercise. And it was primarily to facilitate the transition to the competitive world of having competitive registrars to the common net registries.

From 2003 to 2009, it progressed to a full standard. The IETF travels at a certain speed. It took, what, six years to get to the full standard. It is mandated by ICANN for gTLDs and sponsored TLDs. But it has gained a significant amount of acceptance within the ccTLDs, perhaps because registrars use it to talk to the registries and the ccTLDs want to play in the same pool as the gTLDs. It's useful, not essential, but useful to use the same sort of software.

There is now an IETF working group that has been [inaudible] deal with the extensions that are being developed by various registries and registrars.

A similar graphic of how the protocol works. The registrar EPP client talks over TLS, a secure protocol to the EPP server which communicates with the registry database. In EPP, the exchange is done via XML, Extensible Markup Language. Fairly wordy. JSON is actually sort of a simplified form of XML.

Server inside registry. Clients at registrars. Yes, so the server sits at the registry. The registrars operate the client. Wow, that's all on that.

So EPP is a protocol that is primarily used, as I mentioned, in the transactions associated with creation, modification, and deletion of domain names. Any questions on EPP? Okay.

And I should say at this point, we're getting to areas that I have less familiarity with. I don't generally interact with data escrow or TMCH, so Ed had a much better understanding of this. But I will go through and answer the questions as best as I can.

So the data escrow. The purpose of that is to have a copy for safekeeping of all the data needed to run the top-level domain. That copy is held by a third party. Iron Mountain is the one that I'm most familiar with. And there are rules for releasing the copy of that data to a beneficiary.

There are a number of underlying protocols associated with data escrow, transfers of the encrypted files, verification of the data, a notification of various events that occur in the lifetime of data that's stored within the escrow.

The architecture, conceptually, you have a registry, a registrar. It dumps data into a data escrow agent. Some event happens, like say the registry going bankrupt or something like that. That causes the data to flow up to a beneficiary that then provides that data instead of the registry or registrar.

Setting up the data escrow. There are a set of escrow agents that are pre-approved by ICANN. The escrow participants each have a public/private key pair and they're exchanged as appropriate. And once the registry or registrar has an agreement with the agent, the escrow protection certificate is issued by the escrow agent and is presented to ICANN. This isn't an X.509 or PGP certificate or anything like that. It's just a statement that says things were set up correctly.

Escrow process. Well, either the registry or the registrar needs to make a recording. They actually need to provide the data to the escrow agent. There's a couple of ways that's done. One is a full deposit. The entire database is just provided in complete.

There's another way of providing the data and that's what's changed since the last full deposit [inaudible]. And then there's the incremental which is the changes since the last deposit, either the full or the differential.

The escrow agent will decrypt and open the file to verify the format of the deposit is correct. Ideally, they'll be able to reload the data and ensure that the data is correct. They then put the encrypted files into some sort of storage area, some safe place. I know, for example, Iron Mountain has a facility that's in some salt mines in Pennsylvania, I believe. And for the registrars, ICANN can actually act as the escrow agent.

Notifications. Well, the beneficiary is notified of actions such as the registry or registrar making a deposit. The escrow agent receiving invalid formats, or valid formats, I guess. Also, the escrow agent will say that they haven't been paid. These escrow agents are commercial businesses, generally, and expect to get paid for their services. The registries and registrars are required to actually pay them.

The failure in the notification triggers some sort of response. Some contractual breach, on occasion, will be issued, all of those sorts of [inaudible].

[inaudible] ICANN environment. Well, ICANN is a beneficiary. For example, if a registry fails then the data would then be fed to an ICANN-designated emergency backend registry operator, EBRO, and that EBRO would then restore that data and allow the registry functionality to continue, albeit in a limited sense and via a different party than the original registry that had provided that service.

There are five to ten data escrow agents that have been approved. That's the URL if you're interested in who the agents are. And the rules between the registries and the registrars are actually a bit different.

Schedule of events for a registry. A full deposit is due every Sunday at 23:59 UTC. All other days, a differential is provided. If you're a registrar, a full deposit is done weekly. I imagine Sunday by 23:59 UTC. If the registrar has more than 100,000 transactions, then a daily incremental is required. Otherwise, I guess a differential is used. And, of course, missing or failed deposits are recorded and notified.

What's in a registry deposit? Well, you have this XML file and it has the RDE. I don't even know what that stands for. The type of deposit is [fuller] differential, the ID looks like a date, and then the actual data itself.

So the menu, I guess these are all of the different types of deposits that can be made. There are contacts, hosts, domains, registrars, IDN labels, non-DNS names registered, and the EPP parameters. This is all the information that's necessary to enable a registry to be restored in the event that a registry has failed for one reason or another.

What's in the registrar deposit? It's actually a comma-separated value format with the information you would expect: the registered domain name, the contacts associated with those domain names, and whatever other information is necessary to ensure that the registrar information can be restored in the event of a registrar failure in one form or another.

As I understand it, there has never been an execution of EBRO for a registry. A registry has not failed. There have been cases where the registrar deposits have been extracted where a registrar has failed or been terminated for one reason or another.

If you're interested in the technical specifications of all of this, it's in the registry agreement for Spec. 2 for registries and the Registrar Accreditation Agreement for the 2013 RAA, Section 3.6.

Any questions on the data escrow stuff? Yes?

UNIDENTIFIED MALE: Sorry, I didn't have my glasses. Okay. Just looked, RDE means – I forgot already – Registrar Data Escrow.

[GARRETT]: Hi, my name is [Garrett] [inaudible] for the record. So I have two questions, in fact. The first one is, is there any timeline for the emergency backend registry operator to bring back the registry operation to normal? Because, I mean, I know today there is no the case of registry that went bankruptcy but [inaudible] TLD probably you will see that.

DAVID CONRAD: Yes, there is a specific SLA that has to be met in restoring. The EBRO providers are under contractual SLAs to restore under a certain time limit. Do you know, Joe, offhand what the SLA is for the EBROs?

JOSEPH CATAPANO: There are different components. So I think there's a component for DNS. There are separate components, but I think it's 24 or 48 hours.

DAVID CONRAD: Yeah, I think the registration data is within 48 hours, but the DNS has to be up in six or something like that. It's a relatively short timeframe.

JOSEPH CATAPANO: And there was a test that was just done recently that you might want to—

DAVID CONRAD: Yes, thank you. Yeah, while no registry has failed, there has been a registry that was terminated, the first one ever, I guess.

UNIDENTIFIED MALE: [inaudible]

DAVID CONRAD: Yes, .Dusan was one of the brand TLDs and the Dusan Corporation in Korea decided they didn't really want to do this anymore so they terminated and we, at ICANN, actually took advantage of that to experiment to actually do a case study of executing an EBRO.

[GARRETT]: Yeah, anyway, I think 24 hours is still a big timeline for any business. I mean, if I am a business owner and I have my domain name and my users using this domain name, 24 hours is a lot of time.

DAVID CONRAD: Yeah, and I believe the DNS SLA is actually much shorter. I don't recall exactly what it is. I can get that information to you. If you leave me your card, I'll make sure.

[GARRETT]: Thank you. My second question is about the ccTLD registries. Is there an escrow requirement or policy for them?

DAVID CONRAD: There are no requirements on ccTLDs.

[GARRETT]: The reason why I'm asking because a lot of ccTLDs that we have seen in the past, like .TV or whatever, that are becoming more generic top-level domains selling their domain names to businesses and I imagine it's the same level of importance for the business users and users in general.

DAVID CONRAD: Undoubtedly, that is true. However, ccTLDs are treated essentially as national sovereign entities. ICANN has no way of forcing or encouraging – well, we can encourage but we can't mandate – the ccTLDs to do anything at all, period. So I'm sure there are ccTLDs that make use of data escrow services, particularly if they use a registry backend operator, but we have no information, I mean, no control over that whatsoever at ICANN.

This is one of those situations where the consumers need to be aware of what the facilities are being provided, specifically for cases like failures of the registry. If a ccTLD registry failed, it would be an interesting event.

Any other questions? Steve?

STEVE CONTE: By the way, [inaudible].

SANJAY SINGH: My name is Sanjay Singh and I come from India. In case of a similar sounding or exact name between a geographical entity and a brand – so we're talking of geography versus brand – corporate organizations are supposed to get an NOC from a government before the grant of a domain name. Almost in none of the cases where an NOC is required, because it becomes a question of sovereignty, really, and it is not the ministry of industry. But then it becomes the ministry of interior's business to grant that NOC.

So many potential Internet users get trapped in that situation where they have applied for a registration, paid the money upfront, are waiting for an NOC which will never come. And in the evolution of the Internet, it has been seen that in the power struggle within the stakeholders, the government is getting the better of the corporate, but it's the corporate who brings in the money. So it's in the interest of ICANN to resolve this issue.

DAVID CONRAD: An interesting point. The reality is that ICANN's abilities to impose solutions or to mandate particular approaches is limited to specifically the contracted parties, so the gTLDs that have entered into contracts with ICANN. We can provide guidelines and encouragement that are developed through community-developed policies to other TLDs – ccTLDs, for example – but those are voluntary. There's no way that ICANN can mandate them.

In the case of gTLDs, there have been some limitations on what names can be placed at the second level for whatever reasons, most of them technical. To date, most of the policy development processes have been aimed at sort of facilitating the free and open flow of names, at least within the generic top-level domains.

Any other questions? No. Okay. Moving right along to the Trademark Clearinghouse.

TMCH has a set of protocols, not in the bits over the wire type protocols definition, but ways of interacting. They are primarily a set of interfaces that allow for registries, registrars, and registrants to ensure that they're not stepping on somebody's intellectual property claims. These functions that are provided by the TMCH actually come in two main phases: the sunrise period and the trademark claims period. There are other phases. Different registries use different terminologies. And they use different parts of the protocols as necessary.

So the Trademark Clearinghouse, it consists of a trademark validator that verifies that the trademark exists, is registered in some jurisdiction. It verifies that the mark is actually in use. There is one validator now, but there can be more in the future.

There is a trademark database. This is essentially a registry of all the trademarks that are used for the purposes of DNS registration. Then there's the validator. The validator acts somewhat like a registrar. So the validator will look into the trademark database like the registry database.

The overall architecture of how the TMCH works. So a trademark holder will deposit a trademark into the Trademark Clearinghouse, which is right here. In a sunrise, the trademark holder can become a registrant, talking to a registrar who will transact with the Trademark Clearinghouse which I'll describe more later, which will then transact with the registry, with the trademark database which I will explain more later. And then there's also a Trademark Clearinghouse Certificate Authority that's run by ICANN that ensures that these transactions are actually secure over the network.

The phases. So there's the sunrise period in which trademark holders get to register domain names corresponding to their intellectual property marks before the names are generally made available to the public. And then there's the trademark claims period which follows the sunrise period. And this is during general availability, for at least 90 days, the initial operating period of the general registration.

Anyone attempting to register a domain name matching a mark that's in the TMCH will receive a notification that displays the relevant mark information. That is, if you try to register CocaCola.something, you will get a notification that you're attempting to register a trademark.

As I mentioned, there are other phases. The registries have the ability to define the registration phases as they like. Some of them [are] called founders program, qualified launch program, limited registration period. These different phases have the ability to use different notifications. Some of the phases have other restrictions. Every registry has the ability to do things sort of differently within the context of the contractual agreements. And some TLDs actually remain in the claims phase indefinitely.

Talking about sunrise. During the sunrise, special conditions exist. Only trademark holders can register names and only if the trademark is accepted in the Trademark Clearinghouse. When it is, a signed mark data token is created and is used to indicate whether the registration data has passed those two tests.

Let’s see. The sunrise period establishing the mark. So a trademark holder registers with the clearinghouse and receives a signed mark data token for their mark. Unless the trademark holder is the registrant, the SMD token, Signed Mark Data, is handed over to the registrant.

The trademark holder can then register the domain name by submitting the SMD and the registrar will then give it to the registry who will then allocate the name without the notification, which is what the slide says. So the registrant presents the SMD token when requesting a trademark name. The registrar checks to see if the SMD token is valid and it hasn’t been revoked. The registrar is actually responsible to look up the SMD in a revoked list in the Trademark Clearinghouse. The registrar then, assuming all that passes, it submits the EPP request to the registry. The registry also validates the request, the SMD token, before accepting the registration.

When a name is registered, the trademark holder is notified. The point of this is to allow for an IPR holder to be made aware that their trademark has been used in a domain name.

Obviously, these marks need to be maintained. The trademark holder can withdraw a mark at any time. And the trademark holder is informed when a domain name matching a claim is registered. All of this is done through the TMCH.

Trademark claims. During trademark claims registration, this is more normal, but notifications are delivered. Names can be registered without relation to the trademark designation, but parties are notified. Generally done through click-through to the registrant at the application. Disputes, obviously, can arise here and they need resolution before things can move on. Or the registrations may be permitted to proceed.

Claims. The registrant requests the name. The registry issues a notice to the registrar. The registrar displays names that have the trademark claims. If the names are registered, the TMCH is notified. This is during the general period which anyone can register a name and there's no signed mark data required.

The maintenance and notice are similar to how things are done with the sunrise period. Let's see. Is there anything more to say about that? No, pretty much.

And that is a quick summary of the protocols that are used at a TLD registry. I would be happy to answer any questions anyone might have. Question here and then in back.

UNIDENTIFIED MALE: Sorry, where was the first one?

DIETMAR STEFITZ: Thank you. Very interesting. My name is Dietmar Stefitz from DomainingEurope.com. I have one question about registrars. ICANN is giving a license to registrars if they are ICANN-registered registrars.

DAVID CONRAD: Yes, they're accrediting registrars.

DIETMAR STEFITZ: Right. Okay. So why does ICANN give to hundreds of registrars the license who are only using the registrations for drop-catching? They're just making thousands and thousands of things for drop-catching. And they have hundreds of them.

DAVID CONRAD: Yes, I understand. ICANN does not have any role to make judgment over business models. If a business wants to exist on drop-catching, it's not ICANN's place to say that's an invalid… I mean, let me say that there is no policy that says ICANN should not treat that as a valid business model. It's not our role. If the community decides that's not an appropriate functionality for a registrar, then that would presumably be written into the next version of the Registrar Accreditation Agreement which is, I think it's every four years the RAA gets revised. Four or five years, something like that. So it's revised.

Yes?

UNIDENTIFIED MALE: Just to want to know exactly where the Trademark Clearinghouse got the data of trademarks? Because my understanding of the trademarks is this. I mean, every mark is registered in a specific jurisdiction and place, so if I am based in Africa and I am registering my mark in Tunisia or Morocco, how the Trademark Clearinghouse got this information?

DAVID CONRAD: I should qualify this and say that I am not positive. As I mentioned before, I'm not a particular expert in the TMCH or data escrow side of things. But my understanding is that the registration of a trademark within the TMCH also includes the jurisdiction in which it's registered. But it doesn't actually matter that much because as long as it's a recognized trademark agency, it's incorporated into the TMCH and that's used.

The information that's provided to the registrant, the click-through that's provided, says this name is a registered mark in some jurisdiction. And it's up to the registrant or the registrar to make the decision whether to go ahead and try to do the registry. It's basically a notification that you may be stepping on somebody's trademark and doing so will result in you potentially getting slapped by somebody.

Any other questions? Comments? Yes?

UNIDENTIFIED MALE: Where? I didn't see a hand. Here?

ALIREZA KASHIAN: My name is Alireza from [inaudible] Company. My question is about the bulk request to the WHOIS server. I've seen a lot of businesses around the world using the bulk request sending and I don't know if there is any restrictions for sending a lot of requests to the server.

If there is, then how come there are a lot of companies that they claim that they have millions of records from the history of the WHOIS in the past ten years and they're selling that information to the people? So how does that happen?

DAVID CONRAD: So on the question of bulk WHOIS, I believe the simplest answer would be that registries are allowed to have, essentially, whatever policies they like with regards to access to bulk WHOIS data. Some can allow it. Some can disallow it. Some can require you to pay money to gain access to it. Jim might actually be able to answer this, being a registry that has, I believe, a bulk WHOIS policy.

JIM: First, I'd like to just draw a distinction between bulk WHOIS and a high volume of WHOIS queries. So there is a separate provision for bulk WHOIS and again, now I'm not the expert on the new gTLD requirement, but the only one that we can provide bulk as a gTLD registry operative, the only one we can provide bulk access to is to ICANN or a third party that they designate. I think what you're referring to are WHOIS users who come in and query WHOIS at a high rate on a regular basis. I just wanted to draw that distinction.

And I think David's right. The registry policies about the volume of WHOIS queries that we support is essentially driven by the capacity of our systems to ensure that it is still able to respond. And then most registries, I think, have rate limiting policies to ensure the availability of the service.

DAVID CONRAD: And with regards to sort of the historical data that you've seen, basically, there are organizations out there who have been collecting this data for a very long time. And they just keep it in a database. The information on the Internet never seems to go away even when you want it to.

Any other questions? Then if not, I thank you very much. Oh, sorry.

UNIDENTIFIED MALE: I have one question.

DAVID CONRAD: Sure.

UNIDENTIFIED MALE: Can we access the [inaudible]?

DAVID CONRAD: Yes. All of the materials that are provided in these How It Works tutorials are made available. I'm not exactly sure where.

UNIDENTIFIED MALE: I will address that now.

DAVID CONRAD: Lovely.

UNIDENTIFIED MALE: First of all, I would like to thank David for this. I know it's not his slide deck, but well done. Yes, we are going to make this material available. I'll be getting it uploaded to the meetings website, which process happens so sometime in the next day or two, you'll see it there. It'll also be on the archives so it'll be available from now until the end of us as human beings.

We passed along a sheet of paper. We're actively looking for feedback, how to make this interesting, how to keep it interesting, how to make it better, what do you all want to see that we're not doing that you would like us to do for future meetings. So any feedback you could provide us, we listen to, we read. We will take that into consideration.

Our next session is in 20 minutes. It's with our root server operators and the Root Server Stability Advisory Committee. Security. System, sorry. Too many acronyms, even for me. It's a really interesting presentation. If you guys haven't seen this before and you have an interest in the root server system, it's a really cool presentation. They did this in Dublin and it's really worth sticking around and watching. So that' s in 20 minutes.

And like I've been saying all day, we're doing this entire session day, repeating tomorrow. So if there's somebody who hasn't been here who you think should be here tomorrow, please let them know. It's on the agenda. We'd love to have them attend tomorrow.

So with that, please enjoy your break and hopefully we'll see you all back in 20 minutes.

[END OF TRANSCRIPTION]