

IDNs

A brief whirlwind tour

August 20, 2013



Arabic Internet . Our Future

IDNs

Assumptions

August 20, 2013



Arabic Internet . Our Future

Assumptions

- Familiar with DNS:
 - What it is...
 - What it is used for...
 - High-Level understanding of how it works...
- Familiar with the Domain Name Industry:
 - ICANN
 - Registries
 - Registrars
 - ccTLDs, TLDs
 - etc
- Familiar with issues that surround domain names such as phishing and other forms of domain name abuse

DNS

Traditional DNS

August 20, 2013



Arabic Internet .Our Future

Traditional DNS

- Typically, only allows the characters 'a-z,A-Z,0-9,-'
- In special cases other characters are allowed, but can be ignored for this discussion
- Uses an encoding system known as ASCII (more on this later)
- It is case 'insensitive' which means that the following DNS names are all considered equivalent (that is they will return the same answer when you look them up in the DNS). Which is what as a human we would expect:
 - APPLE
 - apple
 - Apple
 - ApPIE
 - aPpLe
 - APPlE
- This is built into the DNS protocol



Internationalizing DNS

- In order to internationalize DNS, i.e. allow DNS names to include more than just the Latin alphabet, a way had to be found to include the additional characters (called code points) without breaking backwards compatibility
- But first an aside...

Computers & People

ASCII

August 20, 2013



Arabic Internet . Our Future



Computers vs People

- People talk with letters
- Computers talk with numbers
- To make computers human friendly we need a way to translate between the two...
- One of the most widely used systems is ASCII (used by standard DNS)
- ASCII defines a system of numbers that map to 'letters'
- ASCII typically only covers the latin alphabet, digits and some symbols
 - 65 -> A
 - 66 -> B
 - 67 -> C
 - 97 -> a
 - 98 -> b
 - 99 -> c
 - 37 -> %



ASCII example

- A group of letters in sequence is referred to as a 'string'
- So 'abcde' is a string, consisting of the letters a, b, c, d and e
- In a computer the string 'cat' is represented internally as the sequence of numbers 99, 97 and 116 (the corresponding numbers in our ASCII table)
- When a computer is asked 'are these two strings the same' it simply checks to see if all the numbers that represent the letters in each position of the two strings are equal, if they are the strings are the same
- When a computer is asked to convert a string to upper case it will iterate over the 'numbers' and replace each number that represents a lower case letter with the number that represents the corresponding upper case version of that letter
- When a computer is asked to case insensitively compare two strings it will typically convert them both to the same case first, this is typically referred to as lowercasing, or uppercasing

Computers & People

Unicode

August 20, 2013



Arabic Internet . Our Future

Another mapping...



- Another mapping that is widely used with computers these days, to provide ‘internationalization’ is called ‘Unicode’.
- Unicode is, amongst other things, a giant table of ‘code points’. It is defined by a consortium of interested parties – The Unicode Consortium
- A code point is a number that is assigned to represent a different character or glyph, e.g.:
 - 97 -> a
 - 1571 -> á
- In Unicode, the Latin letters maintain the same number mapping as in ASCII
- However number mappings are defined for hundreds of thousands of other ‘characters’ or ‘glyphs’ from all different languages including Simplified Chinese, Traditional Chinese, Arabic, Greek etc and even some obscure things such as musical notes, symbols and Klingon
- Code points numbers are normally represented in Hexadecimal (a short hand way of writing big numbers) and written with a U+ in front of them. E.g.:
 - U+0061 -> a
 - U+0062 -> á

Another mapping...

- Unicode also defines a collection of ‘procedures’ that can be followed to perform things such as lower casing a string, sorting (comparing) strings and so forth
- Unicode has evolved over time, there are new versions of it in constant development
- Backwards compatibility between versions is a consistent goal, that is almost always met
- Almost every major computer system (designed for human use) today (Windows, Linux, OSX etc..) has support for Unicode
- However a lot of ‘infrastructure’ (routers, switches) may be left behind
- Also a lot of software is not written to be Unicode aware (for various reasons, especially older software)
- Most DNS implementations fall into this category



Challenges of Unicode

- Unicode brings with it a new set of challenges, any one who uses it, needs to be aware of these. Some examples:
- Same Glyph with multiple mappings
 - U+0623 (أ) – Arabic Alef with High Hamza
 - U+0672 (آ) – Farsi Alef with High Hamza
 - Latin B (U+0042), (lowercases to b (U+0062) in Latin, β (U+03B2) in Greek and В (U+0432) in Cyrillic), therefore there are 3 code points for B (so the right lowercase can be used):
 - B (U+0042) -> b (U+0062)
 - B (U+0392) -> β (U+03B2)
 - B (U+0412) -> В (U+0432)
- Combing marks (same Glyph, different sequence of code points)
 -  (U+0627,U+0654) - which renders as أ
 -  (U+0623)
- Zero width joiner / non-joiner – a code point that doesn't display!
- To users it is reasonable for them to expect these will all 'be the same' as they all look the same, but to a computer (number by number comparison, they are different (not the same))



Challenges of Unicode

- Special 'case' scenarios
 - German ß (U+00DF), (uppercases to SS, which then lower cases to ss, but there also exists an uppercase version)
 - Greek Σ (U+03A3), lowercase to σ (U+03C3), unless in the final position of a word where it lowercases to ς (U+03C2) (think about how multiple words are used in domain names)
- Technical limitations of implementations
 - Excel may display numbers using code points from other languages, internally they are stored as values and converted on display, this can (not always) affect copy and paste (for example).
- Again users have, reasonable, expectations that these things just work (as they do in the 'languages' they represent)



Challenges of Unicode

- Unicode provides some ‘procedures’ to deal with these
 - Case fold (context agnostic)
 - Case mapping (context sensitive)
 - NFC
 - NFD
 - Etc
- Complicated and not easily understood
- Don’t always do what the user ‘expects’

Computers & People

Computers & Languages Other Than English

August 20, 2013



Arabic Internet . Our Future



Challenges of other languages...

- Regardless of the encoding, other languages bring with them challenges that are not as straight forward as English (which really only has to deal with upper case and lower case equivalency)
- Simplified vs Traditional Chinese
- Special case handling not dealt with (or not dealt with properly) in Unicode (like the eszet case in German, or the Latin character Æ, which sometimes is replaced with AE)
- The nuances that have been developed over time, by speakers of the language working around not having there language work properly (or at all) on a computer (example the word cafe is supposed to be café)
- Users have certain expectations, that in those that speak those languages are just as natural to them as upper case and lower case equivalency is to English speakers. These all create issues when we try to think about domain names in other languages. Especially when we consider problems such as phishing and other potential for misuse, misleading and abuse

IDNs

... and DNS

August 20, 2013



Arabic Internet . Our Future

So what does this mean for DNS?

- Well we needed to develop a way to make Unicode code points available for use in domain names without hurting backwards compatibility.
- Then we need to make sure we address the new issues this creates around Unicode concerns, as well as meet the expectation of users
- Some of these were addressed at the technical level with the invention of a new 'protocol' IDNA – more on that coming up
- Other issues were left to application implementers and policy writers to address

IDNs

IDNA

August 20, 2013



Arabic Internet . Our Future



Enter IDNA...

- IDNA stands for Internationalized Domain Names in Applications
- In the end we didn't actually modify DNS, we left it alone, it was too hard
- What we did was produce another protocol that is layered on top of DNS and sits between applications and the DNS
- This is the IDNA protocol or standard
- There have been 2 versions, an early version in 2003, and a new version called IDNA2008 that was produced to address issues found with the earlier version of the protocol
- We will focus on IDNA2008



So what is IDNA?

- Whilst it is a protocol in the dictionary definition of the term
- It is NOT a protocol in the sense that DNS, HTTP or EPP are protocols
- It's really three main things:
 - A way of converting a Unicode string into an ASCII string so that it can be used in the DNS protocol
 - A sequence of steps that a Registry must follow before accepting a name for registration
 - A sequence of steps that an Application must follow when looking up a name in the DNS

IDNA Protocol

Converting Unicode to ASCII

August 20, 2013



Arabic Internet . Our Future



Converting Unicode to ASCII

- Uses a process known as punycode encoding and decoding
- Punycode takes a Unicode string and encodes it, using a reversible algorithm into an ASCII string. E.g.:
 - شكرا -> mgbti4d
- The ASCII string can be converted back into the Unicode string using the reverse process
- The ASCII string is then able to be used in place of the Unicode string in DNS
- In order to signal to an application that the DNS string is to be interpreted as IDN, the string is prefixed with the 'tag' xn- to become:
 - xn--mgbti4d
- This is the domain name 'thank you' in Arabic as it would be seen in the DNS. Only applications that understand IDNs will know that they need to convert it to its Unicode form to display to users
- However older applications that don't understand IDNs, will still continue to use the domain name as normal, it just wont mean much to users that see it



U-Labels & A-Labels

- Not all valid Unicode strings are valid IDNA domain names
- The IDNA protocol defines which code points in Unicode are valid to use in IDNA (for example the confusing dots are omitted)
- It also defines some special rules called ‘context rules’ that control the use of certain code points (like the zero-width joiner) – code points that are subject to these rules are given a name like ‘context-j’ or ‘context-o’ to define which rules they are subject to
- A Unicode string that is valid according to the IDNA protocol rules is referred to as a U-Label
- An ASCII string that has been created from a valid U-Label is called an A-Label
- An IDN domain name is any domain name where at least one of its labels (one of the strings between the dots) is a valid U-label (or A-label)



So what does this mean?

- This means that IDNA is really, as its name suggests, an application layer protocol
- In theory, the DNS infrastructure does not need to be modified in any way to support IDNs
- However, DNS administration tools (including domain name Registries and Registrars (and resellers)) have a lot of work to do
- This also means that all the work to support IDNs is required to be done in Applications
- It also means that any protocol that has, what we call, a domain name slot, may potentially need to be updated as well to allow the inclusion of internationalised information. E.g. SMTP (email), Jabber, HTTP
- Efforts are underway on these (IRIs, EAI etc) but more work is needed (more on this later)

IDN Registrations

What Registries Must Do

August 20, 2013



Arabic Internet . Our Future

So what must Registries (and Registrars) do?

- Verify that an IDN name being registered is in Unicode NFC form and reject it if not
- If the domain name is provided as an A-label format, generate U-label version using punycode
- If the domain name is provided as a U-label form it is strongly advised not to accept it to avoid any ambiguities
- Validate that both A-label form and U-label form are in fact related, and reject if not
- Reject any name with leading combining marks (Unicode terminology)
- Reject any name that contains consecutive hyphens in the 3rd and 4th positions (in the U-label – to avoid confusion with tagged ASCII names)

So what must Registries do?

- Verify that the domain contains only valid code points as defined by the IDNA standards, reject if it doesn't
- Apply the joiner rules (context j rules), reject if these rules fail
- Verify that for each context o code point, a rule exists in the standard and that when the rule is applied the domain name is still valid, reject if any of these rules fail
- If the domain contains any right-to-left code points apply the BIDI rules, reject if any fail



But that's just according to the protocol...

- In order to be diligent Registries, and to deliver on the expectations of those that have spoken the languages for a long time, Registries must also:
 - Ensure that we protect against phishing and farming scams
 - Determine and 'deal with' names considered equivalent (policy issues)
 - Further ensure that the IDN registrations makes sense – ensure that names make sense in the desired language
- More typically these are referred to as:
 - Verifying the name against the language table
 - Generating blocked variants (Checking for duplicates)
 - Activating appropriate variants

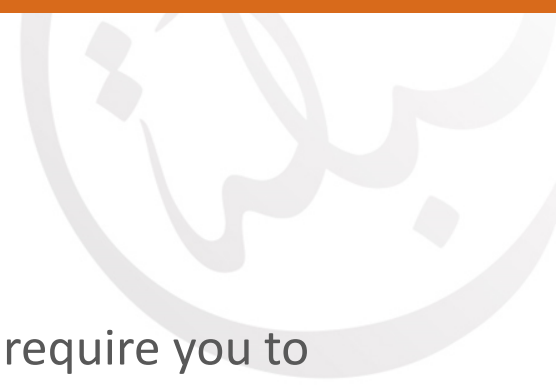
IDN Registrations

Language Tables

August 20, 2013



Arabic Internet . Our Future



Language Tables

- When registering an IDN domain Registries normally require you to specify the 'language' of the registration
- A Registry can support one or more languages for registrations under their namespace
- Registries will define the set of Unicode code points that make up the language or languages supported for registrations in their namespace
- Whilst typically registries can define 'languages' as they want, ICANN has rules that require all registries 'accountable' to ICANN to ensure that all the code points in their language table(s) are from the same script as defined by Unicode
- The distinction between language and script refers to where the code points come from as apposed to the language, one script may be used by many languages



Language Tables

- The distinction between language and script refers to where the code points come from as apposed to the language, one script may be used by many languages
- An Arabic language table will include the Arabic glyphs from the Arabic script
- A Farsi language table will include the Farsi glyphs from the Arabic script
- By excluding the similar looking Farsi code point (Alef from before) from the Arabic table we have helped to address issues such as phishing
- However what if a namespace supported both Arabic and Farsi languages
- This is where checking for duplicate names or ‘variants’ come into it...

IDN Registrations

Blocking Variants

August 20, 2013



Arabic Internet . Our Future



Checking for Duplicate Names

- Duplicate domains are domains that are considered ‘the same’ as one another
- For ASCII domains ‘the same’ is simply a case insensitive compare, e.g.
 - example.com
 - Example.com
 - EXAMPLE.com
 - ExAmPIE.com
- In this particular case this is enforced by the DNS protocol



However with IDNs...

- As discussed, there are many more cases where duplicate registrations may exist e.g.

Convention, visually confusing or historic	Non-visual reasons	Technical reasons
café.com cafe.com	۱۱۱۱۱.com 11111.com	ا.كوم (U+0627,U+0654) أ.كوم (U+0623)

- No single, simple rule can be applied, i.e. just lower casing does not help

Checking for Duplicate Names

- ASCII John's Cafe (because of convention)
 - johnscafe.com ← Sacrificing the é
- IDN John's Café (because now we can)
 - johnscafé.com
- Shouldn't the two be considered the same name?
i.e. Duplicates?

Implementing duplicates – The variant generation method

- The idea that one character is a variant of another character e.g.
 - ‘e’ and ‘é’
- When a domain is created using one representation the other representation is also considered registered or ‘blocked’
 - cafe.com
 - café.com
- This is done by ‘calculating’ all of the variants

Implementing duplicates – The variant generation method (cont.)

- This can happen at time of registration in which case all the variants are then stored for later comparisons

or

- This can happen on input to all commands to the registry (obviously very inefficient)

Implementing duplicates – The variant generation method (cont.)

- Calculating and storing duplicates introduces overhead
- Consider a name where there is only one variation of several of the characters in the name e.g.

e → é

cafeeeeeeeeeeeeeeeee.com

cafeeeeeeeeeeeeeeeé.com

cafeeeeeeeeeeeeeeeeé.com

cafeeeeeeeeeeeeeeeéé.com

.

.

cafééééééééééééééééé.com

In this fictitious case there is 2^{16} combinations i.e. 65,536 variations

Implementing duplicates – The variant generation method (cont.)

- If we have a domain name with just 32 characters in it, each with one variant we would have over 4 billion variants
- There has to be a better way!
- And there is...

Implementing duplicates – The canonical method

- Canonical representation of domain names isn't new
- ASCII domain names use the concept, its built into the protocol - lowercase
- The overall premise is that we assign each character a canonical form

What do we mean by character?

- A character, for the sake of this discussion, is a sequence of one or more code points that represents one particular component of a word.

a

is a character

ا

(single code point) is a character

ا

(multiple code points) is a character



Assigning a canonical form

- Each character is assigned a canonical form
- You can think of it as the base form of the character
- In most cases it just be the character itself
- Sometimes another code point entirely
- Sometimes nothing at all
- The actual character chosen doesn't really matter – its just a concept



Using the canonical form

- Define all canonical mappings for your zone
- Perform a simple substitution of each character for its canonical equivalent
 - This generates the canonical form of the label being registered
- Use this canonical form of the label as the unique key for the domain registration representing ALL forms of the domain name (without each of those forms having to be generated and/or stored)



Using the canonical form

- Lets assume that in our zone we allow the following characters with the canonical mappings listed:

a → a

c → c

e → e

é → e

f → f



Using the canonical form – An example

- We register the name cafe'.com and compute the canonical form

café.com → cafe.com

- The domain is café.com but the unique label is cafe. So when someone tries to register cafe.com we compute the canonical form

cafe.com -> cafe.com

- But this will NOT be allowed as a domain with that canonical label is already registered



Using the canonical form – Another example

- The name cafeeeeeeeeeeeeeeeeeee.com maps to cafeeeeeeeeeeeeeeeeeee.com → cafeeeeeeeeeeeeeeeeeee.com

as does
caféééééééééééééééé.com → cafeeeeeeeeeeeeeeeeeee.com

as does
caféééééééééééééééééééééé.com → cafeeeeeeeeeeeeeeeeeee.com
- So by storing the canonical form and checking all new registration attempts against it we have blocked all other registrations without actually having to calculate them all!



More on canonical...

- Mapping names to a canonical form is nothing new
 - Exactly what happens in existing domain name registries when we lower case names
 - Implied canonical mapping between upper case and lower case (implemented by a function)
 - Just also happens to be enforced by the DNS protocol itself



Making canonical work for Registries

- Just as we lower case the domain name provided to Registry functions such as:
 - Search
 - Domain Check / Update
 - Reserved List Matching
 - WHOIS
 - Etc.
- If we apply the canonical mapping to IDN names passed to registry functions everything just works



Benefits of using canonical

- It just works
- Its linear time regardless of the size of the domain names and desired variant configuration
- It provides speed and efficiency benefits, especially when compared to variant generation methods
- It saves space and memory
- Its a simple algorithm that is easy to implement, less error prone and easier to optimise

IDN Registration

Activating Variants - Bundles

August 20, 2013



Arabic Internet . Our Future



Why Bundles?

- Sometimes blocking is just not enough
- In some scenarios it make sense that a Registrant can make use of multiple versions of a name e.g.
 - cafe.com
 - café.com



In simple terms...

- Its the same as the generating variant model, so it has the same issues
 - If in our zone configuration we said that we wanted the following character variant 'provisioned' or used to create 'bundles'

١ → 1

- And then we registered the name

١١١١١١١١.com

- We still end up with...



Example

- The following variants to be provisioned

\\ \\ \\ \\ \\ \\ \\ \\ .com

\\ \\ \\ \\ \\ \\ \\ 1.com

\\ \\ \\ \\ \\ \\ 1 \\ .com

\\ \\ \\ \\ \\ \\ 11.com

.

.

.

11111111.com

- Which in this case would be 256 variants to be calculated, stored and provisioned in the zone file
- Canonical mappings can't help us here



Bundles (cont.)

- Character variants for blocking of registrations make all combinations important
- ... But when considering bundling.. If we look at the reason people desire variants, another option is presented



Continuing our example...

- In this case it makes sense that someone may enter either of the following domains:

\\ \\ \\ \\ \\ \\ \\ \\ .com
11111111.com

- But does it really make sense that someone would type the following domains names:

\\1\\1\\1\\1.com
\\ \\ \\ \\ 1111.com

- All combinations need to be blocked (which canonical mappings will do) , yet only two out of the 256 variants provisioned in the DNS are required.

IDN Registration

Introducing Mutal Exclusion

August 20, 2013



Arabic Internet .Our Future



Mutual Exclusion

- Mutual exclusion is not a new concept, it is used everywhere in modern-day life
- If we apply it to domain name variants we can achieve the desired behaviour e.g.

Primary Grouping	Sub-Grouping
Numerals	English Numerals e.g. 1,2,3,4,5...
	Arabic Numerals e.g. ١٢٣٤٥...



So the rule is...

- If a domain name contains any characters that are in one sub-group, it is not allowed to contain any characters from other sub-groups of the same primary group to be provisioned in the DNS
- i.e. The characters in one sub-group are mutually exclusive to the characters in another subgroup



Returning to our example...

Primary Grouping	Sub-Grouping
Numerals	English Numerals e.g. 1,2,3,4,5...
	Arabic Numerals e.g. ١٢٣٤٥...

- These are allowed:

١١١١١١١١.com
11111111.com

- But these are not:

١1١١1١1.com
١١١1111.com



Other bundling considerations

- Allowing Registrants to turn parts of a bundle off or on
 - How?
- Impacts on other services offered
 - e.g. DNSSEC
- Charging model
 - Should there be one?
- Flow on effects to accounting and reporting
 - Is a bundle of three domains one registration or three?

IDN Registrations

Why Variants Matter

August 20, 2013



Arabic Internet .Our Future

Why Variants Matter (some examples)?



- Hopefully by now you can see the ‘technical’ reasons for variants
- But why is it important that we deal with this?
- Primarily security
 - Imagine person one owned café.com and person two owned cafe.com
 - Its hard to tell on a billboard as you zoom past in your car if the Arabic name included the combining marks or not
- Also to meet user expectations
 - Users expect things to work as they use them in there daily life
 - When a Farsi speaking person hands an Arabic speaking person a business card written in Arabic script, most user wont understand why the email wont work (even though he is typing the characters as they appear on the card)

IDN Registration

Validating Local Language Rules

August 20, 2013



Arabic Internet . Our Future

What are local language rules?

- In short, they are and can be anything
 - Which Unicode code points make up the language
 - Handling of edge cases
 - ae → æ
 - ss → ß
 - Final form sigma
 - and so on
- Important that the business rule engine is flexible and customisable enough to handle these requirements

IDN Registration

Putting It All Together

August 20, 2013



Arabic Internet . Our Future

Effects on Registrars, Registrants and End Users

- It is different to ASCII domains
- Registrars have a harder job to do now
 - Interpret what the Registrant wants
 - Turn it into something remotely protocol valid (to map or not to map?)
 - Explain all of this to the Registrant
- Provide tools to Registrars
- Ensure consistent message to Registrants and end users



Many other areas to consider

- Registrars & Others understanding
- IDNA – Internationalised Domain Names in Applications
- So that leads us to....

IDNs for the rest of us

Putting It All Together

August 20, 2013



Arabic Internet . Our Future



Problems for all...

- Support for variants
 - Variants making them useful (useable?) beyond just blocking
- Domain Name slots – in applications
 - IE only shows u-label if language configured in browser
 - Hyperlinks (xn-- or u-label?)
 - Firefox only if in known good list
- Domain Name slots – in protocols
 - Email
 - Jabber
 - WhoIs / WEIRDS
 - etc



Problems for all...

- Hosting & DNS Providers (ISPs)
 - Management tools
 - Web hosting
 - Email
 - Educating end users
- Confused about how to configure ‘servers’
- Only used as websites... who know what issues are hiding?
- Digital certificates?
- People are still confused by the standards – even those that created them!!!
- Variants



Arabic Internet . Our Future

dotShabaka Registry
Dubai Marina, Dubai
www.dotshabaka.com
[@dotshabaka](https://twitter.com/dotshabaka)

شبكة نقطة
+٩٧١٤٤٥٣٢٧٦١
مارينا دبي ردي
نقطة شبكة إمارات
[@dotshabaka](https://twitter.com/dotshabaka)