هيئة الإتصالات وتقنية المعلومات
Communications and Information Technology Commission

Proposed Solution for
Writing Domain Names in Different Arabic Script Based Languages

ASIWG 3rd Meeting, Cairo Nov 8-10 2008

Dr. Abdulaziz H. Al-Zoman
Director of SaudiNIC - CITC
Chairman of Steering Committee
Arabic Domain Name Pilot Project

Raed I. Al-Fayez
SaudiNIC - CITC
Chairman of Technical Committee
Arabic Domain Name Pilot Project

.sa

Agenda

- **Introduction**
- **Characteristics of A Desired Solution**
- **What we are trying to solve**
- **Requirements for Languages**
- **Requirements for Registries**
- **How to support new languages**
- **Conclusion**

.sa

Introduction
About Arabic script

- **The 2nd most widely used alphabetic writing system in the world (used by more than 43 countries)**
  - more than one billion potential users could be concerned in using Arabic script domain names.
- **Used by many languages such as: Arabic, Persian, Urdu, Turkish, Kurdish, Pashto, Swahili, …**
  - that may add or change characters to represent phonemes that do not appear in Arabic phonology.
  - A new character usually created based on modifying the basic shape of an existing Arabic character, for example, by adding more dots.
  - These additions have meaning to the new language but not to the original language.
  - Therefore, many characters would easily be confused with some other characters from other languages

Introduction
Unicode Arabic Block (5.1)

- **Subtending marks**
- **Radix symbols**
- **Letterlike symbols**
- **Punctuation**
- **Currency Signs**
- **Poetic marks**
- **Honorifics**
- **Koranic annotation signs**
- **Based on ISO 8859-6**
- **Addition for early Persian and Azerbaijani**
- **Points from ISO 8859-6**
- **Combining Madah and Hamza**
- **Other combining marks**
- **Arabic-Indic digits**
- **Archaic letters**
- **Points**
- **Extended Arabic letters**
- **Extended Arabic letters for Parkari**
- **Eastern Arabic-Indic digits**
- **Signs for Sindhi**
- **Additions for Khowar**
- **Additions for Torwali**
- **Additions for Burushaski**
- **Additions for early Persian**

298 Code points

Introduction
Arabic Script IDN Major Issues

- **Acceptable/disallowed characters**
  - IDNA200x table (Pvalid /Disallowed /ContextO)
  - Language tables
- **Non-spacing Marks**
  - Subtending Marks (U+0600 – U+0603)
  - Honorifics (U+0610 – U+0614)
  - Koranic annotation signs (U+0615 – U+061A, U+06D6 – U+06ED)
  - Points (U+064B – U+0652, U+0670)
  - Combining Maddaa and Hamza (U+0653 – U+0655)
  - Other combining Marks (U+0656 – U+065E)
- **Confusing similar characters (e.g. variant tables)**
- **World/label separators (space, ZWNJ, ZWJ, hyphen)**
- **Bidirectional**
- **They are addressed at different levels:**
  - IDNA protocol level
  - Registry level
  - Application level

Introduction
Registry Challenges!

- **Security issues (stability, trust,…) e.g. phishing**
  - They should be addresses at language level first
- **Not all Arabic-script languages are ready:**
  - Not widely/commonly used
  - Language community are not ready
- **Hard to make decisions on behave of other language communities**
- **Pressure to start with ready languages**
- **Many problems have been escalated from the protocol to be handled by the registry (e.g. variants, bundling ..etc)**
- **… and yet has to provide a simple and transparent registration services**

Introduction
Who is ready?

- **Some language communities are somehow ready (alphabetical order):**
  - Arabic
  - Jawi
  - Pashto
  - Persian
  - Urdu
  - . . .

Introduction
Main problem: Security!

input[0] = U+06a9
input[1] = U+0644
input[2] = U+06cc

input[0] = U+0643
input[1] = U+0644
input[2] = U+0649

Characteristics of A Desired Solution

- **Based on standardized (or agreeable) policies and procedures that are documented on RFC-like or Best-Practice documents**
- **Extendable to allow for adding new languages as they become ready**
- **Easy and fast to be deploy by any registry**
- **Work for both ccTLDs and gTLDs**

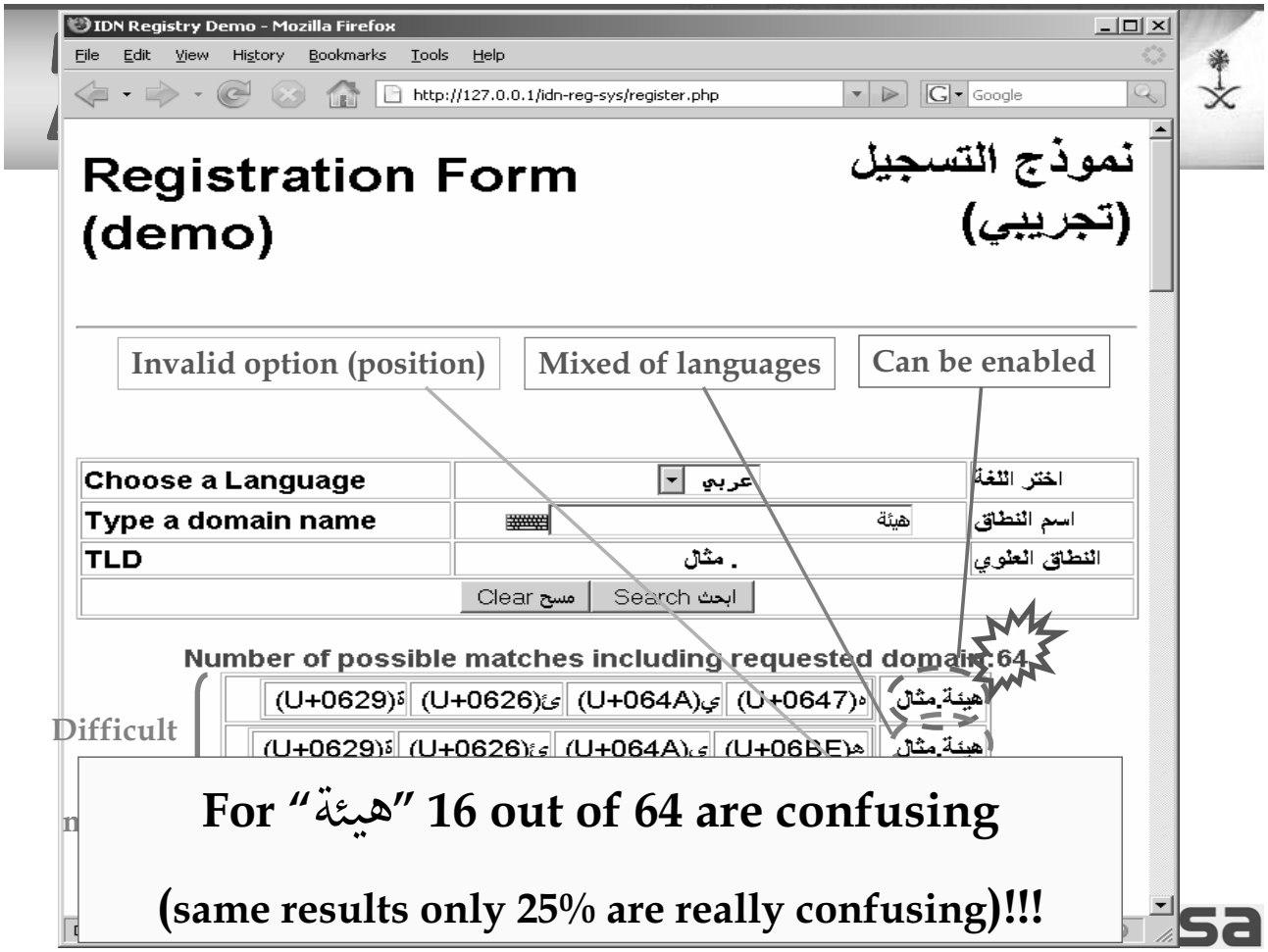
Why we need this?

- **Optimized solution**
  - Only block domains that are **really** confusing to the end users
- **Simplify Registry operations**
  - Fast and accurate Whois service
  - Simple registration & activation services
- **Transparent to end users (registrant and navigator)**
  - Keep it simple & similar to what they used to
- **Automate expandability**
  - No need to meet and discuss the same issues again when a new language is ready!

Why we need this? Example: Optimize the solution

- **If some one want to register "هدهد"**
  - Assuming there are 3 variants for the letter "ه"
  - There will be 16 possible ways to write it
    - Only **4** of them may confuse the end users (25%)
  - So why we block/bundle 75% not confusing domains?

هدهد	هدهد	ہدهد	ەدهد
هدهد	هدہد	هدەد	هدهد
هدہد	هدەد	ہدهد	ہدہد
ہدەد	ەدهد	ەدہد	ەدەد



Why we need this? Simplify the registry operation

- **How to know if a domain name have an already registered variant?**
  - Assuming we have define variants based on the inputs of current 3 ready languages;
  - There are **16,384** possible variants to write the domain "هيئة-الاتصالات-وتقنية-المعلومات"
  - Are we going to store/bundle all of them?
  - Dose the Whois function supposed to search in all registered domains & all possible variants for each one of them?
    - Not possible: Time & Resource consuming!!!!
  - Need alternative way to handle this issue (Master-Key)

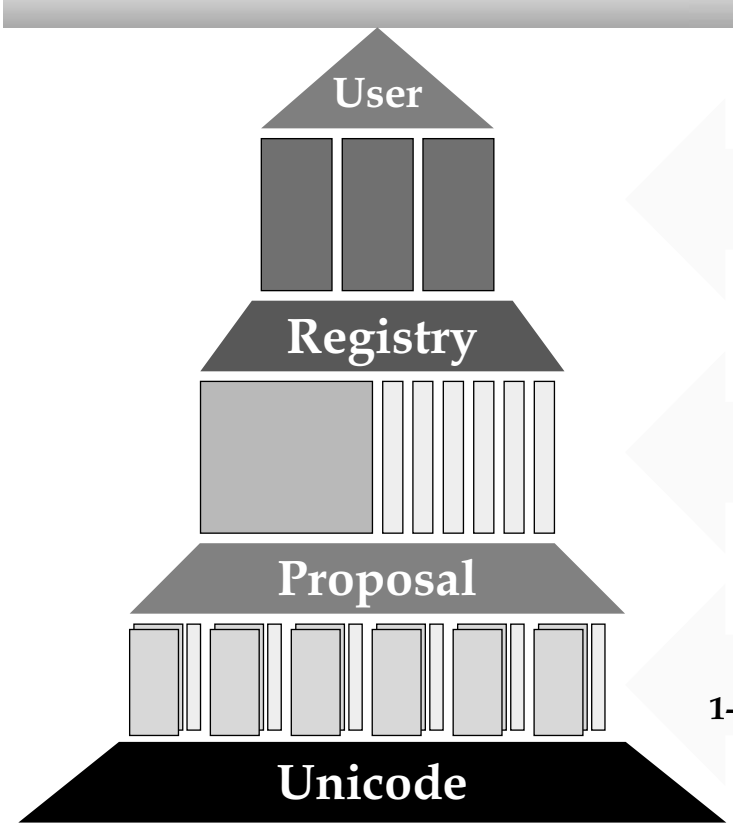
Why we need this? Transparent for the End users

- **Do not annoy/confuse end users with technical/special terms:**
  - Type & Exact Variants
  - Register & Bundle & Activate
- **Regular users should be able to register what he can type using his keyboard**
  - Advance users may seek for other options/solutions to fulfill their needs.

Why we need this? Automate expandability

- **Extendibility against**
  - Adding new languages
  - Updating existing language tables & variants lists
  - Handling Unicode updates (e.g. Unicode 5.2)
- **Define set of rules and algorithms that Registries can use to operate TLD using Arabic Script**
  - Rules for storing some structures for backward compatibly

General overview



Registry functions
(Whois + Register + Activate)

Group variant table +
Language tables

For each language:
1-Language table & 2- variant tables

Language Requirements

- **Language Table**
  - A simple code point set table that includes ONLY needed letters, digits or any other code points used in the language
- **Variant tables**
  - Exact Variant Table (EVT)
    - A comprehensive variant table that includes all codepoints (from the whole script) that are confusingly similar (exact match/mirror image)
  - Typo Variant Table (TVT)
    - A comprehensive variant table that includes all codepoints (from the whole script) that may confuse end users (e.g. typo/style match)
  - Note: A variant table will consist of a list of records, each record contains the following information:
    - Basic character codepoint,
    - List of other variant codepoints (from the rest of the script),
    - Each with position of similarity [**I**nitial, **M**edial, **F**inal, **I**solated]

Language Requirements
Examples of variants

- **Example Exact Variant match**

Representative shape in code chart	Possible shapes in context [Standalone, End, Middle, Beginning]			
ف 0641 FEH	ف	ـف	ـفـ	فـ
ف 06A7 QAF WITH DOT ABOVE	ف	ـق	ـقـ	قـ

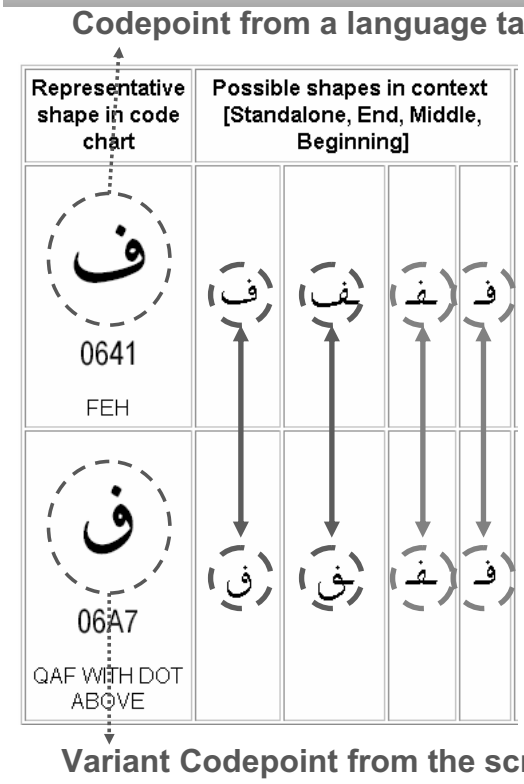
- **Example for Typo Variant match**

Representative shape in code chart	Possible shapes in context [Standalone, End, Middle, Beginning]			
ف 0641 FEH	ف	ـف	ـفـ	فـ
ف 06A7 QAF WITH DOT ABOVE	ف	ـق	ـقـ	قـ

Language Requirements
How to build a variant table?

- **Steps : (done for each basic character)**
  1. List all possible shapes for the basic character
  2. Search for all its variants from the rest of the Arabic script
  3. Then compare the basic character with its variants in all possible positions.
  4. Generate the "similarity position" bits (see next example)
- **Do it in 2 rounds:**
  - 1st round for building Exact variant table (EVT)
  - 2nd round for building Type variant table (TVT)

Language Requirements
Example: Position of similarity



Compare

1st Round (Exact)

Calculate similarity position

S	E	M	B
0	0	1	1

= 1+2 = 3

ف = ف (3)

0641 06A7

2nd Round (Typo)

Calculate similarity position

S	E	M	B
1	1	0	0

= 8+4 = 12

ف = ف (12)

0641 06A7

Exact Variant Table

```
Arabic.evt
# 1.1 Letters
0621;
0622;
0623;
0624;
0625;
0626;
0627;
0628;
0629;
062A;
062B;
062C;
062D;
062E;
062F;
0630;
0631;
0632;
0633;
0634;
0635;
0636;
0637;
0638;
0639;
063A;
0641; 06A7(3)
0642;
0643; 06A9(3)
0644;
0645;
0646; 06BA(3)
0647; 06BE(2), 06C1(8), 06D5(12)
0648;
0649; 06CC(12)
064A; 06CC(3)
```

Typo Variant Table (TVT)

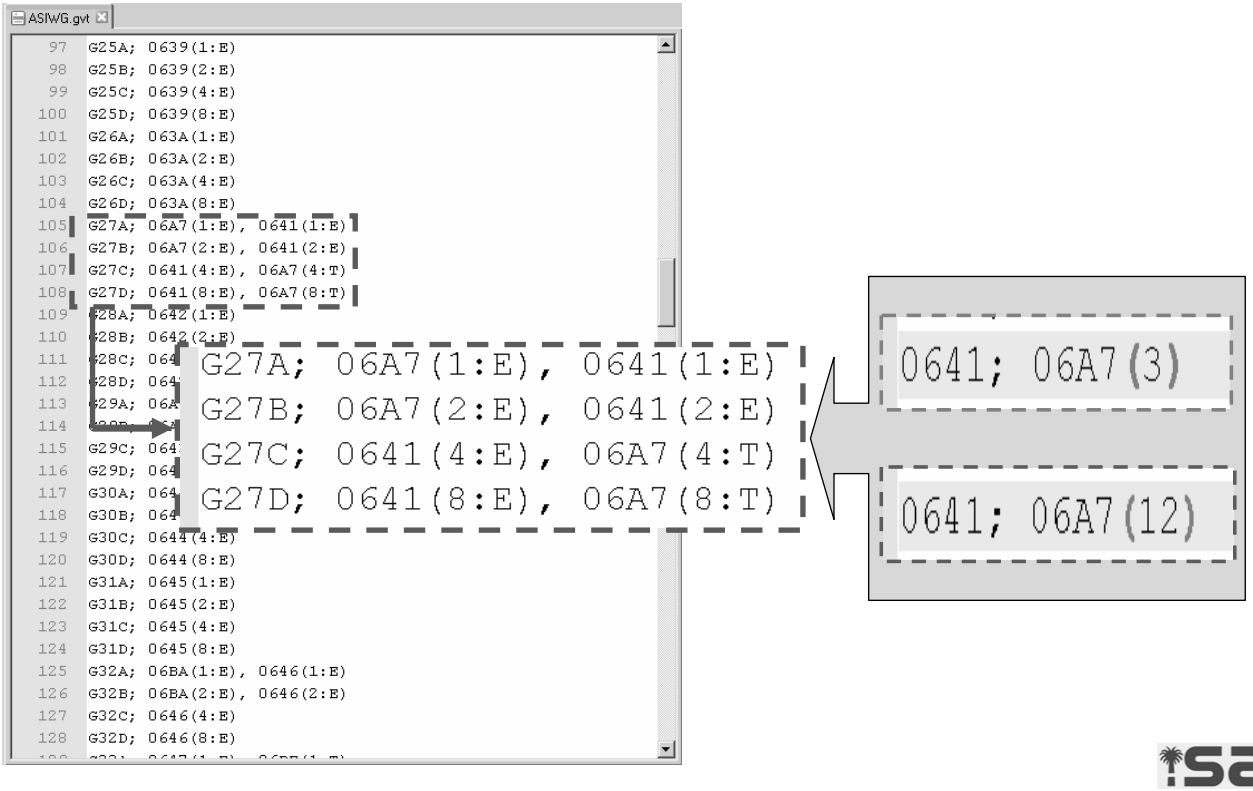
```
Arabic.tvt
# 1.1 Letters
0622;
0622; 0671(12)
0623; 0672(12)
0624;
0625; 0673(12)
0626; 06D3(12)
0627;
0628;
0629; 06C3(4)
062A;
062B;
062C;
062D;
062E;
062F;
0630;
0631;
0632;
0633;
0634;
0635;
0636;
0637;
0638;
0639;
063A;
0641; 06A7(12)
0642;
0643; 06A9(12), 06AA(15)
0644;
0645;
0646;
0647; 06BE(13), 06C1(6)
0648;
0649; 06CD(12), 06D2(12)
064A; 067B(15), 06D0(15)

#  1.2 Digits
```

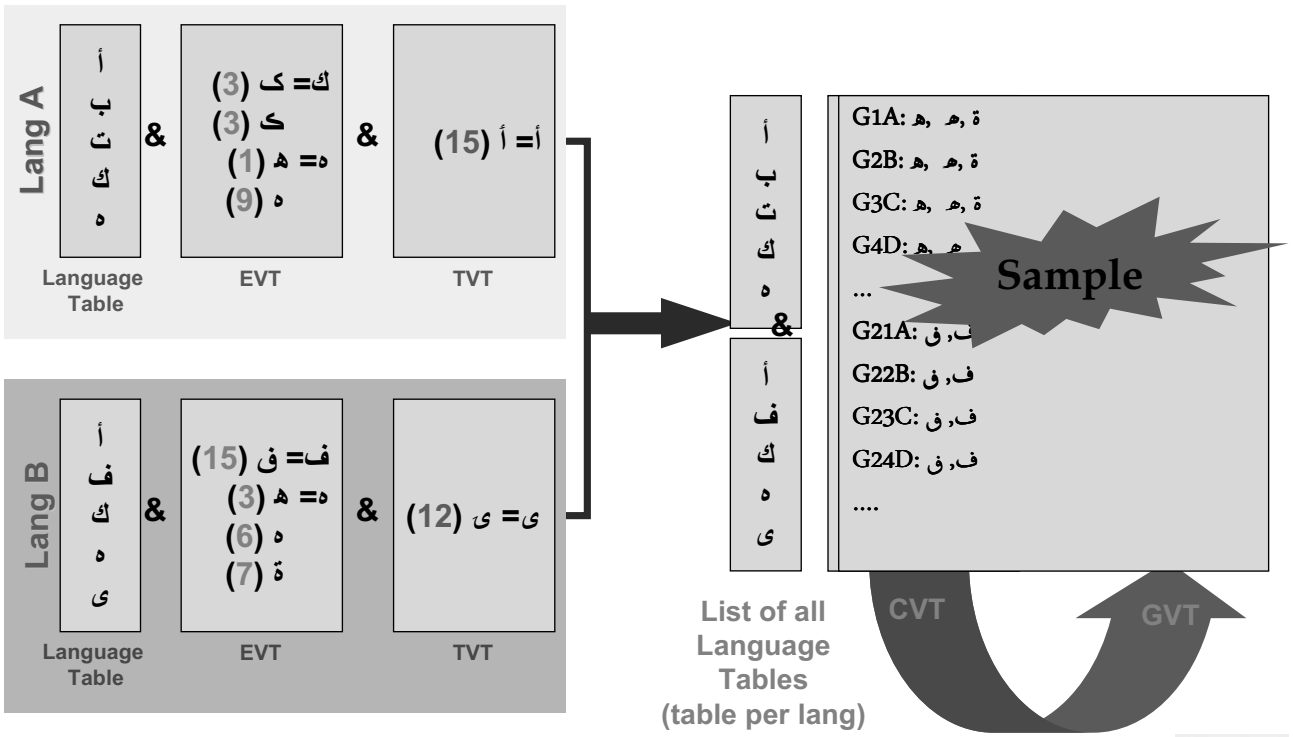
Registry Requirements

- **Language Tables (one for each supported language)**
  - Users can only register domains using code-points from only one language table
- **Group Variant Table (GVT):**
  - To be generated from the variant tables.
    - It combines all variant tables into one table that group codepoints which are similar in a given position.
    - Each group will be assigned a group-key (master key) that uniquely identify the group and will be used to map all string variants to a unique string.

Registry Requirements
Example: Group variant table



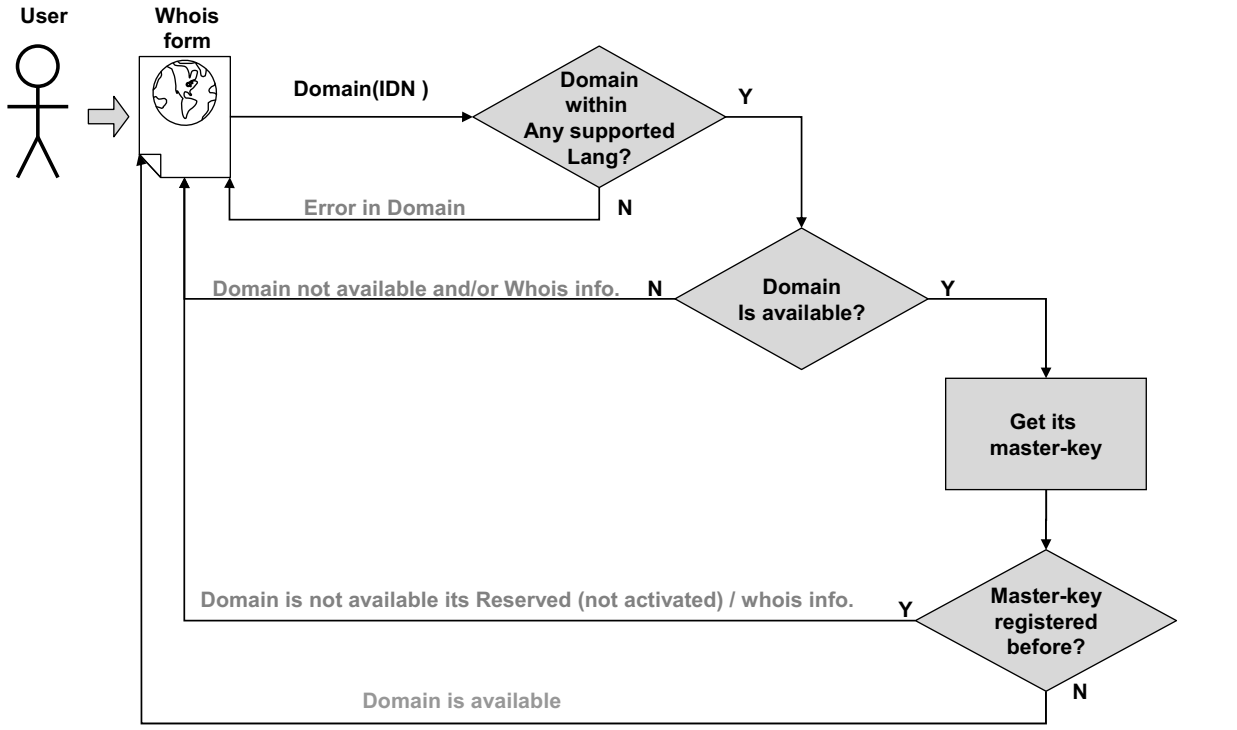
Registry Requirements
Example: Combing the languages work



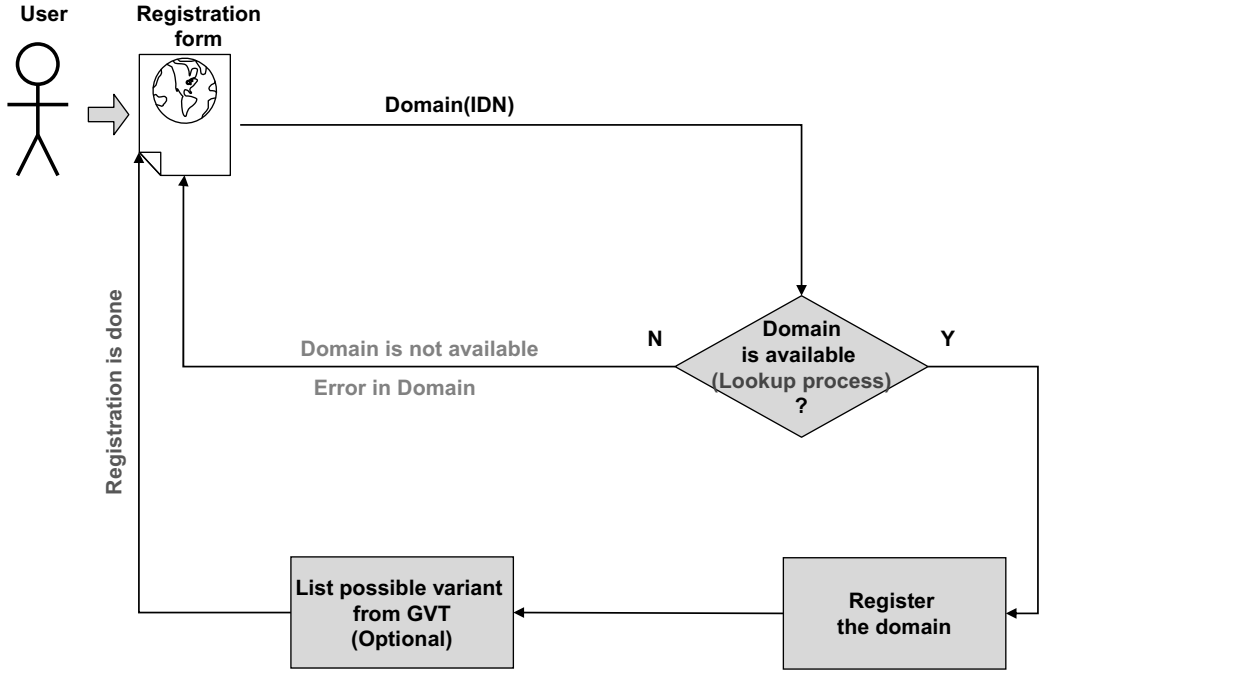
Registry Requirements
Registry-Registrant interface

- **Lookup process (whois)**
  - Check domain syntax
    - Check if the domain within any supported language
  - Check if the same domain is available or not
    - Check if the request domain exists
    - If it is found return the unavailable/whois-information; otherwise continue
  - Check if the domain is variant for a registered domain name (master-key)
    - Get the master- key for the string (based on GVT)
    - Check if the master- key was registered before or not
    - If master- key is found return unavailable/whois-information; otherwise return domain is available
- **Registration process:**
  - Registrant should select one of these languages and type a domain (U-Label)
  - Registry should accept inputs based on the selected language table
  - If domain name is register-able (available based on Lookup process)
    - Register the domain name along with its original image
    - List of allowable exact variants (to be activated if needed)
- **Activation process (enable exact variants)**
  - Original registrant can activate exact variants from his registered domain
    - List possible **Exact** variants that can be typed using one of the supported languages without intermingling between languages tables and taking care of position similarity (suggestion)
    - Activate one/many of the previous variants (if not activated before)

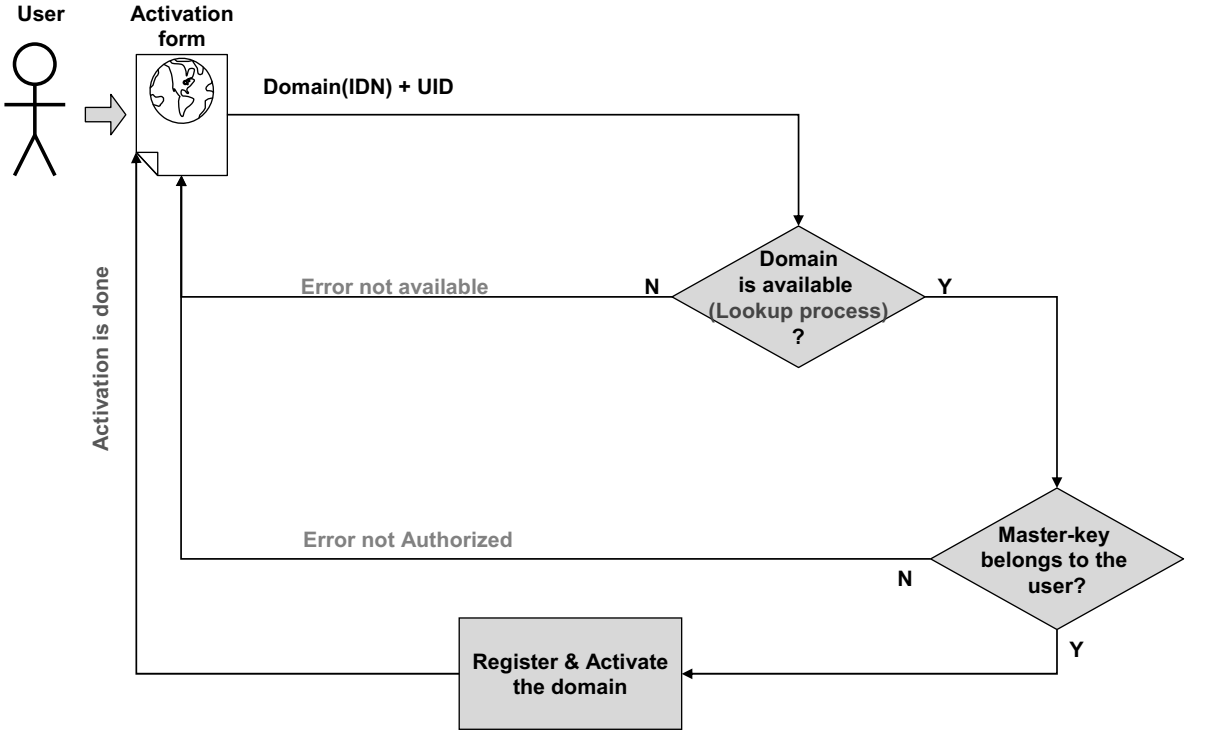
Registry Requirements
1. Lookup process (whois)



Registry Requirements
2. Registration process



Registry Requirements
3. Activation process



Steps to support a new language

- **Need process owners !!!!**
- **Language community should prepare**
  - Language Table
  - Exact Variant Table (EVT)
  - Typo Variant Table (TVT)
- **Support the new language:**
  - Add it to the list of supported languages
  - Add their Language table
  - Regenerate GVT
- **Communicate it with others (ICANN, Registries ..)**

Conclusion

- **We tried to have a prototype that fulfill the concepts of script based registry that is:**
  - Optimized, Simple, Transparent, Automated

- **Next steps:**
  - Prepare the Language tables & variant tables for all members.
  - We need more help to put the proposal in an RFC or best-practice document and communicate it with others

Acknowledgment

- **SaudiNIC**
  - Abdulaziz Al-Zoman
  - Anas Assiri
  - Mohammed Al-Hamed
  - Raed Al-Fayez
- **Outside consultant**
  - Mohamed Al-Abdulakareem (KSU)

Thank you

شكرا

xn--mgbti4d

input[0] = U+0634
input[1] = U+0643
input[2] = U+0631
input[3] = U+0627

شكرا

xn--mgbti28b

input[0] = U+0634
input[1] = U+06a9
input[2] = U+0631
input[3] = U+0627