Arabic Script Internationalized Domain Names

Sarmad Hussain, Nayyara Karamat, Arfan Mansoor
Center for Research in Urdu Language Processing
National University of Computer and Emerging Sciences, Lahore Pakistan

1. Introduction

With the increasing support of standards like Unicode, Common Locale Data Repository, ISO 639 and others, and development of internationalized architectures for applications, it is now possible to deploy contents in local languages on the Internet. However, it is still not possible to access them without knowledge of Latin script because the Domain Name System (DNS) is based on a subset of 7-bit ASCII standard (specifically the letters, digits and hyphen, LDH). Thus, it is not easily possible to encode non-ASCII languages, which would require the 16-bit or 8-bit encoding like Unicode.

An earlier attempt was made to enable domain names in other languages, referred to as Internationalized Domain Names (or IDNs). This proposal, called Internationalized Domain Names in Applications (IDNA) is an accumulation of RFCs 3454 [2], 3490 [3], 3491 [4], and 3492 [5], and is also referred to as IDNA2003. IDNA2003 adds a layer between the DNS and the client, which takes the domain name in a local language as input, normalizes it through the *nameprep* process [4], and converts this non-ASCII string to an ASCII Compatible Encoding (ACE) known as *Punycode* [5]. An algorithm *toASCII()* is described in [3] for this conversion. The reverse of the algorithm *toUnicode()* converts the Punycode back to Unicode, where required. The DNS continues to receive input in ASCII, and thus there are no changes in DNS infrastructure [1].

IDNA2003 has had some issues and was thus re-evaluated by IAB for potential problems and recommendations were given for improvement [6]. Many issues were identified, e.g. the dependence of standard on a particular version of Unicode (i.e. Unicode 3.2) even though currently Unicode 5.2 is applicable. On the basis of these recommendations a formal revision of IDNA protocol is being developed, called IDNA200X [7]. In this revision, a process is defined on the basis of characters properties in the Unicode standard, which makes it version independent, and thus operable with current and future versions of Unicode.

Though IDNA200X defines a protocol for all characters in Unicode, and thus for Arabic script as well, it still stops short of addressing many issues which are particular to these scripts, still requiring additional details for actual implementation. This paper looks at the IDNA200X recommendations in the context of Arabic script, identifies potential issues or limitations of IDNA200X in this context, and describes additional measures which need to be taken to ensure a cohesive, secure and user-friendly experience for the potential users of Arabic script IDNs. The paper also proposes and architecture to resolve these issues. This model can be extended to any other script, which is used for multiple languages, e.g. Cyrillic and Devanagari, etc.

2. IDNA200X for Arabic Script: Description and Issues

IDNA 200X uses an algorithm based on character properties defined by the Unicode standard and generates one of the four values for each of the encoded character: PVALID, DISALLOWED, CONTEXTJ and CONTEXTO characters. PVALID characters (short for Protocol Valid) are the characters which are allowed in the IDNs. DISALLOWED characters cannot be used in the IDNs. CONTEXTJ/CONTEXTO characters can be used in IDNS but with additional context rules.

Appendix A gives the status of Arabic characters, as generated through this algorithm. As can be seen from this list, the algorithm gives a very generic solution, allowing most characters and marks. However, it does not allow punctuation

marks and other similar characters. It assigns some of the other symbols contextual status. As this analysis is based on an algorithm, it still is not sufficient for defining a complete solution. The following classes of issues still need to be explicitly addressed for IDNA200X for Arabic script:

- Many of the PVALID/CONTEXT characters will not be used by any of the language using Arabic script due to the nature of the character. In addition, different set of characters are used by different languages.
- Some DISALLOWED characters are needed for IDNs.
- Some characters have confusably similar glyph shaping, but all these are PVALID.
- There are also some characters which exist in their composed and decomposed form. Both versions are PVALID.
- Arabic script is cursive and if words are written without spaces between them, their legibility is not possible. However, space is not an allowed character in the DNS within the LDH scheme.
- DNS uses the dot "." as the label separator, but some languages using Arabic script use alternated separators, also encoded in the Unicode. These separators are not allowed.

These issues are discussed in more detail below.

2.1. Algorithmically Generated Status of Different Characters

There is already discussion on the status of various characters by Arabic Script IDNs Working Group (ASWIG, archives available at http://lists.irnic.ir/mailman/listinfo/idna-arabicscript). Based on the discussion between the members of ASIWG, the following changes are required.

- Change the following characters from CONTEXTO to DISALLOWED:
  - Punctuation and associated marks and thus will not be used in IDNs: 0600-0603
  - Quranic marks and thus will not be used in IDNs: 06DD
- Change the following from PVALID to DISALLOWED:
  - Stylistic or Quranic marks and thus will not be used in IDNs: 0615, 0640, 06D6-06DC, 06DF-06E8, 06EA-06ED, FE73
- Change from DISALLOWED to PVALID:
  - Glyphs representing words required for Sindhi: 06FD, 06FE

2.2. Confusable characters

As the Arabic script is cursive, it has context sensitive shaping. Each letter can take up to four logical shapes: initial, medial, final and isolated. There are many characters in the Arabic block of the Unicode standard which are visually similar in one of more of these contexts. This visual similarity arises for various reasons, not in the scope of current discussion. Presence of such characters can be a major source of confusion and can thus be a potential source of spoofing. These confusable characters can be further divided into three categories.

2.2.1. Characters with same shape, not distinct in any language

There are some characters which have exactly the same shape and will be confusable for any language using these characters. For example the codes U+0649 (ى) and U+06CC (ى) have same isolated shape, whereas U+0643 (ك) and U+06A9 (ک) have same initial and medial forms, even though they have different isolated forms. A complete list is given in Table 1 below.

Table 1: Confusable Characters in Arabic Script[1]

Characters	Unicode	Forms (order from Left to right: Isolated, Initial, Medial, Final)	Remarks
ك ، ک	i) U+0643 ii) U+06A9	i) ك, ک , ک , ك ii) ک , ک , ک , ک	Initial and medial forms are same
ه ، ہ ، ھ	i) U+0647 ii) U+06C1 iii) U+06D5	i) ه, ھ , ھ , ه ii) ہ, ہ , ہ , ہ iii) ه, ه	Isolated forms of all these are same, and U+0647 and U+06D5 have same final forms
ھ ، ہ	i) U+0647 ii) U+06BE	i) ه, ھ , ھ , ه ii) ھ, ھ, ھ , ھ	Initial forms are same
ى ، ي	i) U+06CC ii) U+064A	i) ى, ىٔ, ىٔ , ى ii) ي, ئ, ئ , ي	Initial and medial forms are same
ى ، ى	i) U+06CC ii) U+0649	i) ى, ىٔ, ىٔ , ى ii) ى, ى	Isolated and final forms are same
ف ، ف	i) U+06A7 ii) U+0641	i) ف, ف , ف , ف ii) ف , ڤ , ڧ , ف	Initial and medial forms are same
ة ، ۃ	i) U+0629 ii) U+06C3	i) ة ii) ة	Isolated forms of both these are same
ٹ ، ٹ	i) U+06BB ii) U+0679	i) ٹ, ڑ, ڑ, ٹ ii) ٹ , ڑ, ڈ , ٹ	Initial and medial forms are same
ث ، ث	i) U+06BD ii) U+062B	i) ث, ث, ث , ث ii) ٹ, ڈ, ڈ , ث	Initial and medial forms are same
٠, ٠	i) U+0660 ii) U+06F0	i) ٠ ii) ٠	Both have same shape
١ ، ١	i) U+0661 ii) U+06F1	i) ١ ii) ١	Both have same shape
٢ ، ٢	i) U+0662 ii) U+06F2	i) ٢ ii) ٢	Both have same shape
٣, ٣	i) U+0663 ii) U+06F3	i) ٣ ii) ٣	Both have same shape

[1] Some characters are not fully supported by the font used and thus not showing the shape variation

٥, ۵	i) U+0665 ii) U+06F5	i) ٥ ii) ۵	Both have same shape
٧, ۷	i) U+0667 ii) U+06F7	i) ٧ ii) ۷	Both have same shape
٨, ۸	i) U+0668 ii) U+06F8	i) ٨ ii) ۸	Both have same shape
٩, ۹	i) U+0669 ii) U+06F9	i) ٩ ii) ۹	Both have same shape

2.2.2. Characters with similar shape, distinct in some languages

There are some characters which are used only in some particular languages and their shape can be confusable for users of other languages who do not recognize the character. For example, U+06A9 (ک) and U+06AA (ڪ) may be distinct letters of Sindhi but they are confusing for Arabic speakers. Similarly, for Urdu speakers, U+064A (ي) may seem similar to U+06CC(ی), though they are considered different characters in Pashto.

Table 2: Confusable Characters in Arabic Script[2]

Characters	Unicode	Forms	Remarks
ک ، ك ، ک	i) U+06A9 ii) U+0643 iii) U+06AA	i) ک , ک , ـک , ک ii) ك, ک , ـک , ك iii) ک, ک, ـک , ک	U+06A9 and U+06AA look shape variants in Urdu and Arabic, but for Sindhi these are separate characters
ی ، ی ، ـی ، ي	i) U+06CC ii) U+0649 iii) U+06CD iv) U+064A	i) ی, ـی , ـیـ , ی ii) ی, ی iii) ـی iv) ي, ـي , ـيـ , ي	U+06CC has similar shape to U+0649 in Urdu and U+06CD. However they are they are considered different in Pashto.
أ ، أ	i) U+0623 ii)U+0672	i) أ ii) أ	U+0623 of Arabic and U+0672 of Balochi and Kashmiri, both have similar shapes
إ ، إ	i) U+0625 ii) U+0673	i) إ ii) إ	U+0623 of Arabic and U+0672 of Baluchi and Kashmiri, both have similar shapes

[2] This list may be further expanded based on different languages.

2.2.3. Characters with different shape, confusable within some language

Some characters have different shapes, but still are confusable for a language. It is because these shapes are considered variant shapes of the same character instead of two different characters in that language. For example U+06D2 (ے) and U+064A (ي) are confusable for Arabic speakers even though their shapes are not similar or same in any context. This list has to be developed for each language.

2.3. Characters with Composed and Decomposed Forms

Unicode encodes some characters, which have a mark with them, in two forms: (i) composed single form is assigned a Unicode, e.g. U+0622 (آ), and (ii) the parts of the same letter are also encoded, assigned distinct codes U+0627 (ا) and U+0653 (̃) and can be composed to form the same character. Thus, the same letter can be formed in two ways in an IDN, causing potential phising problems. These multiple representations of same characters are to be normalized in the IDN process, before punycode is generated. However, there are also some cases which are not defined through the Unicode process and will need to be explicitly addressed. The normalizations required are shown in Table 3 below.

Table 3: Normalization for Arabic Script

Composed Form	Decomposed Form
U+0622 (آ)	U+0627 (ا) + U+0653
U+0623 (أ)	U+0627 (ا) + U+0654
U+0624 (ؤ)	U+0648 (و) + U+0653
U+0625 (إ)	U+0627 (ا) + U+0655
U+0626 (ئ)	U+064A (ي) + U+0654
U+0675 (اٵ)	U+0627 (ا) + U+0674
U+0676 (ۏ)	U+0648 (و) + U+0674
U+0677 (ۏ)	U+06C7 (ۇ) + U+0674
U+0678 (ئ)	U+064A (ي) +U+0674
U+06C0 (ۀ)	U+06D5 (ە) + U+0654
U+06C2 (ۂ)	U+06C1 (ہ) + U+0654
U+06D3 (ۓ)	U+06D2 (ے) + U+0654
U+0681 (ځ)	U+062D (ح) + U+0654 (ٔ)

2.4. Requirement of Space between Words for Readability

Arabic script is cursive in nature with context dependent shaping. The users normally require a space character between words to get correct shaping of characters and thus to make the words readable. Without space, the words will not be readable or may be confused with other words. However, space is not allowed in LDH scheme for the DNS. Thus, it cannot be processed as part of the punycode generation process. Thus, alternate ZERO WIDTH NON JOINER (ZWNJ, U+200C) could be employed to have correct shape of many words. However, ZWNJ is not a visible character and thus cannot be freely allowed as it may not be detected after non-joining characters (which do not join with next character). Thus, ZWNJ in allowed in IDN200X with CONTEXTJ status, which require context rules to determine valid use of ZWNJ. If ZWNJ is used without proper context rule, it may not be visible to user and create security issues.

2.5. Multiple Label Separators

DNS servers only recognize U+002E commonly called *period* or *full stop* as delimiter between different levels of the address, e.g. *www.crulp.org*. However many languages do not use a dot and have their own separators, for example Urdu uses a hyphen-like mark (U+06D4) to mark boundaries. Thus, requirement of dot to separate labels also causes issues in writing IDNs for such languages.

3. Proposed IDN Solution for Arabic Script

The discussion above indicates that the problems with IDNs need to be addressed at four distinct levels: Protocol Layer, Script Layer, Language Layer and Application Layer, as shown in Figure 1.

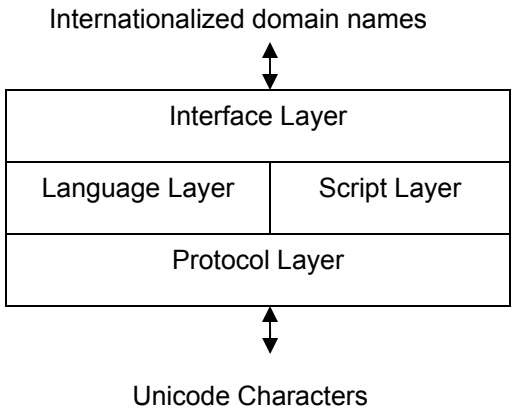

Figure 1: Filtering Layers for Arabic Script Domain Name

When characters are being investigated as being valid part of the IDNs for Arabic script, all characters to be processed must be valid in the protocol (PVALID). This is determined through this lowest level filter, being referred to as the *Protocol Layer*, or the IDN200X standard. This layer addresses the basic inclusion or exclusion of characters within Arabic script. At this level, the decision has to be as liberal as possible for letters and digits (if LDH principle is extended across scripts), but needs to be strict for punctuation and other symbols which are not needed to typically write the languages. The deviations from the current protocol decisions in Appendix A are listed in Section 2.1 above. As these changes are exceptions to the algorithmically generated solution, they need to be part of an exception list to override regular behavior within this layer.

The protocol layer still allows confusable characters, as discussed. This confusability is at two levels: the characters are sometimes confusable at script level having exactly same shape in certain contexts, or they are confusable for some particular language even if they do not have exactly the same shape. This has been discussed in more detail in Section 2.2. Thus, though the DNS allows these characters through the protocol, the registries have to decide how to limit confusable characters depending on how many languages they want to support. For generic top level domains (gTLDs), which would perhaps support multiple languages, script level decision has to be implemented. For country code top level domains (ccTLDs), which would support a more limited set of languages, language based decisions need to be taken. Thus, different filters (or tables) need to be developed for further limiting the set of characters allowed by the protocol, through a *Script Layer* or a *Language Layer* to avoid confusable characters to eventually prevent phishing and thus make the Arabic IDNs secure.

In addition to limiting characters to unambiguously meet the needs of language, a further layer needs to be added to enable the use IDNs in a user-friendly manner. This layer would also hide the limitations or the complications of IDNs caused by character encoding issues not already addressed. This *Interface Layer* would address the normalization related issues discussed in Section 2.3, enable use of space (or ZWNJ) discussed in Section 2.4 and allow other separators discussed in Section 2.5. This layer will take more "natural" input from the user's perspective, and map it onto a more protocol-sensitive form.

These layers need to be deployed at multiple points, including the application, registry and the DNS.

There is a trade-off between domain name space and its security. Developing language filters (for language and script layers) will limit the amount of domain names available in different languages and Arabic script. However, this limitation will also ensure that the Arabic domains provide a secure space for its users. If domain filters are not provided, phishing can possible occur. Also, as pointed out by ASWIG (archives available at http://lists.irnic.ir/mailman/listinfo/idna-arabicscript), limiting domain name space also gives the user an impression of broken internet if registries register just the requested domain name and block the other confusable names. This may be controlled by bundling multiple possibilities of the domain names. However, such policies are to be eventually devised by the registries, as they influence the pricing models.

4. Conclusions

This is an initial effort for defining a possible and secure solution for implementing Arabic script IDNs based on the protocol revisions being undertaken. The paper defines an overall framework, however language and script tables still need to be defined for various contexts. The script table is already being developed by ASWIG, an international committee of volunteers. However, language tables need to be defined by the language communities and the territories implementing the language-based IDNs. Efforts in that context are already underway for many languages and countries, e.g. for Arabic script domain names within the Arabic speaking countries, Persian for Iran, and Pakistani languages for Pakistan. Much more work needs to be done to finalize these tables for eventual deployment of IDNs. Work also needs to be done to incorporate the requirements at the Interface layer within end-user applications, e.g. the web browsers. However, the recommendations for different languages need to be finalized before application developers can implement them.

Much more work needs to be done to provide user-friendly and secure IDNs.

5. References

[1] D. Butt, "Internationalized Domain Names," http://www.apdip.net/apdipenote/9.pdf, APDIP.net, 2006.

[2] P. Hoffman, and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")" http://www.rfc-editor.org/rfc/rfc3454.txt, 2002.

[3] P. Faltstrom, P. Hoffman, and A. Costello, "Internationalizing Domain Names in Applications (IDNA)," http://www.rfc-editor.org/rfc/rfc3490.txt, 2003.

[4] P. Hoffman,M. B. Viagenie, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)" http://www.rfc-editor.org/rfc/rfc3491.txt, 2003.

[5] A. Costello, "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)," http://www.rfc-editor.org/rfc/rfc3492.txt, 2003.

[6] J. Klensin, P. Faltstrom, "Review and Recommendations for Internationalized Domain Names (IDNs)", http://www.ietf.org/rfc/rfc4690.txt, 2006.

[7] IDNABIS discussions available at http://stupid.domain.name/idnabis/.

[8] Hussain, Sarmad Durrani, Nadir , "Urdu Domain Names", IEEE Multitopic Conference INMIC '06, 2006.

[9] P. Faltstrom, Ed., "The Unicode Codepoints and IDNA: draft-faltstrom-idnabis-tables-05.txt", available through http://stupid.domain.name/idnabis/. Feb. 2008.

Appendix A: Decision for Arabic Script Block in Unicode (0600-06FF, 0750-077F) [9]

```
0600..0603      ; CONTEXTO
0604..060A      ; UNASSIGNED
060B..060F      ; DISALLOWED
0610..0615      ; PVALID
0616..061A      ; UNASSIGNED
061B            ; DISALLOWED
061C..061D      ; UNASSIGNED
061E..061F      ; DISALLOWED
0620            ; UNASSIGNED
0621..063A      ; PVALID
063B..063F      ; UNASSIGNED
0640..065E      ; PVALID
06EA..06FC      ; PVALID
06FD..06FE      ; DISALLOWED
06FF            ; PVALID
065F            ; UNASSIGNED
0660..0669      ; PVALID
066A..066D      ; DISALLOWED
066E..0674      ; PVALID
0675..0678      ; DISALLOWED
0679..06D3      ; PVALID
06D4            ; DISALLOWED
06D5..06DC      ; PVALID
06DD            ; CONTEXTO
06DE            ; DISALLOWED
06DF..06E8      ; PVALID
06E9            ; DISALLOWED
0750..076D      ; PVALID
076E..077F      ; UNASSIGNED
```