From Protocol to Production: Implementing the IDNs

Sarmad Hussain
Rabia Sirhindi
Nayyara Karamat

Center for Research in Urdu Language Processing
National University of Computer and Emerging Sciences

www.nu.edu.pk

www.idrc.ca

Center for Research in Urdu Language Processing
National University of Computer and Emerging Sciences
Lahore, Pakistan

Printed by Walayatsons, Pakistan

ISBN: 978-969-8961-09-1

This work was carried out with the aid of a grant from the International Development Research Centre (IDRC), Ottawa, Canada, administered through the Centre for Research in Urdu Language Processing (CRULP), National University of Computer and Emerging Sciences (NUCES), Pakistan.

Preface

Internationalized Domain Names (IDNs) are sure to extend the reach of the Internet to all the linguistic communities, and make it truly global. Development of this protocol has been a laborious process, which has taken a decade to mature, and is now being approved as this report goes for printing. The main challenges in the development of IDNs have been the need to keep the existing highly distributed Internet stable and secure as the namespace expands, while concurrently supporting a variety of writing systems, those encoded in Unicode and those which will be encoded in the future. With the deployment at hand, the near future is sure to see emerging business models, social experiments and technical solutions associated with IDNs.

The current report is written for the community, including registrants, registrars, registries, policy makers and other stakeholders, to give a complete perspective, especially for the implementation of IDNs. This requires organizing the community to record linguistic and cultural requirements, translating these requirements into technical and policy documents, and developing a secure and stable mechanism to deploy the technology for the end users. There are still many challenges, which will be faced as the community experiments with this new technology. However, the careful pace of the development of IDNs, due to the very cautious approach of all those involved, will surely make this journey safer. This has been the purpose of the revision of IDNA and institution of the Fast Track process.

We have learnt much about IDNs from interactions with colleagues over the past few years, which has made this work possible. In the context, we would want to acknowledge our country partners in the PAN Localization project who have helped us gather the requisite data presented in this report. We would also acknowledge the learning we have acquired through the many long and energetic discussions at Arabic Script IDN Working Group (ASIWG) meetings, during IDN Implementation Working Group of ICANN and during ICANN and IGF meetings. We are equally indebted to IDRC, Canada for the financial support provided through PAN Localization project and otherwise, which has assisted travel to ICANN and IGF meetings, and enabled the technical collaboration of the partner countries in the project. We would also like to acknowledge the ICANN Fellowship program which has supported our travel to ICANN meetings over past two years, and appreciate our university management who has been very encouraging in the whole process. Finally, we would like to thank Maria Ng Lee Hoon, Phyllis Lim, Phet Sayo, Adel El Azem at IDRC and Sana Gul and Ammara Shabbir at the PAN Localization regional secretariat for valuable ideas and invaluable support.

Sarmad Hussain
Rabia Sirhindi
Nayyara Karamat
(Lahore, Pakistan)

PAN Localization Project

PAN Localization Project (www.PANL10n.net) is a regional initiative to develop local language computing capacity in Asia. It is a collaboration between Pan Asia Networking (PAN) program of IDRC, Canada (www.IDRC.ca) and Centre for Research in Urdu Language Processing (www.CRULP.org) at National University of Computer and Emerging Sciences, Pakistan (www.nu.edu.pk) to generate technology, build human resource capacity, and advance policy for local language content creation, access and use across Asia.

This project has been divided into two phases. Phase-I (2003-2007) focused on developing local language standards and technology across the partner Asian countries including Afghanistan (Pashto), Bangladesh (Bangla), Bhutan (Dzongkha), Cambodia (Khmer), Laos (Lao), Nepal (Nepali) and Sri Lanka (Sinhala, Tamil). Some major milestones achieved in Phase I include development of Linux distributions for Dzongkha and Nepali, working OCR systems for Sinhala, Bangla and Lao, Lexicon and spell checking utility for Bangla, Dzongkha, Khmer, Lao and Nepali, Text To Speech System for Sinhala and standards for local keyboards, collation sequences and fonts for a number of these languages. Phase II (2007-2010) aims to advance this work, with the following objectives:

1. Examine effective means to develop digital literacy through the use of local language computing and content.
2. Explore development of sustainable human resource capacity for R&D in local language computing as a means to raise current levels of technological support for Asian languages.
3. Advance policy for development and use of local language computing and content.
4. Study and develop coherent instruments to gauge the effectiveness of multi-disciplinary research concerning the adoption of local language technology by rural communities.

Phase II of the project has been extended geographically and linguistically to include Afghanistan (Pashto), Bangladesh (Bangla), Bhutan (Dzongkha), Cambodia (Khmer), China (Tibetan), Indonesia (Bahasa), Laos (Lao), Mongolia (Mongolian), Nepal (Nepali), Pakistan (Urdu) and Sri Lanka (Sinhala, Tamil). The teams have not only been working to enhance the technology already developed in the first phase, but are also actively collaborating to deploy this technology to different end-user groups. The project has been evaluating local language computing adoption models across these countries. IDNs have also been a focus area of this research from policy, technology and deployment perspectives.

Table of Contents

1 INTRODUCTION

Internet usage has grown rapidly across the world in recent years, with 1.67 billion people[1] connected online as of year 2009. This constitutes about a quarter of the total world population. However, though the reach is formidable, three-quarters of the world population still remains unconnected. Specifically in the Asian context, though the number of users are more than any other continent (704 million Asians), this penetration of the Internet as a percentage of population is still very low (only 18.5% of Asian population)[1]. In comparison, 74% of North Americans and 50% or Europeans are connected online[1]. One of the significant reasons behind this difference is the language barrier to online access. Most of the 2200 languages spoken by people of Asia are not represented on the Internet. There have been global efforts to address this challenge. From the inception of Unicode[2], which is striving to encode the scripts used for these languages, to the efforts by World Wide Web Consortium[3] to increase accessibility of the web, the global community has been working hard to develop standards and frameworks to support internationalization (i18n) and localization (l10n) of applications[4].

In the context of the Internet, multilingual support translates to providing (i) ability to post online content in various scripts and languages, and (ii) access the content using domain names in these languages [2]. Although both aspects are very critical for expanding the reach of the Internet, the current work focuses on the recent developments related to (ii) above. Domain names in different languages and scripts are also referred to as Internationalized Domain Names (IDNs). This report provides the background, technical details and implementation process for IDNs. Current chapter provides the introduction and the background. The second chapter provides technical details for developing the IDNs. The third chapter gives details on how the IDNs could be deployed.

1.1 WHAT ARE IDNs

At this time, despite the availability of multilingual content on the Internet, accessing it requires knowledge of Latin script. This is primarily because the address on the Internet, the Domain Name [3, 4], uses names composed only from the following sub-set of Latin characters: Letters a…z, Digits 0…9 and Hyphen, collectively referred to as LDH. Domain names are a means to access the computer

[1] Source of population and Internet penetration data: Internet Usage Statistics, http://www.Internetworldstats.com/stats.htm, accessed on 10th Oct. 2009.

[2] See www.unicode.org for details.

[3] See www.w3c.org for details.

[4] Briefly, internationalization is the process of designing applications and software products to have capability to support language-specific interfaces so that they can be adapted to specific cultural, language and local needs. Localization, on the other hand, refers to the actual adaptation of software application in a particular language (see [1] for detailed definitions). If the application has been adapted to support multiple languages, it is said to have multilingual support.

on which content is located[5], though this easy-to-use name is eventually translated into an Internet Protocol (IP) address, which is a 32-bit number used for actual addressing. As an example, the domain name www.crulp.org corresponds to IP address 98.130.35.75 (a series of four numbers each of which can vary from 0..255). This Domain Name to IP correspondence or mapping is provided by the Domain Name System (DNS) [3, 4].

DNS is a distributed database which partitions Internet hosts into domains and sub-domains and forms a tree-like hierarchical structure, where hosts appear as leaf nodes. At the top of this hierarchy is the parent or *root* domain. The root servers are distributed throughout the world [6], as shown in Figure 1.1.

Top-level domains (TLDs) exist under the root domain. These include generic TLDs (gTLDs) such as com, org, gov, edu, etc., as well as country-code TLDs (ccTLDs) such as pk, la, mn, sa, ca etc. For example, reading from right to left, the domain name www.crulp.org. has the following information:

. represents the root domain
.org represents the gTLD for organizations
.crulp represents the second-level domain for the organization CRULP
www represents the actual host on which CRULP's website is hosted

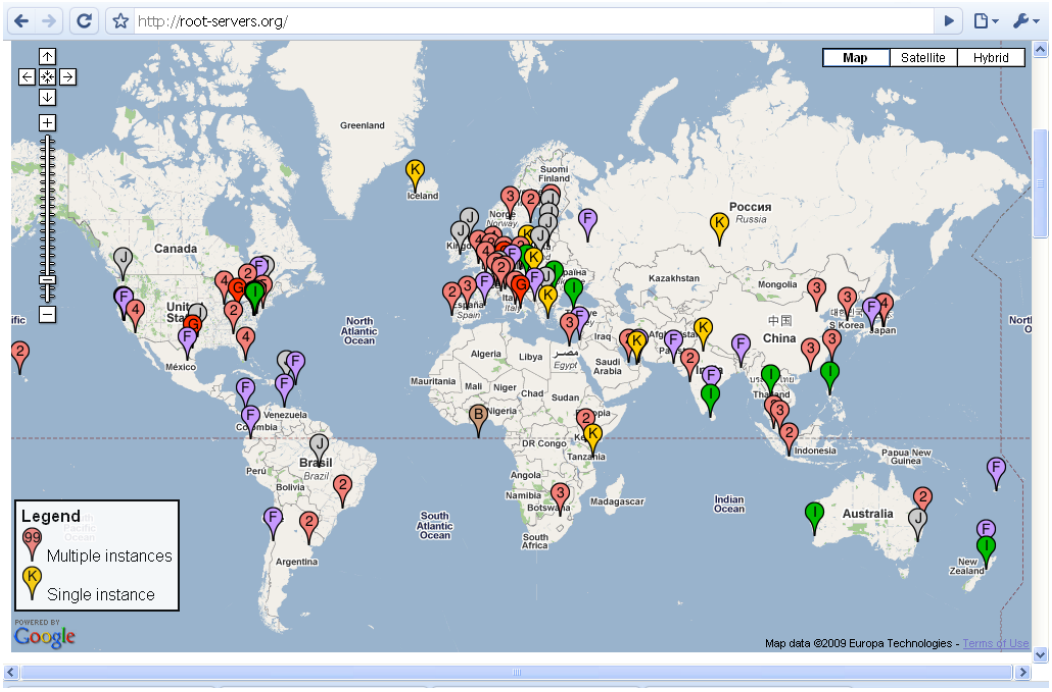


Figure 1.1. Root Servers (accessed from root-servers.org on 10th October)

[5] Generically, address to a web resource is referred to as a Uniform Resource Identifier (URI). It points to an object (webpage, text file, image, etc.) on the Internet. The syntax of URIs is given in RFC 3986 [5].

The IP address for a host is uniquely determined by traversing a path from the root node of the tree (representing the root server) to one of the leaf nodes (e.g. www).

Whenever a URI is encountered in a text or entered in the address bar of a browser, the DNS resolver program on the client forwards a DNS query to the local DNS server which in turn provides a URL-to-IP address mapping. If the local name server is not authorized for the domain in question, it queries a relevant name server. This resolution process may consist of a number of such query-response messages shown in Figure 1.2 and explained below.

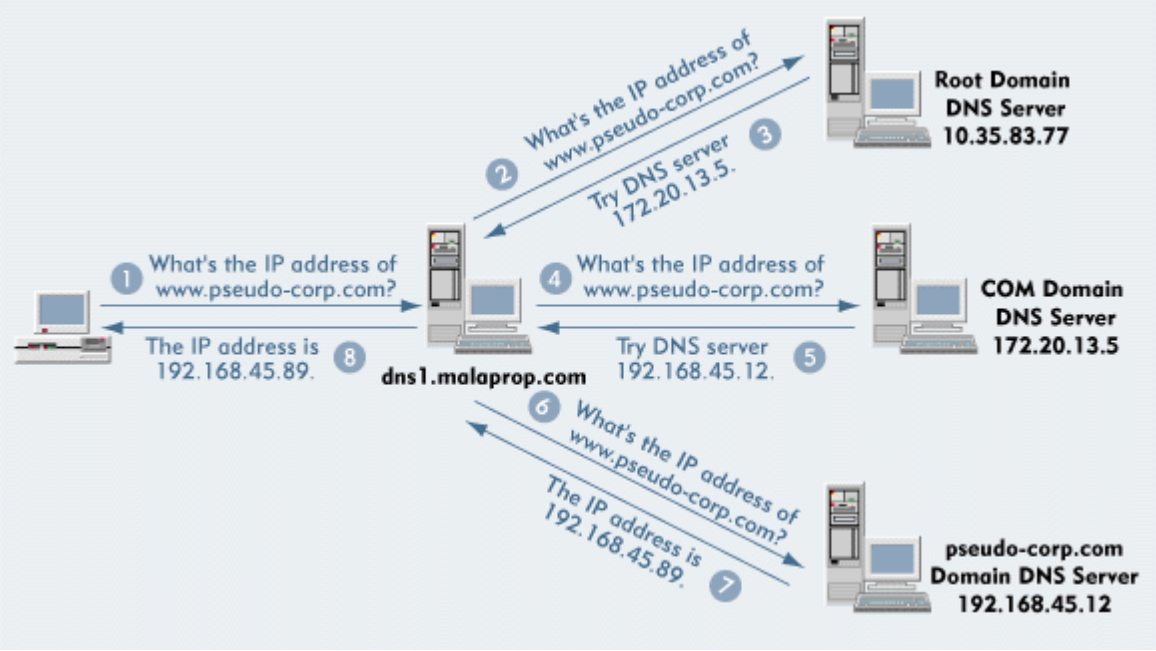


Figure 1.2. DNS Query Resolving Process (accessed from http://support.novell.com/techcenter/articles/img/nc1999_0203.gif)

1. A host on some network requests the IP address of a host named *www.pseudo-corp.com*. This first DNS query is sent to the local DNS server called dns1.malaprop.com (default name-server on the host's LAN or DNS server of local Internet Service Provider).
2. The local DNS server does not find this information (the IP address for www.pseudo-corp.com) in its zone file. In turn, it sends a query to one of the root servers at IP address 10.35.83.77.
3. The root server determines which top-level domain server is authoritative for the domain *.com*. It sends the local DNS server the address of .com name-server, which is 172.20.13.5.
4. The server dns1.malaprop.com sends the next query message to the .com name-server.
5. The .com server in response sends the IP of the next authoritative server for domain *.pseudo-corp.com* which is 192.168.45.12.

6. The local DNS server then queries the .pseudo-corp.com name-server.
7. The .pseudo-corp.com name-server contains in its zone file the name-to-IP mapping for the host *www*. This address record is sent in a DNS response message to the local name-server.
8. The local DNS replies with the IP address (192.168.45.89) of www.pseudo-corp.com to the requesting host.

To eventually enable users to access content by typing the domain name in their own language, the DNS has to be enhanced so that similar query-response process occurs when a domain name is typed in any language. As the current deployed system for resolving domain names is based on LDH scheme using 7-bit ASCII encoding, extending the process to be able to resolve domain names in other scripts (for other languages) would require a modification of the entire domain name system[6]. Owing to the highly distributed nature of DNS, this would be a very complex task and would have impact on the stability of the complete system.

In order to prevent the change of underlying DNS framework in its entirety, an application level solution has been developed to handle IDNs on the Internet. The basic idea is to convert non-ASCII domain name to an ASCII-Compatible Encoding (ACE) containing only LDH characters which is compatible with the existing DNS. This mapping to and from IDNs has to be done at the client application, and only LDH is passed "on the wire", i.e. within the DNS. As a result, the IDNs are enabled, but the core DNS system at the back remains unchanged. This system of enabling IDNs is being referred to as IDNs in Applications (IDNA) because the IDN-to-ACE mapping is done at the application layer. This is illustrated in Figure 1.3.

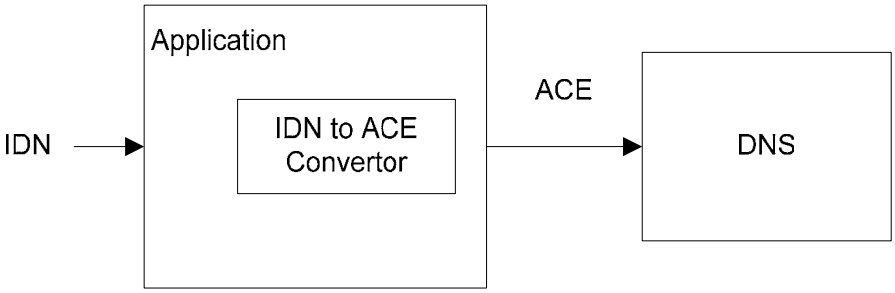


Figure 1.3. Schematic Diagram for IDNs in Applications (IDNA)

1.2 HISTORY OF IDNs

Starting from an initial draft on IDNs by Martin Durst in 1996, the practical research on IDNs started in 1998 at the National University of Singapore, followed by formation of iDNS Working Group by Asia Pacific Networking Group (APNG). The next two years saw the establishment of a number of iDNS test beds in various Asia Pacific countries including China, Japan, Thailand, Taiwan, Hong Kong and Singapore. IDN prototypes were also developed and demonstrated at different

[6] One of the reasons is that such domain names need 8-bit encoding as they use Unicode.

meetings and conferences. The first commercial solution for internationalized domain names and email addresses was created in 1999 by two private companies iDNS.net and iEmail.net. In late 1999 IETF held a meeting on IDNs. Meanwhile, iDNS.net launched the first commercial IDN system in Taiwan which used Chinese characters. In January 2000, IETF formed a working group on IDNs that was chartered to identify requirements and develop protocols for allowing the use of non-ASCII characters in domain names. This was followed by the formation of Multilingual Internet Names Consortium (MINC) in mid-2000. ICANN formed a work group on IDNs in March 2001 subsequently forming a committee on IDNs in September 2001. In 2003, IETF published standards for implementation of IDNs in the form of RFCs, collectively referred to as IDNA 2003. Due to the drawbacks in this standard identified in RFC 4690 [12], revision work on IDNs resumed and is still underway. The current IETF working group on IDNs (the IDNAbis Working Group) has published a revised set of Internet drafts called IDNA 200x[7].

1.3 CURRENT STATUS OF IDNs

The work of IDNAbis working group is to ensure the stability of the system for IDNs. Its work is documented in five Internet drafts namely Definitions, Rationale, Protocol, IDNA tables and Bidi rules. In addition, there is a sixth informational draft on mapping. The main charter of the working group is intended to separate IDNA from specific versions of Unicode using algorithms that determine character validity based on Unicode character properties [13, 20]. Other goals include separating registration-time and resolution-time processes for valid IDNs, revising bidirectional algorithms to produce a deterministic answer to whether or not a label is allowed and permitting effective use of some scripts that were inadvertently excluded by IDNA2003. This working group aims to preserve the current DNS.

The status and summary of current IDNAbis draft documents is given below[8]. The documents are in the period of Last Call for comments at time of finalization of this report. This last call will be closed in mid October 2009.

Table 1.1. IDNA 200x Documents

Draft Document Title	Posting Date	Status
draft-ietf-idnabis-defs-11	2009-09-14	In Last Call
draft-ietf-idnabis-rationale-13	2009-09-14	In Last Call
draft-ietf-idnabis-protocol-16	2009-09-14	In Last Call
draft-ietf-idnabis-tables-07	2009-09-10	In Last Call
draft-ietf-idnabis-bidi-06	2009-09-28	In Last Call
draft-ietf-idnabis-mappings-04	2009-09-03	In Last Call

[7] 'x' would be replaced by the year in which the drafts are formalized.
[8] Accessed on 11th Oct. 2009 from http://tools.ietf.org/wg/idnabis/.

1. *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework (draft-ietf-idnabis-defs-11.txt)* – provides definitions (for terms like A-label, U-label, etc.) and relevant material required to understand the rest of the documents in the set.
2. *Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale (draft-ietf-idnabis-rationale-13.txt)* – provides overview of the revised IDNA system, its relationship with the older standards and how different components work.
3. *Internationalized Domain Names in Applications (IDNA): Protocol (draft-ietf-idnabis-protocol-16.txt)* – provides the actual IDNA protocol including registration and look-up time activities to be performed.
4. The *Unicode code points and IDNA (draft-ietf-idnabis-tables-07.txt)* – provides a rule-based mechanism to determine if a Unicode code-point can be included in the list of permitted characters in IDNs. Exception lists are also tabulated.
5. *An Updated IDNA Criterion for Right-to-Left Scripts (draft-ietf-idnabis-bidi-05)* – specifies rules that labels with mixed-direction characters need to satisfy. Such bi-directional labels contain characters from both left-to-right and right-to-left scripts.
6. *Mapping Characters in IDNA (draft-ietf-idnabis-mappings-04)* – provides non-normative guidelines for operations which can be performed by applications with user input. It explains the mapping process to yield only permissible code-points to be passed to the IDNA protocol.

1.4 WHO MAKES THE IDNs WORK

A broad categorization of Internet's management tasks includes standardization of Internet protocols, allocations of Internet resources and Internet governance issues. The Internet is managed by a number of organizations having distributed roles and responsibilities.

1.4.1 INTERNET SOCIETY

The Internet Society (ISOC) [46], formed in 1992 supports the technical evolution of the Internet through seeking involvement of various scientific, academic and engineering communities, administers educational trainings and seminars all over the world and develops policies to ensure equal Internet access by people globally. ISOC also serves as an organizational home for the Internet Architecture Board (IAB), Internet Engineering Task Force (IETF), Internet Engineering Steering Group (IESG) and Internet Research Task Force (IRTF). Among these, IETF functions as the primary body chartered for the design and development of the protocols that run the Internet infrastructure. The relationship of different organizations within ISOC is given in Figure 1.4.

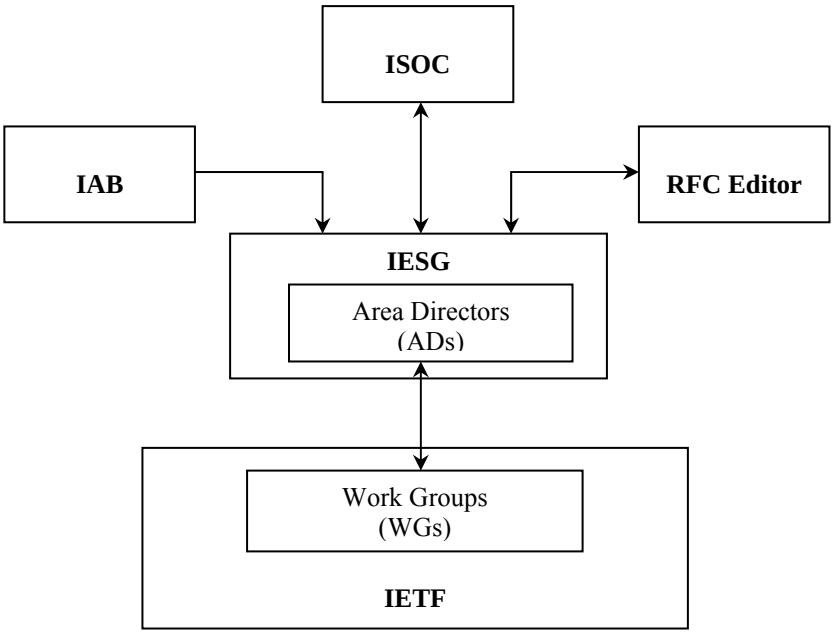


Figure 1.4. Structure of ISOC

1.4.2 IETF, IAB AND IESG

The Internet Engineering Task Force (IETF) [44] is an open organization comprising of network researchers, designers, engineers and vendors working in unison towards the evolution and smooth operation of the Internet. Its mission is to produce high quality technical documents that help improve the design, implementation, efficiency and management of the Internet [45]. IETF works in groups, where each working group is assigned a specific topic or research area relating to networks. These groups are formed in one of the eight different areas of scope of IETF, including Applications Area, General Area, Internet Area, Operations and Management Area, Real-time Applications and Infrastructure Area, Routing Area, Security Area, and Transport Area.

At the top of the hierarchy is the Internet Society [46] which provides legal and financial support to the Internet standardization bodies such as IETF. While ISOC is the public relations liaison for IETF, IAB oversees the technical activities of all IETF workgroups. It brings into public notice different issues related to the Internet architecture's integrity and consistency. It is in charge of reviewing any new workgroup that is formed under the IETF areas even before it is chartered.

It is IESG's mandate to manage individual IETF workgroups in that it authorizes adherence to the process of standardization by particular workgroups. It is involved in the chartering of work groups and reviewing their output documents as part of creating standards. IESG comprises of Area Directors, one for each of the eight specialized areas in IETF. A work group is created and its charter is approved by IESG. The working group has to adhere to its charter and develop standard documents, which are then reviewed by the IESG. Once found conformant with the

charter and tested to fulfill the requirements, the documents are published by the RFC editor.

A specialized working group for internationalized domain names was established by IETF under the Applications Area in January 2000 and has ever since been working towards the standardization of IDNs. The work group, called IDN Work Group, came up with three standards for the realization of IDNs in applications. Implementation of most of the commercially deployed IDNs is based on these standards. A new working group has been formed (called the IDNAbis) which is revising IDN protocol taking into account the deficiencies of the previous specifications.

1.4.3 INTERNET CORPORATION FOR ASSIGNED NAMES AND NUMBERS (ICANN)

Formed in 1998, Internet Corporation for Assigned Names and Numbers (ICANN) [42] is a non-profit corporation aimed at managing Internet related functions such as IP address space allocation and management, generic top-level (gTLDs) and country-code top-level (ccTLDs) domain name system management, protocol identifier assignment and root server management. This function was previously performed by Internet Assigned Numbers Authority (IANA) under the US government.

ICANN's mission is to "*coordinate, at the overall level, the global Internet's systems of unique identifiers, and in particular to ensure the stable and secure operation of the Internet's unique identifier systems*" [43]. ICANN is divided into a number of committees and supporting organizations. A few to mention are the GAC, ccNSO, GNSO, ASO and IANA.

The Government Advisory Committee (GAC) [47] provides formal representation of different countries at ICANN. GAC's role is to provide advice and guidance to ICANN regarding policy issues in naming and addressing activities as they relate to specific governmental concerns. The formation of GAC has been motivated by the need to keep Internet a global platform where different countries can provide input and feedback on secure and reliable execution of Internet's functions.

Country Code Names Supporting Organization (ccNSO) is another policy-making organization that takes into account the range of issues associated with country-code Top Level Domains (ccTLDs, e.g. .pk, .mn, .la, .ca). It is formed by ccTLD managers to provide a global forum for discussion of technical and global policy issues regarding ccTLDs.

Generic Names Supporting Organization (GNSO) [49] is a successor to ICANN's Domain Name Supporting Organization (DNSO), responsible for handling technical and policy related issues regarding generic top-level domains (gTLDs, e.g. .org, .net, .info).

Address Supporting Organization (ASO) [51] is a supporting organization formed in 1999 as a result of the collaboration of three Regional Internet Registries, APNIC, AFNIC and RIPE NCC. Its purpose is to advise ICANN on IP addressing policy.

A complete structure of ICANN is illustrated in Figure 1.5.

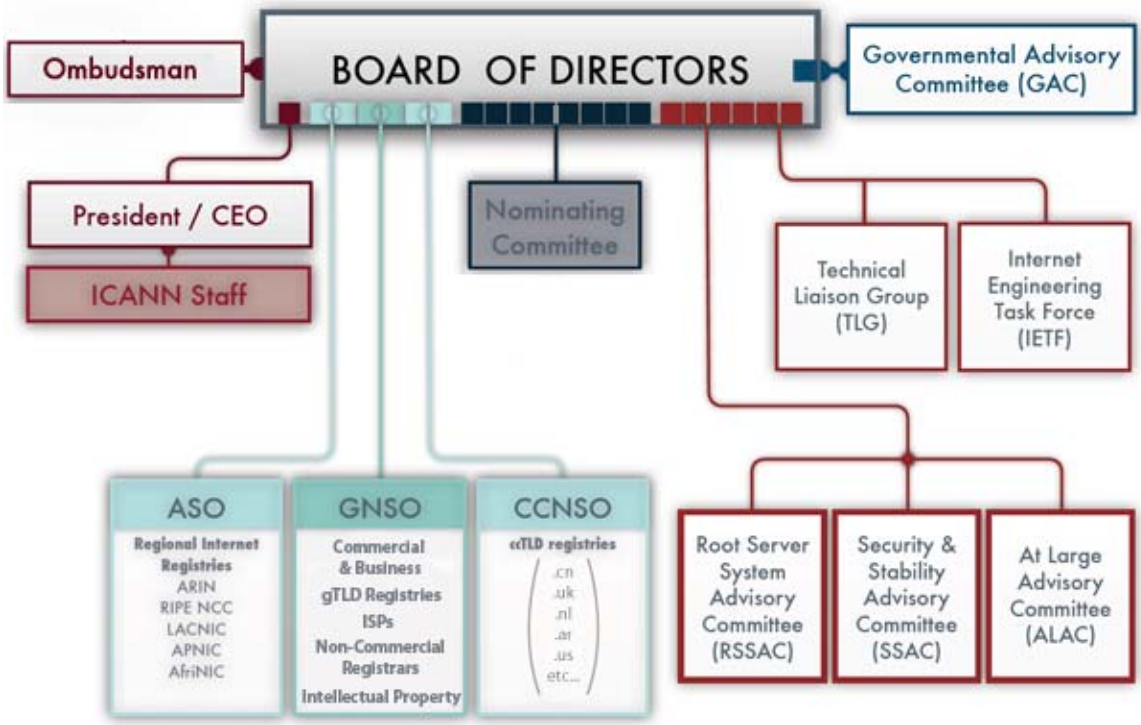


Figure 1.5. ICANN's Organizational Structure [accessed from http://www.icann.org/en/structure/]

The policy development process is not only coordinated through the review and advice of its supporting organizations, but Internet users' participation is also ensured. The At-Large Advisory Community (ALAC) [52] represents Internet users and reflects public opinions in ICANN policy development work. ALAC has liaisons with in all ICANN organizations such as ccNSO and GNSO.

ICANN takes an active part in implementation of IDNs and has published a number of reports and procedures for the adoption of IDNs and IDN TLDs. These provide a general framework for the development of registry policies for the implementation of IDNs. This work is done by developing multiple working groups.

ccNSO has established a number of workgroups which deal with specific issues including selection and deployment of IDN ccTLDs strings. The IDNC WG of ccNSO [48] was established to develop and report on viable methods for the introduction of IDN ccTLDs as the IDN policy is being formulated. The work group published its final report on 26th June, 2008, which detailed a three stage process for the

implementation of IDN ccTLDs. Currently a Joint ccNSO/GAC IDN work group is also functional. Similar work is also being done within GNSO and ALAC.

In addition to the above mentioned committees, there is a President's Advisory Committee for IDNs [50] whose purpose is to advise the board and provide a means to coordinate IDN related developments within and outside ICANN. It is also designated to identify and provide solutions to implementation and policy related challenges faced by IDNs in top-level domains.

1.4.4 INTERNET GOVERNANCE FORUM

The Internet Governance Forum (IGF) was established in July 2006 by the UN Secretary General as an outcome of the second phase of the World Summit on Information Society (WSIS) [54] held in Tunis in 2005. It is a multi-stakeholder forum to discuss Internet governance related issues such as public policy, Internet resource management, communicating with governmental organizations, exchange of information among various scientific communities, improving access to Internet in the developing world, identifying emerging issues, maintaining Internet security and countering cyber terrorism [53].

Three meetings of IGF have been held so far. One of the broad topics of these meetings has been that of the diversity on the Internet, achieved through multilingual support. This involves both creating online content in local languages and introducing multilingual addresses in the DNS so that everyone can access the Internet in their native language. The importance of internationalized domain names in this regard has been repeatedly emphasized in all the IGF meetings.

1.4.5 UNICODE CONSORTIUM

Unicode was formed in 1991, and aims to encode all the writing systems of the world. IDNs use the Unicode encoded character properties and character codes in the process of defining IDN labels, as discussed in more detail in the next chapter.

1.5 STEP-WISE GUIDE TO IMPLEMENTATION OF IDNs

IDN is an emerging standard. Though discussions on it started more than a decade ago, it is still being updated and finalized. However, having gone through the IDNA2003 iteration, and with lessons learnt [12], the revised protocol has promise for successful deployment. In the current version, in addition to the second or third level domains, it will also be possible to have new IDN gTLDs and IDN ccTLDs. This is opening an unchartered name space for the users globally. The following steps are needed to deploy IDNs. This process includes coordinated work by the user community, registries and the relevant public organizations (e.g. language authorities).

1.5.1 CHOICE OF LANGUAGE(S) AND SCRIPT(S) BASED ON THE TARGET USER COMMUNITY

A single script might be used by more than one language. Alternatively, more than one scripts can be used to write a single language. Therefore, the first step in developing IDN system for a registry is to identify the language(s) and script(s)

relevant for the target user community. The process in determining these factors should involve feedback from the relevant community. In case of IDN ccTLDs, this may also involve constitutional stipulations which give certain languages and scripts official, national or formal status within a country or a territory.

1.5.2 DETERMINING THE IDN TLD STRING AND VARIANTS

After the language(s) and script(s) have been determined, the next step requires deciding upon the IDN gTLD or ccTLD. This string should have characters from a single script and may have other restrictions based on linguistic, cultural, and technical factors, e.g. the proposed string should not be confusable[9] with existing TLDs to avoid user confusion, and should result in ACE encoding shorter than 64 characters.

Due to multiple encoding of characters in Unicode for various reasons[10], and sometimes similarity of characters within and across scripts, the proposed string may be confusable with other possible strings. All such strings are referred to as variants and should be identified in the string finalization process. These variants should not have independent delegation.

1.5.3 DEVELOPING LANGUAGE/SCRIPT TABLE

Following the first step, it also becomes necessary to determine the characters within a language and script which will be allowed in an IDN label for registration. Unicode encodes scripts and generally each script block contains many characters which are not relevant, either because they are for supporting other languages using the same script or because they are generally not allowed in domain labels (e.g. punctuation marks). A language table lists characters of a language and/or script which are allowed to be registered in IDNs for that language or script. This is an important resource in the deployment of IDNs regardless of the level of label in the DNS hierarchy. Entities involved in the registration of IDNs (registries and registrants) refer to the above table for choosing IDN strings and performing validation checks. These tables are based on both the IDNA character specification as given by the protocol and the language level requirements and preferences as defined by linguistic community.

Additionally, a language table may also contain information about confusingly similar characters. Registries and registrants can benefit from this information when performing variant analysis. Chapter 2 discusses these issues in much more detail.

1.5.4 DEPLOYMENT OF IDNS

After the IDN TLD string and relevant language/script table has been finalized, the next step involves setting up a registry for the top-level domain. This step is not relevant if IDNs are being implemented within an existing gTLD or ccTLD. It

[9] The definition of confusion is still not very clear in the process, but various groups are working on its formulation.

[10] For example, for reasons of backward compatibility.

requires three parallel processes, pertaining to planning, registration and administration of the new registry supporting the IDNs, as briefly discussed below and explained in more detail in the following chapters.

1. *Registration of TLD with ICANN* - The IDN TLD has to be registered with ICANN first. ICANN has launched Fast Track application process for the registration of IDN ccTLDs [39]. A new gTLD program is also in progress to allow registration of IDN gTLDs [57]. This adds a new entry in the IANA root zone database [41] and the new TLD is delegated to the registry for use. At the time of application, the language/script table is also submitted, which will be used by the registry. See chapter 4 for details.

2. *Domain name policy formulation* - A domain name policy lays out the terms and conditions relating to domain name registration and is part of the agreement between a registry and a registrant. In addition to updating the conventional policies relating to dispute resolution, pricing and Whois information, new policies have to be developed in context of IDN registrations. These include variant management policies for IDNs. More details on variant management can be found in Chapter 3.

3. *Establishment of the registry* - After a TLD has been registered in the root, a registration system would need to be set up. This entails developing a registration process (and an online registration system), and not only requires setting up the required servers, but also necessitates developing the scripts which validate and register (and eventually resolve) IDNs.

With this background on IDNs and the overall process, Chapter 2 provides a comprehensive analysis of technical issues relating to the preparation of IDN labels. Following the discussion, Chapter 3 explains the registration and resolution process for IDNs. Chapter 4 concludes the discussion, with focus of what is expected in the future for IDNs.

2 PREPARATION OF AN IDN LABEL

A domain name contains multiple levels. Each level is represented by a string, also called a label. Thus the domain name www.crulp.org contains three labels, *www*, *crulp* and *org* respectively. In the DNS, labels can only be formed using LDH scheme. However, it is not clear which characters will be allowed for forming labels in the context of IDNs. Every language is represented by a character set, which includes a variety of character types, including consonants, vowels, combining marks, digits, punctuation marks and special symbols. The character set is represented by a script block within Unicode standard, e.g. for Lao, Khmer and Sinhala, etc. For scripts which are used by more than one language, the Unicode standard encodes a superset of all the characters across these languages within a single script block. However, not all these characters are relevant for use in IDN labels for a language. Thus the character set has to be minimized for each language. This is a necessary first step, before IDNs can be deployed.

Even in the current DNS, the labels are limited to LDH. Other ASCII characters, e.g. punctuation marks and other symbols, are not allowed as they are not considered necessary for making domain names. Such category of characters will also be disallowed for labels in other languages, irrespective of the script. However, there are also some characters which may be undesirable not from technical perspective but from a language or cultural perspective. This latter case will vary across scripts and languages.

Thus, a layered model needs to be adopted for determining which characters from the entire Unicode script block can be used in IDNs. The lowest (baseline) is the Protocol layer, which does the core filtering as stipulated by IDNA200x standard developed by IETF. In addition, there may be characters which are allowed by the Protocol but the script community may consider them redundant and would thus filter them for all languages using the script. This has to be done at a separate Script layer. Additional restrictions may be necessary to limit the characters within a single language. This will have to be done at a Language layer, through language specific tables, as determined by the relevant linguistic community. Finally, additional mapping and conversion may be necessary at Interface layer, to facilitate users to enter characters in forms not being allowed by any of these former three layers. This logical division of the validation process into the four layers is illustrated in Figure 2.1 below[11].

[11] This model has evolved out of discussions within Arabic Script IDN Working Group (ASIWG).

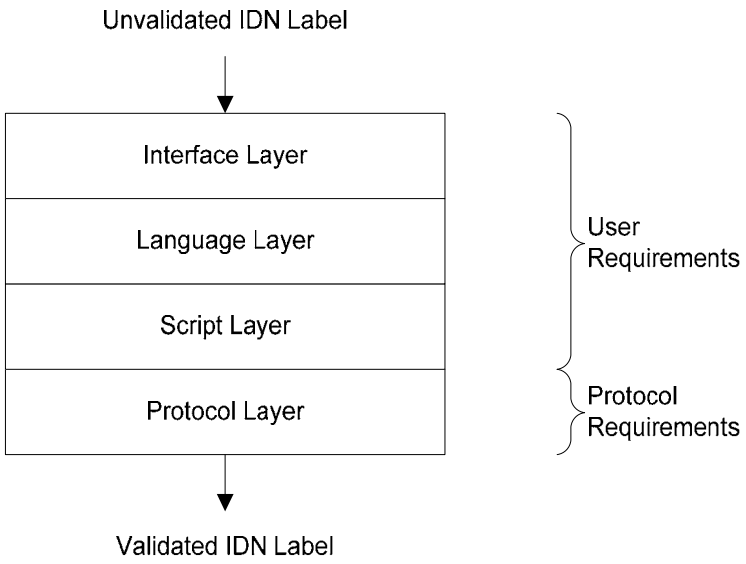


Figure 2.1. Logical Label Validation Levels

These layers can be grouped into two parts, one specified by the Protocol and the other specified by the users. The following sections discuss these two parts, with examples from various languages and scripts[12].

2.1 PROTOCOL – The Baseline

IDNA 200x adopts an inclusion-based approach towards permitting characters in IDNs. A code-point is assumed to be invalid for IDNs unless it is permitted as a result of a Unicode property-based rule or included individually through an exception. When Unicode encodes a character, it associates different properties with it. For example Unicode defines some General_Category (gc) properties such as Lowercase_Letter (Ll), Uppercase_Letter (Lu), Titlecase_Letter (Lt), Nonspacing_Mark (Mn), Decimal_Number (Nd), Math_Symbol (Sm), Unassigned (Cn), etc. [13, 20]. IDNA uses these properties to algorithmically calculate a derived property for each character, which can then be used to determine the validity of that character in IDNs [14].

2.1.1 CHARACTER-LEVEL SPECIFICATION

IDNAbis categorizes all Unicode code-points in four categories: characters which are (i) VALID in the Protocol (PVALID), (ii) licensed in a particular CONTEXT (CONTEXT O/J), (iii) not allowed (DISALLOWED), and (iv) not assigned at this time (UNASSIGNED) [18]. This categorization is derived from the character properties [13, 20] enumerated by the Unicode standard, as already discussed.

[12] The data presented in these examples have been contributed by the country teams of PAN Localization project (see www.PANL10n.net for further details).

2.1.1.1 PVALID Characters

Protocol-Valid characters, commonly abbreviated as PVALID, are allowed in IDNs by the protocol. These contain a variety of character types, including letters, digits, combining marks, symbols, conjunct characters, etc. All PVALID characters will pass through the protocol filter.

2.1.1.2 CONTEXT O/J Characters

IDNA200x also allows some invisible joining characters, in addition to the visible characters for proper rendering of various scripts. These include Zero Width Joiner (ZWJ, U+200C) and Zero Width Non-Joiner (ZWNJ, U+200D) characters. Because they are invisible, they are allowed only in contexts where they are indirectly visible through the characters in the context (where they change the shape of the adjacent characters). In other contexts they are not allowed. Thus, these joiner characters are CONTEXTJ. Other characters which require a context but are visisble would be CONTEXO.

CONTEXTO/J characters need to have an accompanying contextual rule. When such characters are encountered in a string, the corresponding rule is used to determine if they will be allowed in the context. If no such rule is defined, they are DISALLOWED.

The purpose is to restrict the use of CONTEXTO/J characters to certain scripts only. The actual rules and their descriptions can be found in IDNA200x documents [14].

2.1.1.3 DISALLOWED Characters

Characters that are not permitted to be used in IDNs during registration or look-up are categorized as DISALLOWED. If any of these characters are encountered in an IDN label, the label is rejected. This set usually contains characters that are compatibility equivalent of another character, uppercase forms that can be case-folded to other characters, symbols and punctuation marks, etc.

2.1.1.4 UNASSIGNED

Finally, UNASSIGNED value indicates that the code-point is reserved in Unicode and has not been designated. These are not permitted to be used in IDNs. As the UNASSIGNED codes get assigned in the future versions of Unicode, they will transfer to the other three categories.

2.1.1.5 Exceptions

As the character properties were not defined for the purpose of IDNs by Unicode, some characters end up with wrong status when it is derived through the algorithm using these properties. This disparity is corrected through maintaining a list of exceptions [25].

General categories of exceptions with some examples[13] from the IDNA200x are given below:

PVALID -- Would otherwise have been DISALLOWED
00DF; PVALID # LATIN SMALL LETTER SHARP S
03C2; PVALID # GREEK SMALL LETTER FINAL SIGMA
06FE; PVALID # ARABIC SIGN SINDHI POSTPOSITION MEN
0F0B; PVALID # TIBETAN MARK INTERSYLLABIC TSHEG

CONTEXTO -- Would otherwise have been DISALLOWED
00B7; CONTEXTO # MIDDLE DOT
0375; CONTEXTO # GREEK LOWER NUMERAL SIGN (KERAIA)
05F3; CONTEXTO # HEBREW PUNCTUATION GERESH
30FB; CONTEXTO # KATAKANA MIDDLE DOT

CONTEXTO -- Would otherwise have been PVALID
0660; CONTEXTO # ARABIC-INDIC DIGIT ZERO
0661; CONTEXTO # ARABIC-INDIC DIGIT ONE
06F0; CONTEXTO # EXTENDED ARABIC-INDIC DIGIT ZERO
06F1; CONTEXTO # EXTENDED ARABIC-INDIC DIGIT ONE

DISALLOWED -- Would otherwise have been PVALID
0640; DISALLOWED # ARABIC TATWEEL
07FA; DISALLOWED # NKO LAJANYALAN
302E; DISALLOWED # HANGUL SINGLE DOT TONE MARK
3031; DISALLOWED # VERTICAL KANA REPEAT MARK
303B; DISALLOWED # VERTICAL IDEOGRAPHIC ITERATION MARK

2.1.1.6 Representation Schemes

As has been discussed, DNS only allows for domain names in LDH. However, the IDN labels being discussed are in Unicode. These Unicode labels (called U-labels) are converted to ASCII Compatible Encoding (ACE) before they can be transmitted over the Internet. The converted label is in LDH format (called A-label). IDNA recognizes both forms.

A U-label is a valid IDNA string that contains one or more Unicode characters and is in normalization form NFC [16]. It must contain characters from the PVALID set, or additionally characters which are CONTEXTO/J in the context the associated rule is applicable.

An A-Label is an ASCII Compatible Encoding (ACE) of a valid U-label. This conversion is done using a bootsrting algorithm, which converts the Unicode string

[13] The examples are taken from http://tools.ietf.org/html/draft-ietf-idnabis-tables-07. See the document for a complete list.

to a LDH based *punycode* string [11]. However, to distinguish between a normal LDH label and an A-label, a prefix "xn--" is appended before the punycode. As the total string value is limited to 63 characters in the DNS, and four characters are taken up by the prefix, valid punycode string has to be up to 59 characters.

Punycode is a Bootstring algorithm, with IDNA specific parameters. Bootstring is chosen for its efficient encoding and short code length. Punycode takes as input a sequence of one or more Unicode code-points and returns the corresponding ASCII sequence. It uses two functions for this purpose: (i) ToASCII and (ii) ToUnicode. ToASCII converts non-LDH Unicode values, and any LDH codes are returned unaltered. ToUnicode is used to convert back the ASCII sequence to Unicode. Both of them are reversible functions and can be used to convert from Unicode to punycode and vice versa uniquely [9]. For example, the ACE label corresponding to اردوتحقیق is xn--mgbgjgj9ha8b83g.

ACE-labels are finally inserted to the DNS zones during domain name registration and looked-up by client-side applications.

2.1.2 MORPHOLOGICAL CONSTRAINTS AND LICENSING

In addition to character level restrictions, the protocol also places morphological or label level restrictions on the IDN TLD strings. The IDNA protocol restricts some PVALID characters in certain contexts, or allows certain characters which are not PVALID in other contexts. Some specific constraints and licensing scenarios are discussed below.

2.1.2.1 Constraining the Hyphen

Hyphen-Minus (U+002D) is the only ASCII character with Unicode general property value of Dash_Punctuation (Pd) [13, 20], which is present in the LDH set. All other punctuation marks and special symbols are disallowed in ASCII domain names. However, the DNS host name specification [19] does not allow hyphen to be used at the beginning and end of a label. Also, IDNA uses hyphen-minus in a special ACE prefix (xn--) used to denote A-labels. This class of labels, also referred to as XN-Labels in the protocol [29], belong to the Reserved LDH Labels category (R-LDH). Based on these two reasons, the protocol prohibits the use of hyphen (-) in the third and fourth positions of a label, as well as at the start and end of a label.

2.1.2.2 Restricting Combining Marks and Combining Characters

A combining mark or a combining character is intended to be positioned around a base character and can be spacing or non-spacing, i.e. may or may not take space on baseline [56]. They are used in Latin, Cyrillic, Greek, Arabic and many South Asian and Southeast Asian scripts. IDNA protocol does not allow an IDN label to start with combining marks or combining characters.

2.1.2.3 Licensing Contextual Characters

Any CONTEXTO/J character in an IDN label must be accompanied by some context rule that can be checked against the use of such character. The protocol checks whether the label satisfies this rule. Without the rule, or successful contextual firing

of the rule, any such characters are not licensed. The following examples illustrate this process more clearly.

i. Join Control Examples

Some scripts, which are cursive in nature, may need to use join controls to render some characters properly, e.g Zero Width Joiner (U+200D) (to connect otherwise disconnected characters) or Zero Width Non-Joiner (U+200C) (to disconnect otherwise connected characters) in different situations. In case they are needed in an IDN label, the relevant context needs to be identified so that explicit rules may be extracted to license them. As an example, in Urdu a domain name for "bookHouse" should be written as two words کتاب گھر, however, as space is not allowed within domain names, the domain name will appear as connect nonsense word کتابگھر which will not be readable for native speakers of the language. To ensure that the words are separated (in this cursive script) without using space, ZWNJ has to be inserted between them. Similarly, ZWJ is used in some Indic scripts to allow proper formation of consonantal conjuncts [21, 24].

The IDNA protocol defines contextual rules for both ZWJ and ZWNJ. These can be summarized as follows [25].

ZWNJ

1. A ZWNJ can be used between two characters belonging to the same cursive script (e.g. Arabic), so that it breaks a cursive connection between the two otherwise joining characters.

2. A ZWNJ can be used between two characters if the preceding character (before the ZWNJ) is a Virama (for Indic scripts).

ZWJ

1. A ZWJ is only used between two characters if the preceding character (before ZWJ) is a Virama (for Indic scripts).

ii. Arabic Digit Sets

Unicode encodes two sets of Arabic digits and both can be used in Arabic script IDNs. These are called the Arabic-Indic (U+0660..U+0669) and Extended Arabic-Indic digits (U+06F0..U+06F9). The Arabic Indic and Extended Arabic Indic digits are categorized as CONTEXTO characters in IDNAbis, where both cannot be mixed with each other. Thus, if an Arabic-Indic digit is encountered in the IDN label, it is checked for the presence of an Extended Arabic-Indic digit and vice-versa. If any such digit is found then the label is not allowed to be registered. For example, اردو۱۲۳ (containing U+06F1 U+06F2 U+0663) cannot be registered as an IDN. For a more detailed discussion, also see Section 2.2.2.2.

It should be noted here that for CONTEXTJ characters, the contextual rule is checked both at the time of registration and look-up. For CONTEXTO characters, a rule is only checked during registration [18].

2.1.3 SEMANTIC COLLAPSING

Semantically motivated processing of certain characters in the label is also sometimes needed. At the protocol level this is to collapse different mechanisms to encode the same character; a character may be encoded directly and may also be formed by its composing parts. Two such examples are given in Table 2.1 below.

Table 2.1. Equivalent Characters with Different Unicode Encoding

Script	Composed Form	Decomposed Form
Tibetan	ཀྵ (0F69)	ཀ (0F40) + ྵ (0FB5)
Arabic	آ (0622)	ا (0627) + ◌ٓ (0653)

2.1.3.1 Normalization

Such letters can be written in decomposed (base letter followed by a non-spacing/combining mark) or pre-composed form, which are visually equivalent and would represent the same IDN to the user. However, a composite character and its decomposed form yield a different punycode. For example, آ (U+0627 + U+0653) = xn--mgb2g[14] but آ (U+0622) = xn--hgb. Both look exactly the same but their A-label equivalents are different. This is a potential source of user confusion. Unicode provides a Normalization process to address this redundancy [16]. The IDNA protocol requires that an IDN label must be in Unicode Normalization Form C (NFC). Normalization tables for some languages, derived from Unicode are given below. It should be noted that further normalization is stipulated for script blocks by Unicode, however, the following tables only present a subset of these cases as required by the language communities.

Table 2.2. Normalization by Unicode for Bengali Language and Script

Character	Unicode	Decomposition		Description
ো	09CB	09C7	09BE	BENAGALI VOWEL SIGN O
ৌ	09CC	09C7	09D7	BENAGLI VOWEL SIGN AU

[14] Converted using Punycode converter at http://www.charset.org/punycode.php.

Table 2.3. Normalization by Unicode for Mongolian Language in Cyrillic Script

Character	Unicode	Decomposition		Description
й	0439	0438	0306	CYRILLIC SMALL LETTER SHORT I
ё	0451	0435	0308	CYRILLIC SMALL LETTER IO

Table 2.4. Normalization by Unicode for Nepali Language in Devanagari Script

Character	Unicode	Decomposition		Description
ऩ	0929	0928	093C	DEVANAGARI LETTER NNNA

Table 2.5. Normalization by Unicode for Pashto and Urdu Languages in Arabic Script

Character	Unicode	Decomposition		Description
آ	0622	0627	0653	ARABIC LETTER ALEF WITH MADDA ABOVE
أ	0623	0627	0654	ARABIC LETTER ALEF WITH HAMZA ABOVE
ؤ	0624	0648	0654	ARABIC LETTER WAW WITH HAMZA ABOVE
إ	0625	0627	0655	ARABIC LETTER ALEF WITH HAMZA BELOW
ئ	0626	064A	0654	ARABIC LETTER YEH WITH HAMZA ABOVE
ۀ	06C0	06D5	0654	ARABIC LETTER HEH WITH HAMZA ABOVE
ۂ	06C2	06C1	0654	ARABIC LETTER HEH GOAL WITH HAMZA ABOVE
ے‍ٔ	06D3	06D2	0654	ARABIC LETTER YEH BARREE WITH HAMZA ABOVE

Table 2.6. Normalization by Unicode for Sinhala Language and Script

Character	Unicode	Decomposition		Description
ෙ‍ෟ	0DDA	0DD9	0DCA	SINHALA VOWEL SIGN DIGA KOMBUVA
ෙ‍ෝ	0DDC	0DD9	0DCF	SINHALA VOWEL SIGN KOMBUVA HAA AELA-PILLA
ෙ‍ෞ	0DDD	0DDC	0DCA	SINHALA VOWEL SIGN KOMBUVA HAA DIGA AELA-PILLA
ෙ‍ෲ	0DDE	0DD9	0DDF	SINHALA VOWEL SIGN KOMBUVA HA GAYANUKITTA

2.1.4 PRAGMATIC CONSIDERATIONS FOR BACKWARD COMPATIBILITY

For practical reasons, the IDNA protocol maintains a backwards compatible list of code points. If the derived property (i.e. PVALID, DIALLOWED, UNASSIGNED) of any existing code-point changes in the future due to a change in the way it is represented and interpreted by Unicode, or due to a change in the Unicode properties, the current list will be referred [25].

2.2 USER REQUIREMENTS – Cutting the Protocol to Size

The IDNA protocol provides character level categorization of all Unicode characters based on their Unicode properties. However, this yields a large numbers of PVALID characters in a script block all of which may not be relevant for IDNs by a script or a language. Community feedback can help reduce this set for IDN labels. In addition, this feedback can also be used to identify other user needs which are inhibited by the protocol and thus need to be addressed through the application layer, outside the scope of the IDNA200x process. The following sections discuss such issues.

2.2.1 CHARACTER-LEVEL SPECIFICATION

As has been discussed earlier, Unicode encodes a variety of character types, including consonants, vowels, combining marks, digits, punctuation marks and symbols. Many of these symbols are not relevant for IDNs and are DISALLOWED by the protocol. However, many more such symbols are PVALID but are still redundant as they are not needed for a particular language or for any of the languages using the particular script. Thus, additional constraints need to be added on top of the protocol to contain such cases. This section looks into such cases, with examples from various script and language communities.

2.2.1.1 SVALID Characters

In many cases, there are characters in a script which are PVALID but are not needed for any language using that script. Such characters may be blocked at registry level using a language table. However, instead of repeating the process for each language using the script, a separate script level analysis could be performed generating a script level table, which then gets applicable to all languages using it.

The characters which are allowed through this script level analysis can be referred to as SVALID characters, and can be listed in a script table. This would always be a sub-set of PVALID characters for the script. To make the table, the whole script community would need to come together and agree to restrict these characters. The restriction can be self-binding, with each registry volunteering to adopt the table, or binding if the participating registries enforce the decision through a collaborative agreement.

The Arabic Script IDNs Working Group (ASIWG) is a self-organizing group involving members from language communities that use Arabic script[15], including Arabic, Persian, Jawi, Urdu, Sindhi and other languages. In 2008, this working group agreed to disallow additional characters on the script level which are otherwise classified

[15] See http://www.arabic-script-domains.org/wiki/Main_Page for details.

as PVALID by the protocol. This is based on two observations. First because these are not used by any of the written languages, and second that their use in IDNs might cause confusion for users. These characters include the Quranic marks and stylistic/formatting characters: 0616..061A, 06D6..06DC, 06DF..06E8, 06EA..06ED.

Similar efforts have been done through JET for CJK languages [30]. However there are many other scripts which are used by multiple languages and thus need to take similar measures. Cyrillic, Devanagari and Tibetan are some examples of such scripts.

2.2.1.2 LVALID Characters

To serve a particular linguistic community, and not all the languages using a script, an even smaller set of characters could be required. For example, Arabic language requires a sub-set of SVALID characters from Arabic script block. Persian also requires a smaller set, but different from that required for Arabic language. A more interesting case is for Pakistan, which requires support for all the sixty-plus languages spoken in the country, in a single table as decided by the community. However, this set is also a sub-set of the SVALID table. Thus, there is need to have language(s) specific tables which list the Language Valid (LVALID) characters. LVALID character set should be a sub-set of SVALID set. For example Arabic letter ٻ (U+067B) is an SVALID character LVALID for Sindhi but not LVALID for Arabic, Persian or Urdu. Similarly, Cyrillic letter ѱ (U+0471) is a PVALID and SVALID character but not LVALID for Mongolian language.

As examples, this section provides the list of PVALID characters which are not LVALID for eight language communities. The languages include Bengali, Dzongkha, Lao, Khmer, Mongolian, Nepali, Sinhala and Urdu. This can be controlled by putting an additional filter through language table at the registry level allowing only LVALID characters.

A more critical situation arises when IDNAbis categorizes some characters as DISALLOWED in the protocol whereas they are essential to represent a language and required by the language community. This has to be rectified at this time, by adding these characters to the exception list of the protocol. Many such characters have been identified and have been fed into the IDNAbis exception list (see Section 2.1.1.5 above).

Bengali language has many signs which are not needed by the community, as shown in Table 2.7. In addition there are a few characters which are DISALLOWED, but are needed by the language community. However, these can be composed from a sequence of PVALID characters, which can then be used in Bengali IDNs.

Table 2.7. Bengali Characters and Decision by IDNA and Language Community

Unicode Range	Description	IDNAbis Decision	Language Community Decision
0981..0982	# BENGALI SIGN CHANDRABINDU.. BENGALI SIGN ANUSAVRA	PVALID	NO
09BC..09BD	# BENGALI SIGN NUKTA.. BENGALI SIGN AVAGRAHA	PVALID	NO
09C4	# BENGALI VOWEL SIGN VOCALIC RR	PVALID	NO
09E0..09E3	# BENGALI LETTER VOCALIC RR.. BENGALI SIGN VOCALIC LL	PVALID	NO
09F0..09F1	# BENGALI LETTER RA WITH MIDDLE DIAGONAL.. BENGALI LETTER RA WITH LOWER DIAGONAL	PVALID	NO
09DC..09DF	# BENGALI LETTER RRA.. BENGALI LETTER YYA	DISALLOWED	YES

There are also a large number of characters that are PVALID but not likely to appear in Dzongkha IDN labels, as shown in Table 2.8. These can be DISALLOWED at the registry level. Moreover, there are characters that are needed by the Dzongkha language community but are DISALLOWED by the protocol. However, these can be composed through other PVALID characters.

There are two characters in Tibetan script which are used to mark the syllabic boundary in Dzongkha and required to correctly write the words. TIBETAN MARK INTERSYLLABIC TSHEG (U+0F0B) is required in domain names and TIBETAN MARK DELIMITER (U+0F0C) is recommended to be normalized to U+0F0B to avoid ambiguity in domain names. Previously, these two characters were DISALLOWED. However, through this feedback from the community, in the latest draft of tables TIBETAN MARK INTERSYLLABIC TSHEG (U+0F0B) is updated to PVALID through the exception list, whereas TIBETAN MARK DELIMITER (U+0F0C) is still DISALLOWED and should be normalized in the application layer.

Table 2.8. Dzongkha Characters and Decision by IDNA and Language Community

Unicode Range	Description	IDNAbis Decision	Language Community Decision
0F00	# TIBETAN SYALLABLE OM	PVALID	NO
0F18..0F19	# TIBETAN ASTROLOGICAL SIGN	PVALID	NO

	KHYUD PA.. TIBETAN ASTROLOGICAL SIGN SDONG TSHUGS		
0F35	# TIBETAN MARK NGAS BZUNG NYI ZLA	PVALID	NO
0F37	# TIBETAN MARK NGAS BZUNG SGOR RTAGS	PVALID	NO
0F3E..0F3F	# TIBETAN SIGN YAR TSHES.. TIBETAN SIGN MAR TSHES	PVALID	NO
0F7E..0F7F	# TIBETAN SIGN RJES SU NGA RO.. TIBETAN SIGN RNAM BCAD	PVALID	NO
0F82..0F83	# TIBETAN SIGN NYI ZLA NAA DA.. TIBETAN SIGN SNA LDAN	PVALID	NO
0F86..0F8B	# TIBETAN SIGN LCI RTAGS.. TIBETAN SIGN GRU MD RGYINGS	PVALID	NO
0F96	# TIBETAN SUBJOINED LETTER TTA	PVALID	NO
0F9A..0F9C	# TIBETAN SUBJOINED LETTER TTHA.. TIBETAN SUBJOINED LETTER DDA	PVALID	NO
0FC6	# TIBETAN SYMBOL PADMA GDAN	PVALID	NO
0F43	# TIBETAN LETTER GHA	DISALLOWED	YES
0F4D	# TIBETAN LETTER DDHA	DISALLOWED	YES
0F52	# TIBETAN LETTER DHA	DISALLOWED	YES
0F57	# TIBETAN LETTER BHA	DISALLOWED	YES
0F5C	# TIBETAN LETTER DZHA	DISALLOWED	YES
0F69	# TIBETAN LETTER KSSA	DISALLOWED	YES
0F73	# TIBETAN VOWEL SIGN II	DISALLOWED	YES
0F75	# TIBETAN VOWEL SIGN UU	DISALLOWED	YES
0F93	# TIBETAN SUBJOINED LETTER GHA	DISALLOWED	YES
0FA2	# TIBETAN SUBJOINED LETTER DHA	DISALLOWED	YES
0FA7	# TIBETAN SUBJOINED LETTER BHA	DISALLOWED	YES
0FAC	# TIBETAN SUBJOINED LETTER DZHA	DISALLOWED	YES
0FB9	# TIBETAN SUBJOINED LETTER KSSA	DISALLOWED	YES

Table 2.9 lists the characters which are not needed in IDN labels by the Khmer language speakers. These can be restricted at the registry/language table level above the protocol level. Two Khmer characters KHMER VOWEL INHERENT AQ (U+17B4) and KHMER VOWEL INHERENT AA (U+17B5) are not wanted by the language community and have been updated to DISALLOWED in the IDNAbis tables. There is no character that is DISALLOWED by IDNA and required by the language speakers.

Table 2.9. Khmer Characters and Decision by IDNA and Language Community

Unicode Range	Description	IDNAbis Decision	Language Community Decision
179D..179E	# KHMER LETTER SHA.. KHMER LETTER SSO	PVALID	NO
17A3..17A4	# KHMER INDEPENDENT VOWEL QAQ.. KHMER INDEPENDENT VOWEL QAA	PVALID	NO
17CE	# KHMER SIGN KAKABAT	PVALID	NO
17D1	# KHMER SIGN VIRIAM	PVALID	NO
17D3	# KHMER SIGN BATHAMASAT	PVALID	NO
17DC..17DD	# KHMER SIGN AVAKRAHASANYA.. KHMER SIGN ATTHACAN	PVALID	NO

The following table contains Lao characters which are PVALID but not considered necessary for domain names by Lao language community. The Lao characters shown in the last two rows of Table 2.10 are needed but DISALLOWED. However they can be generated by composing other PVALID characters.

Table 2.10. Lao Characters and Decision by IDNA and Language Community

Unicode Range	Description	IDNAbis Decision	Language Community Decision
0EAF	# LAO ELLIPSIS	PVALID	NO
0EC6	# LAO KO LA	PVALID	NO
0EB3	# LAO VOWEL SIGN AM	DISALLOWED	YES
0EDC..0EDD	# LAO LETTER HO NO.. LAO LETTER HO MO	DISALLOWED	YES

In Mongolia, Mongolian language is written using Cyrillic script. As Cyrillic script is used to represent a number of other languages, there are many Cyrillic characters that are not used in Mongolian and can be safely restricted from use in Mongolian IDNs. These are recommended to be removed from the character set at the registry level using the Mongolian language table. Table 2.11 gives a complete list of these characters.

Table 2.11. Mongolian Characters and Decision by IDNA and Language Community

Unicode Range	Description	IDNAbis Decision	Language Community Decision
0450	# CYRILLIC SMALL LETTER IE WITH GRAVE	PVALID	NO
0452..045F	# CYRILLIC SMALL LETTER DJE.. CYRILLIC SMALL LETTER DZHE	PVALID	NO
0461	# CYRILLIC SMALL LETTER OMEGA	PVALID	NO
0463	# CYRILLIC SMALL LETTER YAT	PVALID	NO
0465	# CYRILLIC SMALL LETTER IOTIFIED E	PVALID	NO
0467	# CYRILLIC SMALL LETTER LITTLE YUS	PVALID	NO
0469	# CYRILLIC SMALL LETTER IOTIFIED LITTLE YUS	PVALID	NO
046B	# CYRILLIC SMALL LETTER BIG YUS	PVALID	NO
046D	# CYRILLIC SMALL LETTER IOTIFIED BIG YUS	PVALID	NO
046F	# CYRILLIC SMALL LETTER KSI	PVALID	NO
0471	# CYRILLIC SMALL LETTER PSI	PVALID	NO
0473	# CYRILLIC SMALL LETTER FITA	PVALID	NO
0475	# CYRILLIC SMALL LETTER IZHITSA	PVALID	NO
0477	# CYRILLIC SMALL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT	PVALID	NO
0479	# CYRILLIC SMALL LETTER UK	PVALID	NO
047B	# CYRILLIC SMALL LETTER ROUND OMEGA	PVALID	NO
047D	# CYRILLIC SMALL LETTER OMEGA WTH TITLO	PVALID	NO
047F	# CYRILLIC SMALL LETTER OT	PVALID	NO
0481	# CYRILLIC SMALL LETTER KOPPA	PVALID	NO
0483..0486	# COMBINING CYRILLIC TITLO.. COMBINING CYRILLIC PSILI PNEUMATA	PVALID	NO
048B	# CYRILLIC SMALL LETTER SHORT I WITH TAIL	PVALID	NO
048D	# CYRILLIC SMALL LETTER SEMIDOFT SIGN	PVALID	NO
048F	# CYRILLIC SMALL LETTER ER WITH TICK	PVALID	NO
0491	# CYRILLIC SMALL LETTER GHE WITH UPTURN	PVALID	NO
0493	# CYRILLIC SMALL LETTER GHE WITH STROKE	PVALID	NO
0495	# CYRILLIC SMALL LETTER GHE WITH MIDDLE HOOK	PVALID	NO
0497	# CYRILLIC SMALL LETTER ZHE WITH DESCENDER	PVALID	NO
0499	# CYRILLIC SMALL LETTER ZE WITH	PVALID	NO

	DESCENDER		
049B	# CYRILLIC SMALL LETTER KA WITH DESCENDER	PVALID	NO
049D	# CYRILLIC SMALL LETTER KA WITH VERTICAL STROKE	PVALID	NO
049F	# CYRILLIC SMALL LETTER KA WITH STROKE	PVALID	NO
04A1	# CYRILLIC SMALL LETTER BASHKIR KA	PVALID	NO
04A3	# CYRILLIC SMALL LETTER EN WITH DESCENDER	PVALID	NO
04A5	# CYRILLIC SMALL LIGATURE EN GHE	PVALID	NO
04A7	# CYRILLIC SMALL LETTER PE WITH MIDDLE HOOK	PVALID	NO
04A9	# CYRILLIC SMALL LETTER ABKHASIAN HA	PVALID	NO
04AB	# CYRILLIC SMALL LETTER ES WITH DESCENDER	PVALID	NO
04AD	# CYRILLIC SMALL LETTER TE WITH DESCENDER	PVALID	NO
04AF	# CYRILLIC SMALL LETTER STRAIGHT U	PVALID	NO
04B1	# CYRILLIC SMALL LETTER STRAIGHT U WITH STROKE	PVALID	NO
04B3	# CYRILLIC SMALL LETTER HA WITH DESCENDER	PVALID	NO
04B5	# CYRILLIC SMALL LIGATURE TE TSE	PVALID	NO
04B7	# CYRILLIC SMALL LETTER CHE WITH DESCENDER	PVALID	NO
04B9	# CYRILLIC SMALL LETTER CHE WITH VERTICAL STROKE	PVALID	NO
04BB	# CYRILLIC SMALL LETTER SHHA	PVALID	NO
04BD	# CYRILLIC SMALL LETTER ABKHASIAN CHE	PVALID	NO
04BF	# CYRILLIC SMALL LETTER ABKHASIAN CHE WITH DESCENDER	PVALID	NO
04C2	# CYRILLIC SMALL LETTER ZHE WITH BREVE	PVALID	NO
04C4	# CYRILLIC SMALL LETTER KA WITH HOOK	PVALID	NO
04C6	# CYRILLIC SMALL LETTER EL WITH TAIL	PVALID	NO
04C8	# CYRILLIC SMALL LETTER EN WITH HOOK	PVALID	NO
04CA	# CYRILLIC SMALL LETTER EN WITH TAIL	PVALID	NO
04CC	# CYRILLIC SMALL LETTER KHAKASSIAN CHE	PVALID	NO
04CE	# CYRILLIC SMALL LETTER EM WITH TAIL	PVALID	NO
04CF	# CYRILLIC SMALL LETTER PALOCHKA	PVALID	NO
04D1	# CYRILLIC SMALL LETTER A WITH BREVE	PVALID	NO
04D3	# CYRILLIC SMALL LETTER A WITH DIAERESIS	PVALID	NO
04D5	# CYRILLIC SMALL LIGATURE A IE	PVALID	NO
04D7	# CYRILLIC SMALL LETTER IE WITH BREVE	PVALID	NO

04D9	# CYRILLIC SMALL LETTER SCHWA	PVALID	NO
04DB	# CYRILLIC SMALL LETTER SCHWA WITH DIAERESIS	PVALID	NO
04DD	# CYRILLIC SMALL LETTER ZHE WITH DIAERESIS	PVALID	NO
04DF	# CYRILLIC SMALL LETTER ZE WITH DIAERESIS	PVALID	NO
04E1	# CYRILLIC SMALL LETTER ABKHASIAN DZE	PVALID	NO
04E3	# CYRILLIC SMALL LETTER I WITH MACRON	PVALID	NO
04E5	# CYRILLIC SMALL LETTER I WITH DIAERESIS	PVALID	NO
04E7	# CYRILLIC SMALL LETTER O WITH DIAERESIS	PVALID	NO
04EB	# CYRILLIC SMALL LETTER BARRED O WITH DIAERESIS	PVALID	NO
04ED	# CYRILLIC SMALL LETTER E WITH DIAERESIS	PVALID	NO
04EF	# CYRILLIC SMALL LETTER U WITH MACRON	PVALID	NO
04F1	# CYRILLIC SMALL LETTER U WITH DIAERESIS	PVALID	NO
04F3	# CYRILLIC SMALL LETTER U WITH DOUBLE ACCUTE	PVALID	NO
04F5	# CYRILLIC SMALL LETTER CHE WITH DIAERESIS	PVALID	NO
04F7	# CYRILLIC SMALL LETTER GHE WITH DESCENDER	PVALID	NO
04F9	#CYRILLIC CAPITAL LETTER YERU WITH DIAERESIS	PVALID	NO
04FB	# CYRILLIC SMALL LETTER GHE WITH STROKE AND HOOK	PVALID	NO
04FD	# CYRILLIC SMALL LETTER HA WITH HOOK	PVALID	NO
04FF	# CYRILLIC SMALL LETTER HA WITH STROKE	PVALID	NO

Nepali language uses a subset of Devanagari characters. Table 2.12 gives details of characters that are not used by Nepali language and thus can be filtered using the language table.

Table 2.12. Nepali Characters and Decision by IDNA and Language Community

Unicode Range	**Description**	**IDNAbis Decision**	**Language Community Decision**
0904	# DEVANAGARI LETTER SHORT A	PVALID	NO
0911..0912	# DEVANAGARI LETTER CANDRA O.. DEVANAGARI LETTER SHORT O	PVALID	NO

0931	# DEVANAGARI LETTER RRA	PVALID	NO
0934	# DEVANAGARI LETTER LLLA	PVALID	NO
0944..0946	# DEVANAGARI VOWEL SIGN VOCALIC RR.. DEVANAGARI VOWEL SIGN SHORT E	PVALID	NO
0949	# DEVANAGARI VOWEL SIGN CANDRA O	PVALID	NO
0951..0954	# DEVANAGARI STRESS SIGN UDATTA.. DEVANAGARI ACUTE ACCENT	PVALID	NO
0960..0963	# DEVANAGARI LETTER VOCALIC RR.. DEVANAGARI VOWEL SIGN VOCALIC LL	PVALID	NO
090C..090E	# DEVANAGARI LETTER VOCALIC L.. DEVANAGARI LETTER SHORT E	PVALID	NO
093C..093D	# DEVANAGARI SIG NUKTA.. DEVANAGARI SIGN AVAGRAHA	PVALID	NO
094A	# DEVANAGARI VOWEL SIGN SHORT O	PVALID	NO
097B..097F	# DEVANAGARI LETTER GGA.. DEVANAGARI LETTER BBA	PVALID	NO

Only one character in Sinhala language in Sri Lanka, which uses its own Sinhala script, is not supported by the language community and that is also DISALLOWED by the IDNAbis tables. All remaining characters are required by the language speakers and are declared as PVALID.

Urdu language uses Arabic script. Feedback from the language community for Urdu in Pakistan suggests the following changes which should be managed through the language table.

Table 2.13. Urdu Characters and Decision by IDNA and Language Community

Unicode Range	**Description**	**IDNAbis Decision**	**Language Community Decision**
0614..0615	# ARABIC SIGN TAKHALLUS.. ARABIC SMALL HIGH TAH	PVALID	NO
0657	# ARABIC INVERTED DAMMA	PVALID	NO
0659	# ARABIC ZWARAKAY	PVALID	NO
065A..065E	# ARABIC VOWEL SMALL V ABOVE.. ARABIC FATHA WITH TWO DOTS	PVALID	NO
066E..066F	# ARABIC LETTER DOTLESS BEH.. ARABIC LETTER DOTLESS QAF	PVALID	NO
0671..0674	# ARABIC LETTER ALEF WASLA.. ARABIC LETTER HIGH HAMZA	PVALID	NO
067A..067D	# ARABIC LETTER TTEHEH.. ARABIC LETTER TEH WITH THREE DOTS ABOVE	PVALID	NO

	DOWN		
067F..0685	# ARABIC LETTER TEHEH.. ARABIC LETTER HAH WITH THREE DOTS ABOVE	PVALID	NO
0687	# ARABIC LETTER TCHEHEH	PVALID	NO
0689..0690	# ARABIC LETTER DAL WITH RING.. ARABIC LETTER DAL WITH FOUR DOTS ABOVE	PVALID	NO
0692..0697	# ARABIC LETTER REH WITH SMALL V.. ARABIC LETTER REH WITH TWO DOTS ABOVE	PVALID	NO
0699..06A8	# ARABIC LETTER REH WITH FOUR DOTS ABOVE.. ARABIC LETTER QAF WITH THREE DOTS ABOVE	PVALID	NO
06AA..06AE	# ARABIC LETTER SWASH KAF.. ARABIC LETTER KAF WITH THREE DOTS BELOW	PVALID	NO
06B0..06B9	# ARABIC LETTER GAF WITH RING.. ARABIC LETTER NOON WITH DOT BELOW	PVALID	NO
06BB..06BD	# ARABIC LETTER RNOON.. ARABIC LETTER NOON WITH THREE DOTS ABOVE	PVALID	NO
06BF	# ARABIC LETTER TCHEH WITH DOT ABOVE	PVALID	NO
06C4..06CB	# ARABIC LETTER WAW WITH RING.. ARABIC LETTER VE	PVALID	NO
06CD..06D1	# ARABIC LETTER YEH WITH TAIL.. ARABIC LETTER YEH WITH THREE DOTS BELOW	PVALID	NO
06D5..06DC	# ARABIC LETTER AE.. ARABIC SMALL HIGH SEEN	PVALID	NO
06DF..06E8	# ARABIC SMALL HIGH ROUNDED ZERO.. ARABIC SMALL HIGH NOON	PVALID	NO
06EA..06EF	# ARABIC EMPTY CENTRE LOW STOP.. ARABIC LETTER REH WITH INVERTED V	PVALID	NO
06FA..06FC	# ARABIC LETTER SHEEN WITH DOT BELOW.. ARABIC LETTER GHAIN WITH DOT BELOW	PVALID	NO
06FF	# ARABIC LETTER HEH WITH INVERTED V	PVALID	NO

For Urdu, some of these characters are not SVALID for Arabic script and thus can be filtered through script level tables.

Similar efforts need to be done for other language communities, which aim for deploying IDN support for their community.

2.2.2 MORPHOLOGICAL CONSTRAINTS

As at the protocol level, there are certain restrictions which the user community may also require at a morphological or label level, and not just the character level. These constraints cannot be handled by script or language tables and must be handled by scripting rules as validation checks during registration process. Again, such rules should be based on the needs of the language communities.

2.2.2.1 Script Mixing

Some languages require more than one script for complete representation. For example the Japanese language uses an ideographic writing system where each word is represented by a symbol. Japanese originally used Chinese ideographs, but later developed two syllabaries namely Hiragana and Katakana. These are simplified or stylized versions of certain ideographs. Chinese ideographs (called Kanji) are still used in combination with Hiragana and Katakana in modern Japanese [27]. In this case an IDN TLD string will consist of characters drawn from more than one script, thereby producing a mixed script label. A similar case is presented by the Sinhalese community. Sinhala and Tamil are both national languages of Sri Lanka, therefore, the language community may require a Tamil-Sinhala mixed script IDN label [22].

On the contrary, there are other languages that do not require mixing of script in an IDN label. For example, the Khmer language community only requires Khmer script in IDNs.

The entities responsible for registration of IDNs need to perform script-mixing validation checks on IDNs. The following possible cases can be implemented, through these checks.

1. Only single script characters are allowed (ووو-اردو-ادارہ)
2. Single script characters are allowed within a label, but different levels of labels can be in either this script or in LDH (www.اردو.org)
3. Free mixing of script characters with LDH is allowed within labels (www.ابجدabc.com٢)
4. Single script characters are allowed within a label, but different levels of labels can be in different scripts (not limited to Latin + one script) (www.दो.ادارہ)
5. Free mixing of script characters with any other script characters is allowed within labels. No restriction is applied (www.ابجد -abc-दो.newTLD)

The IDNA200x protocol allows for all these possibilities [18]. However, mixing of scripts should generally not be allowed. Normally the first case is applicable, though the second case may also be applicable if the existing gTLDs and ccTLDs start supporting IDN labels within their domain space. The third case is normally not needed within a language, though there may be specific needs especially for digits (see next section). The fourth and fifth cases above are quite rare, though there are authentic cases, e.g. as discussed for Japanese earlier.

Allowing free mixing of scripts can also cause some usability issues. In such cases it becomes more important to restrict mixing. This is especially true for scripts containing similar looking or confusable characters. For instance Cyrillic letter ä (U+04D3) and Latin letter ä (U+00E4) are visually identical, but produce different A-labels. Thus, mixing of Latin and Cyrillic characters might lead to the possibility of having two exactly similar IDN pointing to two different locations.

ICANN does not encourage mixing of scripts in an IDN TLD unless there is legitimate need. *"Exceptions to this guideline are permissible for languages with established orthographies and conventions that require the commingled use of multiple scripts"* [40].

2.2.2.2 Digit Mixing

DNS permits the use of numerals or digits in a domain name in the LDH schemes. However, most scripts also have their own digit sets. This creates the following possible cases for digit mixing.

1. Only local script digits are used
2. Only Latin digits are used
3. Local script digits and Latin digits are allowed at different levels
4. Local script and Latin digits are allowed to be mixed within a label
5. Digits from other scripts are allowed across levels in addition to this script and ASCII digits
6. Digits from other scripts are allowed to be mixed within labels at the same level

Language communities could decide between these options. Many languages would prefer the first option. However, some scripts do not have digits or use Latin digits even though local digits are available and thus may choose the second option. Where scripts are mixed across levels, digits may also be mixed across levels. However, the fourth option would be very rare. In this case as well, as in the case of script mixing, there are some legitimate needs. For example, Jawi language (spoken in Malaysia and written with Arabic script) reduplicates[16] to create words. The reduplicated word *rama-rama* means a butterfly. However, in Jawi instead of writing it as راماراما it is written as راما۲ (i.e. instead of rama-rama, written as rama2). This is the only context and the only Arabic digit used in Jawi. Otherwise, Jawi uses Latin digits for numbers. Thus, if one had to write "butterfly10" in Jawi, one would need to write 10راما۲ which would require mixing digits within a label[17]. Fifth and sixth options are possible but there are no known examples where such possibilities are needed.

[16] Reduplication is a linguistic phenomenon, in which part or whole word in a language is repeated to create new words, e.g. see http://en.wikipedia.org/wiki/Reduplication for examples from different languages.

[17] Example shared by MYNIC staff during ASIWG meetings.

Whatever the language community decides will have to be implemented through registry level constraints.

Arabic script generally presents another interesting example in this regard. Unicode encodes two sets of digits sets in the Arabic script block, as shown in Table 2.14. The two sets are only different in shape for the digits 4, 5, 6 and 7. Other digits have same shape. In this context, an additional constraint needs to be put to ensure that the two digits within the script table are also not mixed. For Arabic, this has been handled by putting appropriate rule in the Bidi document of the protocol [26].

Table 2.14. ASCII, Arabic-Indic and Extended Arabic-Indic Digits

ASCII		ARABIC-INDIC		EXTENDED ARABIC-INDIC	
0	0300	٠	0660	۰	06F0
1	0301	١	0661	۱	06F1
2	0302	٢	0662	۲	06F2
3	0303	٣	0663	۳	06F3
4	0304	٤	0664	۴	06F4
5	0305	٥	0665	۵	06F5
6	0306	٦	0666	۶	06F6
7	0307	٧	0667	۷	06F7
8	0308	٨	0668	۸	06F8
9	0309	٩	0669	۹	06F9

2.2.3 SEMANTIC DISAMBIGUATION

Section 2.1.3 discusses semantic disambiguation as achieved by the IDNA protocol through Unicode NFC. Beyond this normalization, IDNA does not include any mapping or case-folding steps as part of the standard. If required, these have to be performed at the registry or application level. This section describes steps that should be taken to distinguish similar looking IDN labels at the language/script level.

2.2.3.1 Extended Normalization

Unicode does not define normalization for all characters. This is especially true of composite forms that can be created using a base character followed by a combining mark. Equivalence of such characters has to be defined at the script level. If any such character is encountered in an IDN TLD, the corresponding additional script level normalization has to be performed.

As an example, ۆ (U+06C6) can be formed by composing و (U+0648) with ◌ٚ (U+065A). However, even though both sequences give exactly the same character visually, they are not defined as equivalent in Unicode and would not be normalized through the protocol defined process. Same is the case with Khmer Vowel Sign OO

ါ (U+17C4) and Khmer Vowel Sign OE ေ (U+17BE). Both have equivalent decomposed forms, but Unicode does not provide a mapping between the decomposed and composed forms.

Examples from some of these languages and scripts are given in the tables below.

Table 2.15. Khmer Script Characters Requiring Extended Normalization

Character	Unicode	Decomposed Form		Description
ါ	17C4	17C1	17B6	KHMER VOWEL SIGN OO
ေ	17BE	17C1	17B8	KHMER VOWEL SIGN OE

Table 2.16. Arabic Script Characters Requiring Extended Normalization

Character	Unicode	Decomposed Form		Description
ئ	0626	0649	0654	ARABIC LETTER YEH WITH HAMZA ABOVE
ئ	0626	06CC	0654	ARABIC LETTER YEH WITH HAMZA ABOVE
ۀ	06C0	0647	0654	ARABIC LETTER HEH WITH YEH ABOVE
ۀ	06C2	0647	0654	ARABIC LETTER HEH GOAL WITH HAMZA ABOVE
ځ	0681	062D	0654	ARABIC LETTER HAH WITH HAMZA ABOVE
ڑ	076C	0631	0654	ARABIC LETTER REH WITH HAMZA ABOVE
ئ	0678	0649	0674	ARABIC LETTER HIGH HAMZA YEH
ئ	0678	06CC	0674	ARABIC LETTER HIGH HAMZA YEH
ۇ	06C7	0648	064F	ARABIC LETTER U
ۈ	06C8	0648	0670	ARABIC LETTER YU
ڤ	06CF	0648	06EC	ARABIC LETTER WAW WITH DOT ABOVE
غ	063A	0639	06EC	ARABIC LETTER GHAIN
ض	0636	0635	06EC	ARABIC LETTER DAD

خ	062E	062D	06EC	ARABIC LETTER KHAH
ڿ	06BF	0686	06EC	ARABIC LETTER TCHEH WITH DOT ABOVE
ذ	0630	062F	06EC	ARABIC LETTER THAL
ز	0632	0631	06EC	ARABIC LETTER ZAIN
ڶ	06B6	0644	06EC	ARABIC LETTER LAM WITH DOT ABOVE
ڧ	06A7	066F	06EC	ARABIC LETTER QAF WITH DOT ABOVE
ف	0641	06A1	06EC	ARABIC LETTER FEH
ن	0646	06BA	06EC	ARABIC LETTER NOON
ڪ	06AC	0643	06EC	ARABIC LETTER KAF WITH DOT ABOVE
ڮ	0762	06A9	06EC	ARABIC LETTER KEHEH WITH DOT ABOVE
۾	0765	0645	06EC	ARABIC LETTER MEEM WITH DOT ABOVE
ځ	0772	062D	0615	ARABIC LETTER HAH WITH SMALL ARABIC LETTER TAH ABOVE
ٹ	0679	066E	0615	ARABIC LETTER TTEH
ڑ	0691	0631	0615	ARABIC LETTER RREH
ڈ	0688	062F	0615	ARABIC LETTER DDAL
ڑ	0771	0697	0615	ARABIC LETTER REH WITH SMALL ARABIC LETTER TAH AND TWO DOTS
ں	0768	0646	0615	ARABIC LETTER NOON WITH SMALL TAH
ڈ	068B	068A	0615	ARABIC DAL WITH DOT BELOW AND SMALL TAH
ڻ	06BB	06BA	0615	ARABIC LETTER RNOON
ۍ	063D	06CC	065B	ARABIC LETTER FARSI YEH WITH INVERTED V
ۋ	06C9	0648	065B	ARABIC LETTER KIRGHIZ YU
ش	077E	0633	065B	ARABIC LETTER SEEN WITH INVERTED V

ڈ	06EE	062F	065B	ARABIC LETTER DAL WITH INVERTED V
ڑ	06EF	0631	065B	ARABIC LETTER REH WITH INVERTED V
ۿ	06FF	06BE	065B	ARABIC LETTER HEH WITH INVERTED V
ۿ	06FF	0647	065B	ARABIC LETTER HEH WITH INVERTED V
ۓ	063F	06CC	06DB	ARABIC LETTER FARSI YEH WITH THREE DOTS ABOVE
ۓ	063F	0649	06DB	ARABIC LETTER FARSI YEH WITH THREE DOTS ABOVE
ڜ	0634	0633	06DB	ARABIC LETTER SHEEN
ڛ	069C	069B	06DB	ARABIC LETTER SEEN WITH THREE DOTS BELOW AND THREE DOTS ABOVE
ڻ	062B	066E	06DB	ARABIC LETTER THEH
څ	0685	062D	06DB	ARABIC LETTER HAH WITH THREE DOTS ABOVE
ڑ	0698	0631	06DB	ARABIC LETTER JEH
ډ	068E	062F	06DB	ARABIC LETTER DUL
ڠ	06A0	0639	06DB	ARABIC LETTER AIN WITH THREE DOTS ABOVE
ڤ	06A4	06A1	06DB	ARABIC LETTER VEH
ڧ	06A8	066F	06DB	ARABIC LETTER QAF WITH THREE DOTS ABOVE
ڭ	06AD	0643	06DB	ARABIC LETTER NG
ڴ	06B4	06AF	06DB	ARABIC LETTER GAF WITH THREE DOTS ABOVE
ڵ	06B7	0644	06DB	ARABIC LETTER LAM WITH THREE DOTS ABOVE
ڼ	06BD	06BA	06DB	ARABIC LETTER NOON WITH THREE DOTS ABOVE
ڪ	0763	06A9	06DB	ARABIC LETTER KEHEH WITH THREE DOTS ABOVE
ڔ	0628	066E	065C	ARABIC LETTER BEH

ڈ	068A	062F	065C	ARABIC DAL WITH DOT BELOW
ڈ	068B	0688	065C	ARABIC LETTER DAL WITH DOT BELOW AND SMALL TAH
ڕ	0694	0631	065C	ARABIC LETTER REH WITH DOT BELOW
ڤ	06A3	0641	065C	ARABIC LETTER FEH WITH DOT BELOW
ڼ	06B9	0646	065C	ARABIC LETTER NOON WITH DOT BELOW
ۻ	06FB	0636	065C	ARABIC LETTER DAD WITH DOT BELOW
ٹ	0751	062B	065C	ARABIC LETTER BEH WITH DOT BELOW AND THREE DOTS ABOVE
۾	0766	0645	065C	ARABIC LETTER MEEM WITH DOT BELOW
ۑ	06CE	06CC	065A	ARABIC LETTER YEH WITH SMALL V
ۑ	06CE	0649	065A	ARABIC LETTER YEH WITH SMALL V
ٻ	0756	066E	065A	ARABIC LETTER BEH WITH SMALL V
ڻ	0769	0646	065A	ARABIC LETTER NOON WITH SMALL V

2.2.3.2 Variant Mapping

Apart from pre-composed characters, there are some visually confusable characters within a script or a language. Mapping has to be defined at the appropriate level to maintain equivalence of such characters and reduce user confusion and susceptibility to phishing.

As an example, in Arabic script, for instance, each character can take one of four visual forms depending upon its context in a word. These are (i) Initial, (ii) Medial, (iii) Final and (iv) Isolated forms. Characters that look similar in any of the four forms are said to be visually confusable. For e.g. Arabic script provides two versions for the letter Kaf; an Arabic version ك (U+0643) and a Persian version ک (U+06A9). Both look exactly the same in their initial and medial positions, e.g. كتاب vs. کتاب(*kitab*, 'book').

As discussed in the section of Script Mixing, variant mapping may also be needed across scripts, where a script may have similar looking letters to LDH. Moreover, a letter in a script can also be similar to another script allowed within the label, e.g. Cyrillic letter ӓ (U+04D3) and Latin letter ä (U+00E4) are visually identical. This

similarity sometimes is not restricted to visual similarity. For example, in the case of Chinese and Simplified Chinese, two ideographs may be visually different but may still be confusingly similar for the user.

Finally, mapping may also be required at language level, as two characters may be distinct at script level, but may be confusingly similar at language level. An example is that of Arabic Kaf ک (U+06A9) and Arabic Swash Kaf ڪ (U+06AA). In Sindhi, the two are distinct letters but in Urdu the latter would be considered stylistic variation of the former character. Such decisions should be made by the relevant linguistic community.

Table 2.17. Urdu Language Characters Requiring Variant Mapping[18]

Character	Unicode	Mapped To	Unicode
ئ	0626	ء	0621
ة	0629	ۃ	06C3
ك	0643	ک	06A9
ه	0647	ہ	06C1
ى	0649	ی	06CC
ي	064A	ی	06CC
ۀ	06C0	ۂ	06C2
ڪ	06AA	ک	06A9

2.2.3.3 Script Level Case Folding

Among other properties, Unicode defines *Case* as a normative character property in certain scripts such as Latin, Greek, Cyrillic, Armenian, etc. [55]. The three case forms for characters in Unicode are Uppercase, Lowercase and Titlecase. Case folding is the process that maps a character in one case to its canonical form in another case, so that *caseless* comparisons can be performed. In IDNs, case-folding is required to establish equivalence between the two characters which convey the same meaning. Uppercase characters are not permitted to be used in IDNs. If any uppercase characters are encountered, they must be case-folded to lower case. An informative reference to some case-mapping possibilities is the mappings document [17]. These have to be done at application level, and also re-checked at the registry, instead of rejecting strings which may contain upper case letters. There can be some exceptions as well, where such mapping is ambiguous, e.g. Sigma and Eszett. Language community needs to decide the course of action in such cases.

2.2.4 Pragmatic Considerations - User Cultural Conventions and Preferences

In addition to encoding issues or visual similarity, there are additional limitations imposed by various ad hoc factors. These may include cultural conventions, localization support and similar reasons.

[18] Due to shape similarity in at least one of the four cursive forms.

2.2.4.1 Digits

Choice of digits has been discussed in the section on Digit Mixing earlier. However it must be reiterated that the even with a script having its own set of digits, a language community may arbitrarily decide to use a different set of digits. Such decisions are culturally driven and based on a variety of pragmatic and socio-linguistic reasons.

2.2.4.2 Label Separators

Labels in a domain name are separated using a label separator or delimiter. Dot "." is the only permissible delimiter "on the wire". However, certain languages use different delimiters to mark the end of a label. Preferred label separator for use in IDNs needs to be identified so that it can be mapped onto dot during pre-processing in the application. The document on mapping [17] in the IDNA200x standard suggests such possibility. As an example, Dzongkha language community suggests using the Rnam Bcad ་ (U+0F7F) as label separator. Urdu community uses Arabic Full Stop (U+06D4) generally, but has decided to switch to the dot for Internet domain names. This is similar to Nepali community preferring dot over the conventional Devanagari delimiter । (U+0964).

2.2.4.3 Honorifics and Other Symbols

Arabic script contains some non-spacing combining characters that are used with names of certain personalities as a symbol of respect. These are called honorifics and their use is part of the cultural and religious conventions. Thus they have been included for IDNs. However, due to the potential spoofing threats imposed by combining characters in general, on the recommendation of ASIWG, the Urdu language community has agreed to block them till a time when technology has matured to a level where these combining marks are not confusingly rendered[19].

2.2.4.4 Technology Maturity and Localization Support

There is a varying degree of support of different scripts on different technology platforms. Input methods, locales, fonts, rendering engines and other tools are still not mature for many scripts. This means that even if all the relevant homework is done for IDN deployment, the users still may not be able to benefit from their use. For example, ASIWG recommends restricting the use of combining marks in Arabic script at the script level until the technology advances to a point when these marks can be clearly displayed in address bars within web browsers, etc.

It should also be noted that while some native language characters are encoded in Unicode, they are not always supported by existing fonts for the language/script. Thus some characters of an IDN label or part of the label might not be displayed due to font issues. For example Arabic characters 063C..063F are not supported in the

[19] At this time, the address bar in web browsers is optimized for LDH characters, and is too short in height to display combining marks properly and would need to be re-engineered for properly displaying IDNs for many scripts.

font being used in the current system. These are PVALID and SVALID characters but appear as boxes: □, and thus would be difficult to use within domain names.
Many times when the support is available, keyboards are not available to input the text. For example it may not be easily possible to enter Arabic text for a person traveling through China, or to enter Greek text from Canada. There could also be cases that local language keyboards are not available even in the resident country of the language. Languages have found quite unique solutions to deal with such problems, Chinese Pinyin system being a good example, which allows users to use a Latin keyboard to enter Chinese characters.

3 RELAYING THE IDN LABELS "ON THE WIRE"

As discussed earlier, IDNs add a layer on top of the existing DNS. This layer inputs U-labels and converts them to A-labels, the latter being in the same format as labels in the existing DNS. Thus, the data being sent through the DNS, i.e. on the wire, remains unchanged. This has already been illustrated in Figure 1.3. The filtering and mapping role of each layer has been discussed in Chapter 2. This chapter focuses on the process needed for registering and resolving the IDNs.

3.1 PHYSICAL MAPPING OF THE LOGICAL LAYERS

The implementation of IDNs involves application, language, script and protocol level processing. This processing has to be done redundantly at three places: (i) client-end applications, (ii) registry and, (iii) root, for correctly registering and resolving a request. This section describes how these checks are distributed across different places during registration and resolution, as summarized in Figure 3.1.

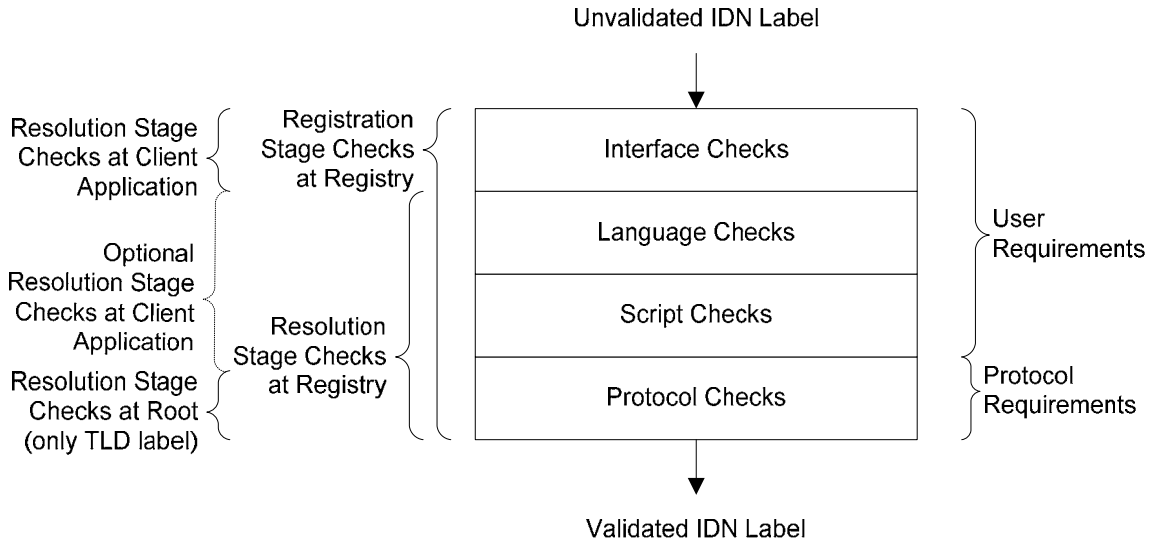


Figure 3.1. Distribution of Label Validation Application, Registry and Root during Registration and Resolution of IDNs

3.2 ROLE OF CLIENT APPLICATION

Client application is the web-browser, email client or any other program the user is using to generate a request to access online resources through DNS, e.g. to access a web page by clicking on a link. Client application does not play any significant role in registering a domain name, as this is normally done through submitting paperwork or through an online interface directly provided by the registry.

3.2.1 RESOLUTION PROCESS

Client application plays a significant role at the look-up time, for example, when the user submits a URI request through the web-browser. The application layer takes two input parameters: (i) an IDN (containing one or more Unicode code-points) to be looked-up, and (ii) a set of rules to process the IDN (which will have to be integrated within the application by the application developer). It is responsible for converting user input to A-label. This requires at least the following steps:

1. Ascertain the language/script of the input through user input or otherwise
2. Convert input to Unicode, in case another encoding is being used
3. Separate each label, using the label separator. This label separator may be the dot (U+002E) or some other language/script specific character (see Section 2.2.4.2)
4. Case fold characters, if applicable (see Section 2.2.3.3)
5. Normalize each label to Unicode NFC form (see Section 2.1.3.1)
6. Optionally perform protocol level analysis
7. Optionally perform language and script level analyses
8. Convert to A-label

Any mappings done in the process are not part of the IDNA 200x protocol and may vary with linguistic community. The protocol does provide an informative document for mapping [17], but specific needs have to be documented by the community itself and passed on to the application developers. This may include additional information on label separators, label display order specifications, guidelines on input method to U-label conversion and similar mechanisms, not part of the language and script tables.

The application may additionally perform protocol, script and language level checks. This would entail loading and running rules and tables. Though protocol level tables can be fed in, such tables would need to be updated with each version of Unicode[20]. Similarly, the language level tables may also be needed if script and language level processing is to be done. The relevant application can either directly ask the user to configure this information or alternatively pick it up from the locale. This mechanism will also eventually evolve. No guidelines for this process are provided by the IDNA standard[21].

How the A-label is displayed to the user when a query is returned is also the job of the application. Client application (web browsers, email clients, etc.) have to properly convert the A-label to U-label and do any reverse mappings to meet the

[20] A more stable solution would be to have a web-service which performs the check and is automatically updated with Unicode versions. Such a service could be used by the application layer, and provided by the relevant organizations, e.g. Unicode or ICANN, etc.

[21] In addition to the existing information, locale may eventually also be extended to contain community preferences for IDNs. This is one way to making the language tables available across all applications.

expectations of the user. This may require reversing some of the steps listed above, e.g. converting the dot back to the label separator used by the language.

3.3 ROLE OF REGISTRY

A registry supporting internationalized labels is required to perform appropriate checks both at the registration and the look-up time. The registry must perform protocol level checks, while also taking into account language-tables (including variant-tables) for validation of labels. Language-specific mappings and restrictions are applied on the top of the protocol level rules for each character. Furthermore, a registry might prohibit use of mixed-script labels and digit mixing as determined by the community, to prevent confusability. Various possible decisions relevant at registry level are discussed in the previous chapter and need to be applied at both registration and resolution levels. In the registration process, as the user is directly interacting with the registry interface and there is no application layer, the registry also needs to do application level processing. This is illustrated in Figure 3.2.

In addition to performing different verification checks, the registry also has to implement policy finalized to handle variants. Various approaches for variant handling are discussed in subsequent sections. Finally, the registry would also need to implement other policies discussed later in this section.

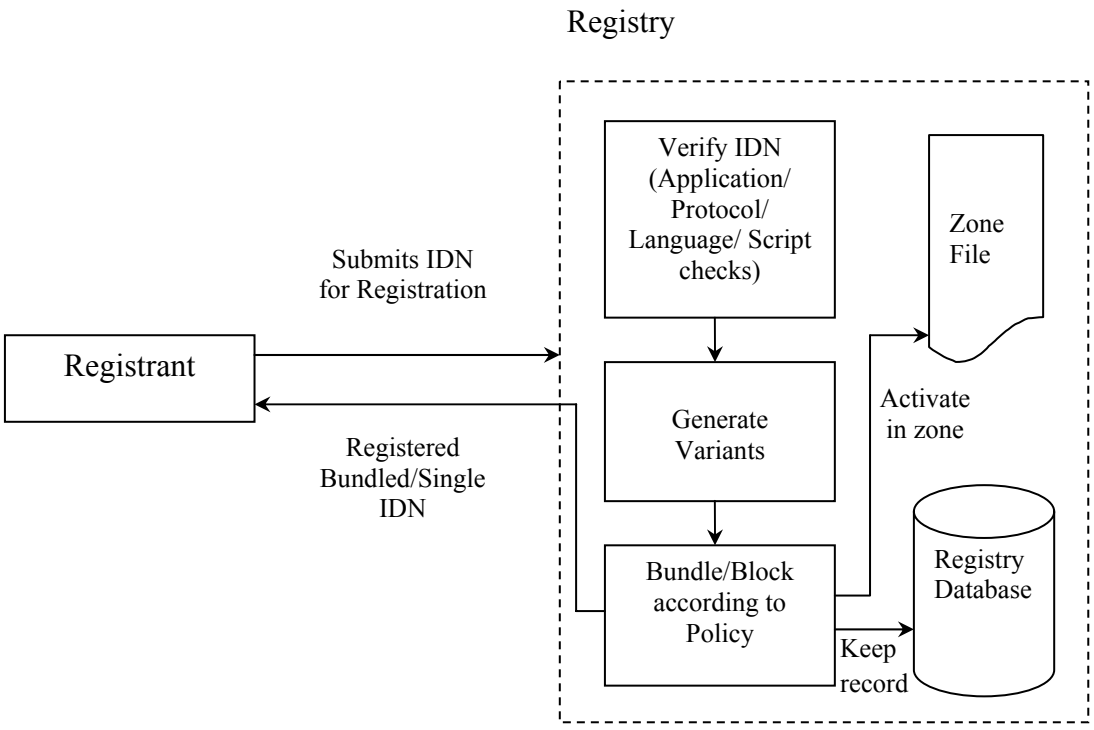


Figure 3.2. Block Diagram Showing IDN Registration Process

3.3.1 REGISTRATION PROCESS

To start the registration process, a registrant submits an IDN for registration. This might be done by filling out an online form. In this case, the registry providing the

service is responsible for handling application level checks on the IDN. These include checking encoding of the internationalized string, separating labels based on local label separator and normalizing the string to Unicode NFC. The actual registration checks are performed on the putative U-label. Figure 3.3 depicts a possible way of registry level verification before a U-label is converted to an A-label and placed in the zone. This step-by-step process is detailed below.

1. When a domain name is submitted by the registrant for registration (through a web interface or otherwise), the application layer of the registration service checks the encoding of the label and notifies the applicant to submit a Unicode string if it is not in Unicode[22].

2. Labels in the IDN string are separated using the label separator. There may be language/script specific label separators which may be used.

3. Each label is checked for normalization, and converted to NFC if required.

4. A label containing any DISALLOWED and/or UNASSIGNED characters is rejected.

5. Protocol level label validation checks are performed. Any label that contains any of the following is rejected.

    a. Hyphen in third and fourth positions.
    b. Leading combining mark.
    c. Context character for which no rule is defined. This includes both the CONTEXTO/J characters [14].
    d. Context character for which a rule is found and the context does not agree with that rule.
    e. Labels containing mixed direction characters violating bidirectional rules as given in the IDNAbis-Bidi document [15].

6. If a label passes the above tests, it is a U-label. This label can be subjected to further registry level restrictions which are a localized function of the registry. The following registry level tests may be performed. Language tables and rules are input to this step of the registration process.

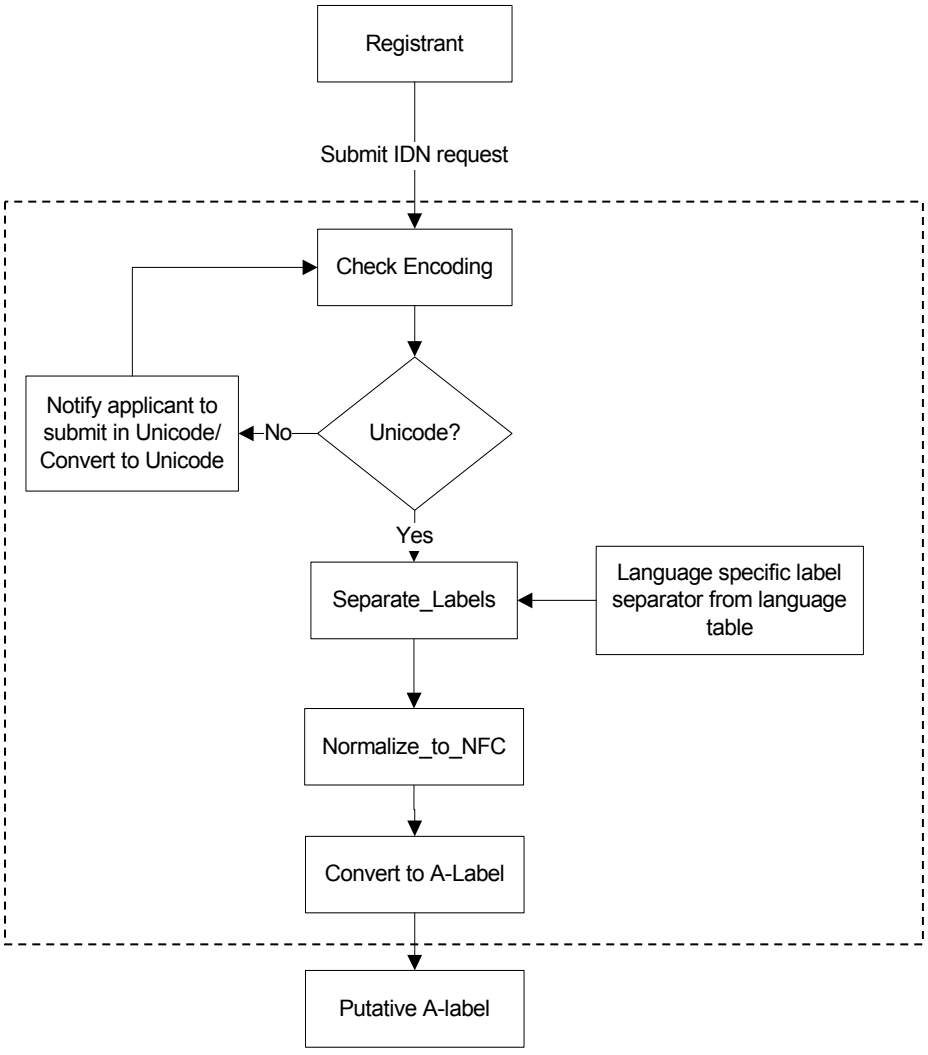
    a. Any additional characters that are prohibited by the registry are checked. Labels containing all such characters will be rejected by the registry.
    b. If registry does not allow script mixing, a label found with mixed scripts (i.e. abcاب) is rejected. Same is done for digit mixing.

[22] Or alternatively a service may convert the sting to Unicode, for communities which are still predominantly using non-Unicode fonts.

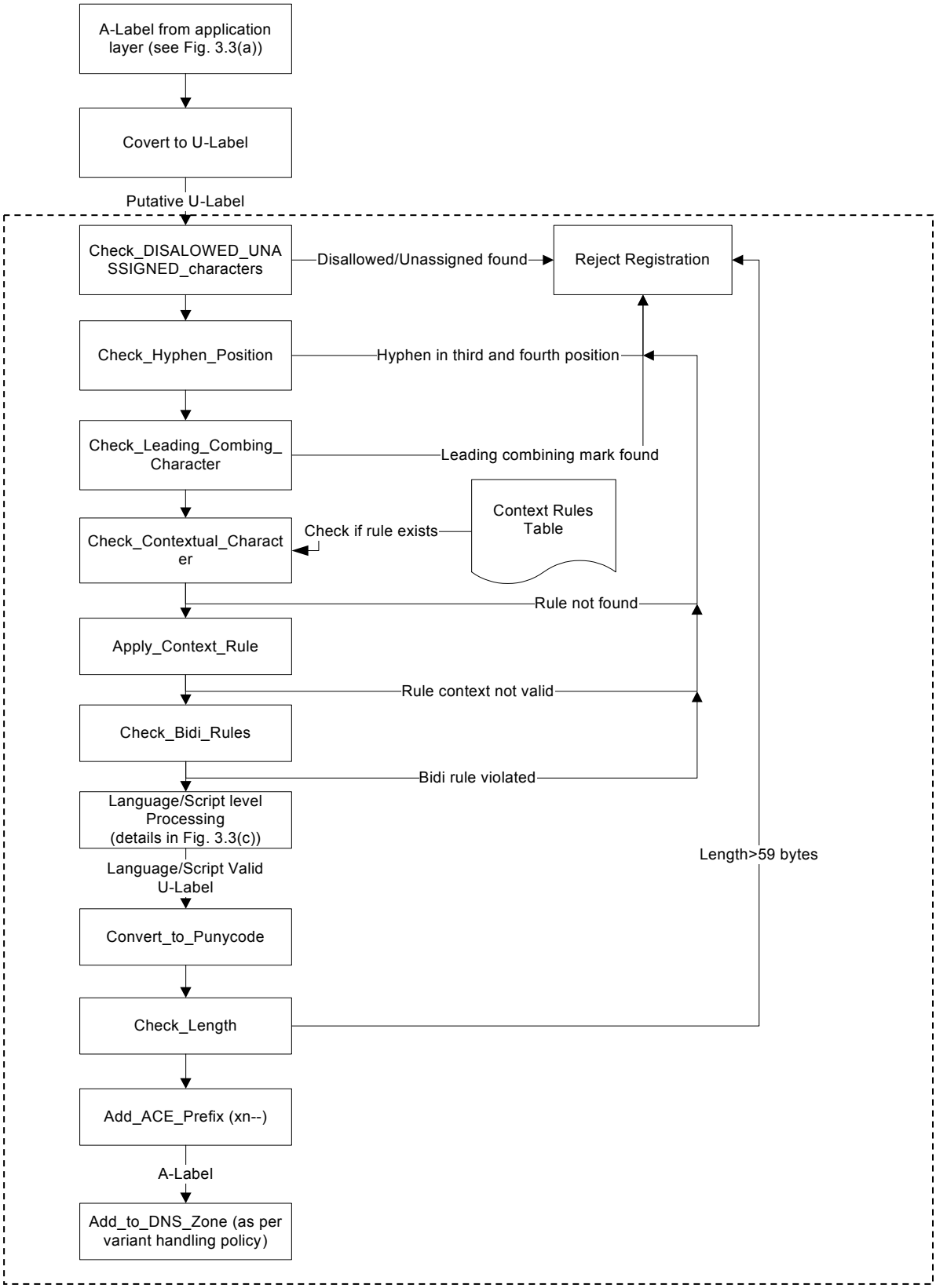
c. Variants are generated at this step of the process using language variant tables. All labels generated are either bundled with the original label (and activated in the zone), or blocked from registration by any other registrants. This depends on the registry policy. An entry in registry database is placed for any bundled labels.
d. The resulting U-label(s) are converted to punycode.
e. The label(s) should fulfill the DNS length requirements for ASCII labels (which is 59 bytes or less).
f. ACE-prefix (xn--) is appended to the result of punycode.
g. The resulting A-label(s) are placed in the registry name-server's zone file.

7. If the label can be registered as a valid U-label, the registrant has to provide contact information to complete the registration process. This includes providing details such as name, email, administrative contact, administrative address, billing contact and technical contact for the domain name. This information is also placed in the Whois database of the registry for future public access and queries.

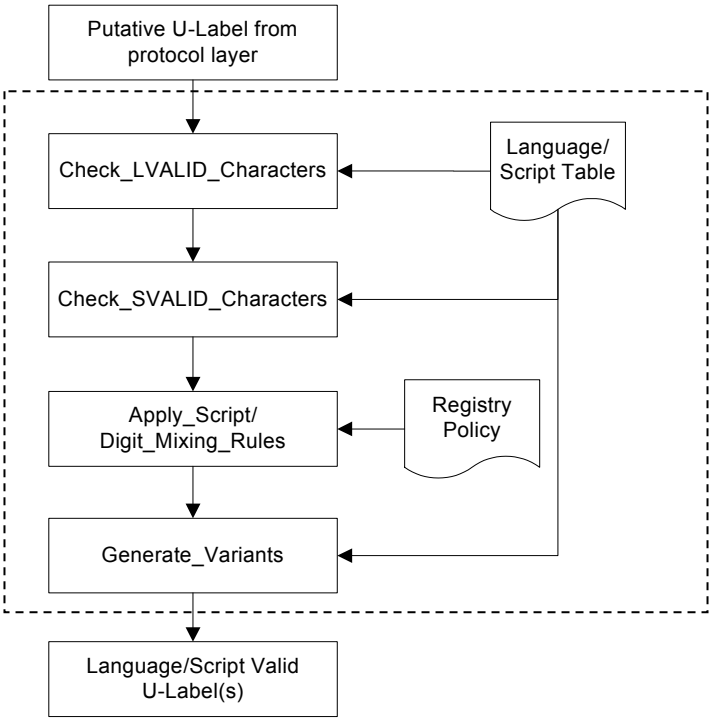
This process is illustrated in the figures below. The figures have been divided into three portions, Figure 3.3 (a) show the Application layer processing, (b) shows the protocol layer processing, and finally (c) details the language and script level processing.


Registrant
Submit IDN request
Check Encoding
Unicode?
No
Notify applicant to submit in Unicode/ Convert to Unicode
Yes
Separate_Labels
Language specific label separator from language table
Normalize_to_NFC
Convert to A-Label
Putative A-label

(a)


A-Label from application layer (see Fig. 3.3(a))
Covert to U-Label
Putative U-Label
Check_DISALOWED_UNASSIGNED_characters
Disallowed/Unassigned found
Reject Registration
Check_Hyphen_Position
Hyphen in third and fourth position
Check_Leading_Combing_Character
Leading combining mark found
Context Rules Table
Check_Contextual_Character
Check if rule exists
Rule not found
Apply_Context_Rule
Rule context not valid
Check_Bidi_Rules
Bidi rule violated
Language/Script level Processing (details in Fig. 3.3(c))
Length>59 bytes
Language/Script Valid U-Label
Convert_to_Punycode
Check_Length
Add_ACE_Prefix (xn--)
A-Label
Add_to_DNS_Zone (as per variant handling policy)

(b)



(c)

Figure 3.3. Registration of IDNs: (a) Application layer processing, (b) Protocol level processing, and (c) Language/Script level processing

3.3.2 REGISTRATION POLICY

Some issues relating to IDNs have to be resolved by the registries through policy making; the protocol does not layout any specific rules or constraints for them. One such issue relates to script/digits mixing. At present, ICANN does not allow the use of mixed-script IDN labels, as documented in the IDNs implementation guidelines [40]. Possible exceptions to this case are languages that require comingled use of scripts such as the CJK languages.

Another question that has to be addressed through policy formulation relates to variant management. As discussed in the previous chapter, languages and scripts contain several similar looking characters that are a source of visual confusion. Some of these characters look exactly the same, while other may be slightly different. Such characters, also called confusable characters, create a phishing possibility when used in IDNs. Typically, a registrant of IDNs would certainly want all similar looking labels to point to the same Internet resource, to avoid user confusion. For e.g. in Arabic the characters ک (U+06A9) and ك (U+0643) are exact shape in their initial and medial forms. Consequently, the word پاکستان (Pakistan) with either character will look exactly the same. There are other types of variants, based on shape and/or meaning similarity, even when the match is not exact.

There are a number of issues associated with how variants are managed by the registry, including allocation, delegation, billing and dispute resolution. A registry receiving variant allocation requests must maintain a variant table in addition to the language table. Alternatively, information about characters having variants can be stored in the same table. The following section lists possible variant management mechanisms that can be implemented by a registry.

3.3.2.1 Handling Redundancy

When a registry receives a label for registration that contains characters for which variants/equivalents exist, it is up to the registry to do one of the following. A variant management policy must be developed by the registry to reflect its decisions in this regard.

i. Single registration

Requested label is registered in the zone file as single entity. This means that no variant names are registered (placed in the zone file), or blocked for registration in the future, or reserved for the applicant. Thus, any other registrant can register the variant domain name separately from the original base label. This is not a preferred method as it would create user confusion and possible phishing problems.

ii. Bundling

Requested label and its preferred variants are allocated to the same registrant in a bundle. The registrant may receive two or more delegations for the same or additional price. Bundling is also required for maintaining backwards compatibility with IDNA2003. Query for a bundled domain name variant will result in a successful look-up.

iii. Blocking

The requested label is registered and delegated to the registrant, while all other variants are blocked for registration. This means that the variant labels do not appear in the zone file and can never be registered. A blocked variant domain name will result in a negative response to a DNS query. Though user experience is not as good as bundling, but security issues are addressed.

iv. Reservation

The requested label is registered and delegated. Preferred variants are reserved and all other variants are blocked from registration. Reservation implies that only the registrant can release the reservation and register the domain name in question. This may be the preferred way if incremental pricing is to be instituted.

3.3.2.2 Dispute Resolution

A uniform dispute resolution policy (UDRP) defines how the registry settles disputes when two simultaneous requests for registration of the same domain name are received. In case of IDNs, it also includes how disputes over variant domain

names are resolved. Similar measures need to be in place for trademark protection. Sunrise periods are normally used by registries implementing dispute resolution for trademark handling.

3.3.2.3 Whois[23]

A Whois service is provided by most registries to make information about the registered domain names publically available. This usually contains the administrative and technical contacts of a domain and the details of its name-servers. The Whois service is used to query about a specific domain and may also be used to check the availability of domain names. Currently the Whois servers store information as plain ASCII text. This may also need to be localized in case of IDNs.

Moreover, a registry might need to keep multiple records for the same registrant if variants are also allocated and/or delegated. Since a Whois server is usually queried for the availability of a domain name, this information must be available to notify the potential registrant of any reserved and/or blocked variants.

3.3.2.4 Pricing

The IDN registration policy must also detail how the registrant will be billed, according to its business model. This is especially important for variant management. The registry has to decide whether the bundled and/or reserved variant package comes at an extra price or allocated as a unit.

3.3.2.5 Security Considerations

Finally, a registry implementing IDNs needs to address security concerns associated with visually confusable characters. A registry has necessary data available about alternative registered names, and can process that information efficiently at the time of registration, using policies to reduce phishing. For example a registry might restrict use of script and/or digit mixing as part of its security policy (also see [23]). The registry may also require both A-label and U-label pairs of IDN labels as a compulsory registration practice to ensure correct registration.

3.3.3 Resolution Process

Domain name resolution, also called DNS look-up, refers to the retrieval of a record from the DNS given a domain name. When a user enters a URL in an address bar, or clicks on a link, a series of steps are performed to provide access to the requested resource. The IDNA200x protocol treats registration and look-up as separate processes. The IDNA200x look-up protocol involves less restrictive checks on the domain name than those performed at registration time, because it is assumed that the domain names already present in the DNS are valid.

The IDNA200x domain name resolution process is summarized below.

1. The user submits an IDN, e.g. by typing it in the address bar of the browser or clicking on a hyperlink.

[23] This policy for IDNs is not yet finalized.

2. The application layer converts the input to an A-label, using the steps discussed in Section 3.2.1. Protocol level checks are necessary in this case. It may be noted that at resolution time CONTEXTJ rules are applied but CONTEXTO rules are not applied [18]. Language and script level checks may optionally be performed.

3. The A-label is looked up in the DNS, using normal DNS resolution procedures and given in Figure 1.2. As all variants are already handled by making requisite redundant entries in the zone files during the registration, there is no need to generate any variants in the look-up process.

3.4 ROLE OF ROOT

The role of root is limited to the top-level domains (TLDs), while all the other labels are handled by the TLD registry directly.

Consequently, root is only responsible for registering ccTLDs and gTLDs. This is also true for IDNs, i.e. root is responsible for registering all IDN ccTLDs and IDN gTLDs. An IDN TLD is inserted in the IANA root zone database after protocol level checks have been performed and the label is validated. The delegation process is complete after the IDN TLD has been inserted at the root. An IDN TLD registry may register its IDN tables with IANA, which makes these publically available on its website [36]. The Fast Track process of ICANN puts further restrictions on registration of IDN TLDs. It is discussed in the next chapter.

It should be noted here that like second-level IDN labels, the IDN TLD label might also have variants. So, a variant management policy at the root level is also needed for registration and look-up. This becomes even more important in cases where the variants look exactly the same. Two alternatives can be adopted, i.e. either reserve preferred variants and block all other variants, or delegate preferred variants and block all other variants[24]. Only delegation of variant TLD in the root would allow it to be looked-up. It can technically be managed either through twin delegations in the root for the same TLD (with two NS records), or with the use of DNAME resource record [34] in the root to enable domain aliasing. The policy of how variants will be managed at root level is still being devised at the time of publication of this report. Once the IDN TLDs are delegated, the resolution process will proceed as per the variant management policy deployed.

3.5 CASE-STUDIES

This section describes efforts of some individual regional work groups towards the implementation of IDNs in their specific scripts and languages. The CJK/Japanese, Russian, Tamil and Arabic languages are included.

[24] This is being discussed in the context of IDN Implementation Support working group at the time of publication of this work.

3.5.1 JOINT ENGINEERING TEAM (JET)

3.5.1.1 History

Joint Engineering Team was established in 2000 to investigate the technical issues faced by Japanese Network Information Center (JPNIC under the Asia Pacific Networking Group (APNG). The group worked on the CJK (Chinese-Japanese-Korean) languages. JET was established as a result of MINC (Multilingual Internet Naming Consortium) activities of bringing together regional groups in the Asia Pacific regions and coordinating their activities towards the deployment of domain names in regional languages. The team comprises of members from CNNIC (Chinese Network Information Center), TWNIC (Taiwan Network Information Center), KRNIC (Korean Network information Center) and JPNIC (Japanese Network information Center).

3.5.1.2 Activities

The purpose of establishment of JET was to develop language tables for CJK languages. The workgroup released the guidelines for the registration and administration of IDNs for CJK which were published in an IETF informational document [30]. IESG made a specific note on this effort of JET and advised all other language communities to adopt JET's approach in developing language and variant tables as well as enforcing guidelines and policy decisions regarding IDN registration.

JET tries to address the deployment, registration and policy enforcement issues related to IDNs through proper administration at the registries instead of just restrictions at the protocol level. The RFC published in this respect specifically aims at resolving confusable-character issue through the use of Language Variant Tables (LVTs) to reflect the language preferences.

An LVT is a three-column table. The first column corresponds to a list of Valid Code Points (VCPs). These are the code-points which are required by the language and are allowed in the protocol as well. The second column of the table lists code-points against each VCP which provide suitable substitution for it. These are called Preferred Variants. The third and final column of the table specifies a second list of code-points against each VCP. These are called Character Variants. Preferred and Character Variants are different from each other in that the former are actually registered in a zone file and activated whereas the character variants are not registered in the zone file but rather reserved for each valid code-point. Thus a registrant registering a domain name ends up with a bundle of domain names (with Preferred and Character Variants) called an IDN package. Domain name are registered and deleted from a zone in the form of an atomic IDN package.

3.5.2 RUSSIAN LANGUAGE WORK GROUP /CYRILLIC LANGUAGE INTERNET NAMING CONSORTIUM

3.5.2.1 History

The Russian Language Work Group (RLWG) was created in November 2001 under MINC. Later, with support of Russian Chamber of Commerce and prominent Russian registries, Russian Language Network Information Center (RLNIC) and Cyrillic Languages Internet Names Consortium (CyrLINC) were formed. These entities, with technology licensed from iDNS.net, aimed at launching interoperable IDN testing within Russia. The role of the RLWG has been to ensure that language and policy issues relating to the Russian script as used in Russia will be adequately adhered to by all parties.

CyrLINC [31] aims to coordinate efforts to develop a multilingual Internet addressing system within Cyrillic script communities. The Cyrillic script is used for writing a number of languages including Russian, Byelorussian, Ukrainian, Bulgarian, Macedonian, Mongolian, Serbian, Kazakh, Kirghiz, Uzbek, Azerbaijani, Bosnian, Turkmen etc. To achieve the projected objectives, CyrLINC plans to work as follows.

1. Consider and discuss IDN related policy issues
2. Develop protocols related to internationalized Internet addressing systems
3. Manage the implementation of IDN system for registries, registrars and related organizations

3.5.2.2 Activities

A Best Current Practices document titled “Internationalized Domain Names Registration and Administration Guideline for Russian, Ukrainian, Bulgarian and Byelorussian languages in ASCII TLDs” has been published by this working group. It was published in 2003 and revised in 2007 [28].

This document is a guideline for registries on registering IDNs in these languages. The first step is to identify a set of valid code-points that will be accepted under IDNA for the language. Again, the approach adopted here is that of using variant tables that list all valid code-points and also specifies additional character variants that might cause confusion. It is up to the registry whether to register IDNs in a package (like CJK guidelines), or register one domain name and reserve all the rest containing confusable/variant characters. These tables are language-specific. The registry can either allocate all the labels (valid and variant) to the same registrant, block all other labels to prevent any further registrations of the variant labels or use a hybrid allocation reserving some labels while permitting others to be registered to different registrants.

RWLG is currently not active.

3.5.3 INTERNATIONAL FORUM FOR INFORMATION TECHNOLOGY IN TAMIL

3.5.3.1 History

International Forum for Information Technology in Tamil [32] was formed in 2000 based on a pre-INFITT group under MINC. It is a worldwide group with 13 member countries – India, Singapore, Malaysia, UK, USA, Sri Lanka, Germany, Switzerland, Mauritius, South Africa, Canada, Hong Kong and Australia. The IDN working group (WG-03) was initiated in 2000. The initial objectives of this work group were to collaborate with MINC and other appropriate stakeholders to coordinate the test deployment of Tamil domain names, to participate in any language or application interoperability testing, and to recommend to INFITT on the feasibility of operational Tamil domain names.

3.5.3.2 Activities

The IDN gTLDs equivalent for .com, .net, org, .edu and .gov were finalized and released in 2001 as shown below and were approved by INFITT. It was decided to expand the range of TLDs in Tamil and also release equivalents for country-code top level domains.

Table 3.1. gTLD Translations in Tamil

com	வணீ
net	இணை
org	அமை
edu	கல்வீ
gov	அரசு

3.5.4 ARABIC SCRIPT IDNS WORKING GROUP

3.5.4.1 History

Arabic Script IDNs Working Group (ASIWG) was established in early 2008 as a self-organizing group involving members from language communities that use Arabic script in their writing systems [33]. The work group aims at providing a framework for the deployment of Arabic script IDNs on the Internet. The initial work group goals include developing language tables for Arabic script IDNs, resolving technical and policy issues associated with the deployment and use of Arabic IDNs and proposing guidelines on implementation of Arabic IDNs. It includes representations from ICANN, Unicode, IETF, ISOC Africa and Arab League, in addition to members from Egypt, Iran, Kuwait, Pakistan, Saudi Arabia, Syria, UAE, Malaysia, Jordan, US and other countries. A number of languages are represented including Arabic, Persian, Urdu, Sindhi, Pashto and Jawi.

3.5.4.2 Activities

Four meetings of the work group have been held so far, with major outcomes relating to the development of language tables. Some major issues concerning

Arabic script domain names are the presence of confusable characters within and across the script, use of ZWJ/ZWNJ, more than one digits block, non-spacing characters and development of variant tables. The work group has prepared a set of documents on the above mentioned issues and identified the sets of visually similar characters. It has also proposed a layered model to classify handling of issues at four different levels.

1. *Protocol Level* – protocol level decisions about characters are made at this level. This reflects the PVALID, DISALLOWED and CONTEXTJ/O properties of code-points. The group has given feedback to the protocol process on character properties within the Arabic script block and also the rules concerning the contextual character ZWNJ. The issue of digit mixing has also been discussed in detail and feedback has been integrated into the protocol development process.

2. *Script Level* – this level takes into account issues pertaining to the Arabic script in general. This is applicable to all languages using the script. The group has agreed to block certain characters across all languages, including some combining marks, especially until the technology is mature to handle them without creating user confusion. Extended normalization is also being documented.

3. *Language Level* – this is the level where registries implement/enforce language level rules and preferences. Decisions taken and restrictions imposed at this level are specific to a particular language. The confusable character issue is addressed at this level. The group has been collecting different language and variant tables for various languages in this context.

4. *User Level* – this the top-most level where application level issues are addressed such as rendering and display of text to the user. Normalization and other relevant issues are being documented by ASIWG at this level.

The work group is currently active.

4 THE NEXT STEP - FAST TRACK PROCESS

As a result of the final report of IDNC WG in June 2008, ICANN approved the development of a Fast Track implementation plan [37] for introducing a limited number of IDN country-code Top Level Domains (ccTLDs) corresponding to the two-letter ISO-3166 codes [38] in the root. The draft plan lays out the procedure for countries and territories to request their country name in the local language or script as an IDN ccTLD and is based on the feedback received during different public comment periods since 2008. The proposed launch for ICANN ccTLD Fast Track Process is in November, 2009.

4.1 PARTICIPATION CRITERIA

Countries interested in participating in the Fast Track process are required to meet the following criteria:

1. The country must be listed in the International Standard 3166-1 two-letter country codes list.

2. The IDN ccTLD manager must show documented support from the country or territory corresponding to the relevant ISO 3166-1 entry. This could be signed letter of support from a governmental authority or a senior representative of the department which is responsible for domain name administration.

4.2 IDN ccTLD STRING CRITERIA

The proposed IDN ccTLD string should be a minimum of two characters long in Unicode, while its length should not exceed 63 characters in ACE. The language of the string must be an official language of the country. This should be supported by a letter of verification by the public authority. In addition, languages based on Latin script are not eligible for the Fast Track process.

The IDN ccTLD string must be a meaningful representation of the country name. This has to be supported by documented evidence from an internationally recognized expert or organization. An ICANN accredited list of such organizations is given in [39].

Only one string per official language or script per country is allowed. For those countries that require delegation of variant TLDs in the root, the desired variants have to be submitted along with the requested IDN ccTLD. These will be reserved based on the string evaluation criteria, but not necessarily delegated.

Finally, the requested string must fulfill the technical string requirements as specified in the IDNA protocol and ICANN guidelines [40].

4.3 GENERAL PROCESS OVERVIEW

The general Fast Track process is divided into three phases [39].

1. *Preparation Stage*
   The requester identifies and selects the languages of the IDN ccTLD string, selects a meaningful string and develops the necessary IDN tables and variant tables. In addition, the requester prepares relevant documents from the public authority as evidence for the government and local community support, official language and meaningfulness of string.

2. *Request Submission for String Evaluation Stage*
   In the string evaluation stage, ICANN carries out a number of steps namely request completeness validation, linguistic process validation, DNS stability evaluation and publishing of validated string. This is to ensure that requests are complete in all respects with required documented evidence of support and the string fulfills all technical and language requirements.

3. *Request Submission for Delegation Evaluation Stage*
   After a request has successfully passed Stage-2, the standard IANA delegation function is performed as already followed for ASCII two-letter ccTLDs [41]. The steps followed by IANA ensure that the prospective ccTLD manager has requisite technical competence and resources to administer the domain. It also verifies that the ccTLD manager has necessary authority and support of the respective country's government to operate the TLD appropriately.

This follows a final ICANN Board review to evaluate whether the requests are consistent with administration policies and ICANN bylaws. After the approval of the request, ICANN follows its regular IANA root zone change function and the ccTLD is delegated in the root zone.

4.4 CONCLUSION

After many years of patience and perseverance, IDNs are soon to become a reality. It is certainly a significant new chapter in the evolution of the Internet. However, with many core issues still not well-articulated, this step forward will certainly have its share of challenges. Variant definition, variant handling policy, Whois policy, significant expansion of namespace, dispute resolution and many other issues still await clear specifications from the community. Nevertheless, the resolve, which the community has already shown, leaves no doubt that the challenges will be addressed as we gain more experience with IDNs.

REFERENCES

[1] W3C Internationalization Activities, FAQ: Internationalization vs. Localization. Accessed from http://www.w3.org/International/questions/qa-i18n.

[2] P. Twomey (2007). "Effect of Multilingualism on the Internet. International Issues that Affect How Governments and Economies Address Issues Relating to a Global Infrastructure." NSF/OECD Workshop on Social and Economic Factors Shaping the Future of the Internet.

[3] P. Mockapetris (1987). "Domain Names - Concepts and Facilities." RFC 1034, IETF. Accessed from http://www.ietf.org/rfc/rfc1034.txt.

[4] P. Mockapetris (1987). "Domain Names - Implementation and Specification." RFC 1035, IETF. Accessed from http://www.ietf.org/rfc/rfc1035.txt.

[5] T. Berners-Lee, R. Fielding, L. Masinter (2005). "Uniform Resource Identifier (URI): Generic Syntax." RFC 3986, IETF. Accessed from http://www.ietf.org/rfc/rfc3986.

[6] Wikipedia. "Root Nameserver." Accessed from http://en.wikipedia.org/wiki/Root_nameserver.

[7] J. Seng (2009). "History of IDN." Presented at APNIC meeting Beijing, accessed from http://meetings.apnic.net/data/assets/pdf_file/0014/14009/seng-idn-overview.pdf.

[8] P. Hoffman (2002). "Preparation of Internationalized Strings ("stringprep")." RFC 3454, IETF. Accessed from http://www.rfc-editor.org/rfc/rfc3454.txt.

[9] P. Faltstrom, P.Hoffman, A. Costello (2003). "Internationalizing Domain Names in Applications." RFC 3490, IETF. Accessed from http://www.rfc-editor.org/rfc/rfc3490.txt.

[10] P. Hoffman, M. Blanchet (2003). "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)." RFC 3491, IETF. Accessed from http://www.rfc-editor.org/rfc/rfc3491.txt.

[11] A. Costello (2003). "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)." RFC 3492, IETF. Accessed from http://www.rfc-editor.org/rfc/rfc3492.txt.

[12] J. Klensin, P. Faltstrom, C. Karp (2006). "Review and Recommendations for Internationalized Domain Names (IDNs)." RFC 4690, IETF. http://www.rfc-editor.org/rfc/rfc4690.txt.

[13] The Unicode Consortium. Unicode character data. Accessed from http://www.unicode.org/Public/UNIDATA/UnicodeData.txt.

[14] P. Faltstrom (2009). "The Unicode code points and IDNA." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/draft-ietf-idnabis-tables-07.

[15] H. Alvestrand, C.Karp (2009). "Right-to-Left Scripts for IDNA." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/draft-ietf-idnabis-bidi-06.

[16] M. Davis, K. Whistler, M. Dürst (2009). "Unicode Normalization forms." Unicode Standard Annex #15, Unicode Consortium. Accessed from http://unicode.org/reports/tr15.

[17] P. Resnick, P. Hoffman (2009). "Mapping Characters in IDNA." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/draft-ietf-idnabis-mappings-04.

[18] J. Klensin (2009). "Internationalized Domain Names in Applications (IDNA): Protocol." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/draft-ietf-idnabis-protocol-16

[19] K. Harrenstien, M. Stahl and E. Feinler (1985). "DOD Internet Host Table Specification." RFC 952, IETF. Accessed from http://tools.ietf.org/html/rfc952.

[20] The Unicode Consortium (2009). "Property Value Aliases". Accessed from http://unicode.org/Public/UNIDATA/PropertyValueAliases.txt.

[21] M. Davis (2009). "Unicode Identifier and Pattern Syntax." Unicode Standard Annex #31, Unicode Consortium. Accessed from http://unicode.org/reports/tr31/.

[22] R. A. Wasala, C. Liyanage, H. Wijayawardhana, R. Weerasinghe (2008). "Implementation of Internet Domain Names in Sinhala." PAN Localization Project.

[23] M. Davis (2006). "Unicode Security Mechanisms." Unicode Technical Standard #39, Unicode Consortium. Accessed from http://www.unicode.org/reports/tr39/.

[24] P. Constable (2004). "Proposal on Clarification and Consolidation of the Function of ZERO WIDTH JOINER in Indic Scripts". Public Review Issue # 37, Unicode Consortium. Accessed from http://www.unicode.org/review/pr-37.pdf.

[25] P. Faltstrom (2009). "The Unicode code points and IDNA." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/draft-ietf-idnabis-tables-07.

[26] Right-to-left scripts for IDNA (draft-ietf-idnabis-bidi-06). http://tools.ietf.org/html/draft-ietf-idnabis-bidi-06.

[27] The Unicode Consortium (2006). "East Asian Scripts". *The Unicode Standard 5.0*. Addison Wesley, USA.

[28] Sergey Charikov (2007). "Internationalized Domain Names Registration and Administration Guideline for Russian, Ukrainian, Bulgarian and Byelorussian languages in ASCII TLDs." Internet Draft. Accessed from http://www.cyrlinc.org/docs/draft-cyrlinc-idn-reg-01.txt.

[29] J. Klensin (2009). "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework." Internet Draft (work in progress). Accessed from http://tools.ietf.org/html/ draft-ietf-idnabis-defs-11.

[30] K. Konishi, K.Huang, H. Qian, Y. Ko (2004). "Joint Engineering Team (JET) Guidelines for Internationalized Domain Names (IDN) Registration and Administration for Chinese, Japanese, and Korean." RFC 3743, IETF. Accessed from http://tools.ietf.org/html/rfc3743.

[31] Cyrillic Languages Internet Names Consortium. http://www.cyrlinc.org/
[32] International Forum for Information Technology in Tamil. http://www.infitt.org/.
[33] ASIWG. http://www.arabic-script-domains.org/wiki/Main_Page.
[34] M. Crawford (1999). "Non-Terminal DNS Name Redirection." RFC 2672, IETF. Accessed from http://www.ietf.org/rfc/rfc2672.txt.
[35] J. Gargano, K. Weiss (1995). "Whois and Network Information Lookup Service, Whois++." RFCs 1834, IETF. Accessed from http://www.faqs.org/rfcs/rfc1834.html.
[36] IANA Repository of IDN Practices. Accessed from http://www.iana.org/domains/idn-tables
[37] ICANN's ccTLD Fast Track Process. http://www.icann.org/en/topics/idn/fast-track/.
[38] ISO Maintenance Agency. http://www.iso.org/iso/english_country_names_and_code_elements.
[39] Internet Corporation for Assigned Names and Numbers (ICANN) (2009). "Fast Track Implementation Plan." Accessed from http://www.icann.org/en/topics/idn/fast-track/idn-cctld-implementation-plan-30sep09-en.pdf.
[40] Internet Corporation for Assigned Names and Numbers (ICANN) (2007). "Guidelines for the Implementation of Internationalized Domain Names". Accessed from http://www.icann.org/en/topics/idn/implementation-guidelines.htm.
[41] IANA Guide to Delegation Procedures. http://www.iana.org/domains/root/delegation-guide/.
[42] Internet Corporation for Assigned Names and Numbers. http://www.icann.org/en/about/.
[43] ICANN Mission. http://www.icann.org/en/general/bylaws.htm#I.
[44] Internet Engineering Task Force (IETF). http://www.ietf.org/index.html .
[45] IETF's Mission. http://www.ietf.org/rfc/rfc3935.txt .
[46] The Internet Society (ISOC). http://www.isoc.org/isoc/.
[47] Governmental Advisory Committee (GAC). http://www.icann.org/en/committees/gac/.
[48] ccNSO IDNC WG. http://ccnso.icann.org/workinggroups/idncwg.htm.
[49] GNSO. http://gnso.icann.org/.
[50] ICANN's PAC for IDNs. http://www.icann.org/en/committees/idnpac/.
[51] ASO. http://aso.icann.org/about.html.
[52] ALAC. http://www.atlarge.icann.org.
[53] Internet Governance Forum. http://www.intgovforum.org/cms/index.php/aboutigf.
[54] World Summit on Information Society. http://www.itu.int/wsis/index.html.
[55] The Unicode Consortium (2006). "Character Properties". *The Unicode Standard 5.0*. Addison Wesley, USA.
[56] The Unicode Consortium (2006). "General Structure". *The Unicode Standard 5.0*. Addison Wesley, USA.

[57] Internet Corporation for Assigned Names and Numbers (ICANN) (2009). "Draft Applicant Guidebook version 3." Accessed from http://www.icann.org/ en/topics/new-gtlds/draft-rfp-clean-04oct09-en.pdf.