The IDN Variant Issues Project: A Study of Issues Related to the Delegation of IDN Variant TLDs

Discussion Draft - 19 December 2011

Contents

1 Overview of this Report

This integrated issues report considers the issues associated with the possible inclusion in the DNS root zone of IDN variant TLDs. It builds on the work of six case study teams who examined the range of variant issues associated with particular scripts. See Appendix 1 for references to the case study issues reports.

In developing this integrated issues report, ICANN has used the work of the case studies as background to create a common framework for consideration of these issues.

The introductory sections (1 & 2) provide a context for the subject matter and the scope of the project undertaken by ICANN. The identified issues discussed in the report are organized under three broad headings:

- Issues concerning how to classify the range of potential variant cases identified in the script team reports. These are discussed in section 3 of the report.
- Issues concerning how variant TLDs are established. These are discussed in section 4 of the report.
- Issues concerning how variant TLDs are treated once established. These are discussed in section 5 of the report.

Additional related issues, such as those associated with the use of **code points** not currently permitted in the root zone, are discussed in section 6.

A review of the issues in the report, and possible considerations for next steps are discussed in section 7.

The sets of issues in this report are analyzed in light of certain overriding considerations: the security and stability of the DNS, and user experience. Maintaining the security and stability of the DNS is central to ICANN's core mission. It is of the utmost importance that the actual operation and maintenance of the DNS, on which many services rely, are not adversely impacted by the introduction of IDN variant TLDs. Where relevant, issues specifically impacting the stability of the DNS are discussed. Secondly, user experience considerations are a key theme throughout the discussion, with an extended examination of these issues in section 5. These issues concern avoidance of delegating variant TLDs in a manner that creates user vulnerabilities or a probability of confusion, as well as an interest in functionality and efficiency for the user experience.

1.1 Fundamental Assumptions

The mission of ICANN 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. When considering the possible

delegation of IDN variant TLDs, ICANN has the responsibility to undertake these activities in a manner that will not adversely affect the security or stability of the DNS.

At the current time, a variant management mechanism for the top level does not exist. In considering the issues associated with developing such a mechanism, certain existing references are used as an underlying foundation. The Unicode standard[1] (currently version 6.0) provides a repertoire of code points used in world scripts, including various classifications of character properties, and normalization rules. The Internationalizing Domain Names in Applications (IDNA) protocol (RFCs 5890-5)[2] specifies rules for determining whether a code point, considered in isolation or in context, is a candidate for inclusion in a **domain name**.

It is assumed that these reference points will continue to be applicable to the DNS. In addition to the assumption of stable references, ICANN has also distilled from the case study team reports and the mission and core values of ICANN a set of fundamental assumptions, articulated below, which have been used in the development of this report.

1. The root zone is a shared resource, and the management of the root zone should accommodate, to the maximum extent possible, the needs of users of multiple global scripts. Principles of fair and equitable treatment should be adhered to, avoiding undue consideration for users of particular scripts in a space that is used by all. In addition, both gTLDs and ccTLDs co-exist in the root zone and are relied upon by Internet users around the world. While gTLDs and ccTLDs may involve different operating environments, it is critical that a reliable user experience is produced across the TLD space. As a result, any **label generation rules** for TLDs will need to be adhered to by both ccTLDs and gTLDs. (See section 4 for a discussion of label generation rules.)

[1] The Unicode Standard is a character coding system designed to support the worldwide interchange, processing, and display of the written texts of the diverse languages and technical disciplines of the modern world. See http://unicode.org/standard/standard.html. As of this writing, Unicode is at version 6.0, but a new version (6.1) is contemplated for publication in February of 2012. Various ancillary documents are being prepared for an update to align with Unicode 6.1. Public review and comment are invited on the drafts Issue #208: Proposed Update UTR #36: Unicode Security Considerations (see http://www.unicode.org/review/pri208/) and Issue #209: Proposed Update Unicode Technical Standard #39 Unicode Security Mechanisms (see http://www.unicode.org/review/pri209/).

[2] See http://www.rfc-editor.org/rfc/rfc5890.txt; http://www.rfc-editor.org/rfc/rfc5891.txt, http://www.rfc-editor.org/rfc/rfc5892.txt; http://www.rfc-editor.org/rfc/rfc5893.txt; http://www.rfc-editor.org/rfc/rfc5894.txt, http://www.rfc-editor.org/rfc/rfc5895.txt.

2. In the early stages of developing a variant management mechanism, a cautious approach should be adopted; if necessary, a more liberal approach may be adopted later. A principle of incrementalism appears well-suited to actions in this arena. Given that experience in this area is limited, and actions taken will create precedents and outcomes that cannot be undone, variant TLD labels should be narrowly defined, and restrictive rules for active use of **variant labels** in the DNS should be adopted. Wherever possible, instead of adding a new type of variant TLD label, an alternative approach should be used – for example, using an existing ICANN evaluation or objection process that delivers an appropriate way of **blocking** undesired TLD strings. If there is a process already in existence that delivers a similar result to what is desired, that process should be used rather than establishing a new type of variant label. The goal should be to maximize efforts toward the prevention of future problems, and to minimize active entries in the DNS to those where an explicit need has been established, the user experience implications have been fully studied, and no negative impacts to security or stability have been identified.

3. The root zone is a special case, and the approach taken for variant management in the root need not prescribe that taken by individual TLD registries. ICANN must adopt and maintain a consistent policy for consideration of requests for IDN variant TLDs, and this may entail specific criteria and rules applicable for variant labels to be allowed at the top level. However, formulation of TLD registry policy often takes into account the specific user context, and there may be corresponding reasons for different criteria or rules relating to variant label generation and use, subject to certain minimum requirements necessary for security or stability reasons. Accordingly, TLD registry operators should not have an automatic obligation to abide by all of the same variant tables and policies used at the top level.

4. Users dealing with technology learn very quickly and can accommodate to a wide variety of behaviors, provided that they can build some model of the behavior, and provided that the behavior is consistent and predictable once learned. Recognizing the need for usability of multiple scripts in the DNS, and the desirability of reasonable approximations of natural language usage, it is also assumed that users are not dependent on the ability to use the full natural language without restrictions and will be able to accommodate certain limitations to the full natural language where necessary. To create the consistency required for logical user adaptation, variant management mechanisms should be based on known and predictable rules and procedures, fully available to all users.

1.2 Variants and the Current Environment

"Variant" is a term that has been used in multiple ways, to indicate some sort of relationship between two or more labels or names. It has been used variably to refer to, for example, a particular relationship between specific characters or code points in a particular script, or a set of alternate labels where some linkage relationship is articulated, or a desired procedure whereby names are registered in multiples, or a desired functionality causing shared behavior by some set of identifiers.

In the DNS environment today, there is no accepted definition for what may constitute a variant relationship between top-level labels, nor is there a "variant management" mechanism for the top level, although such has often been proposed as a way to facilitate solutions to a particular problem. Several communities have indicated an urgent need for solutions in this area, to support deployment of the full range of products and services made possible by bringing IDN capabilities to the namespace.

In the discussion of issues, the report reveals a tension between the unmistakable interest in creating greater functionality to address a range of potential variant cases, and the difficulties of using the DNS to meet these objectives. In proposing a clear definition of "variant" and outlining what cases are most appropriate for development of solutions, the report attempts to highlight the cost-benefit analysis that needs to be done before undertaking any implementation.

The notion of variants is extremely compelling. The most aggressive sort of variant proposals require some idea of using what are strictly speaking different names in the DNS as though they are the same, or treating a pair of separate names in a way that accesses separate, but closely related, content. The basic problem is that, owing to the vagaries of the encoding of characters, the rendering of **fonts**, and various other historical facts, strings that do not match one another as far as a computer is concerned are "the same" or "exactly equivalent" to a user of some natural languages. Many discussions of variant issues appear to be based on a commonly-held assumption that making the user experience as intuitive as possible can be easily accomplished via certain adjustments to the operation of the DNS, with the outcome that users will not be surprised.

Regardless of how tempting, however, this assumption conceals three important problems. One is that "users" are not a single group. This issue is discussed in detail later in the report (see Section 5). The other two problems are more fundamental, and must be considered before proceeding.

The DNS is designed as a known-item search system, and it works by exact match (with the single exception of a wildcard label, which matches everything). Many of the proposals commonly referred to as "variants" are attempts, however well-meaning, to make the DNS appear to match any number of different items as though they were the same item. But the DNS is extremely badly adapted to that sort of use. First, while there are ways of **aliasing** one name to another (CNAME and DNAME), they are not reversible and therefore do not capture the sort of relationship that is needed in many cases. The DNS is only loosely coherent in its design, so making names "be the same" by provisioning them equivalently is error-prone, and in any case, is guaranteed sometimes to be inconsistent and therefore surprising. Finally, since the DNS has distributed management, the desired consistent user experience cannot actually be assured (except, perhaps, by very heavy-handed administrative or contractual procedures) because a **zone** lower in the tree might adopt different conventions. This leads to more uncertainty on the part of users, and not less. All of the above suggests that trying to avoid user surprise and achieve the desired user experience using the DNS (or at least the DNS as it stands today) is an ill-suited strategy to achieve the optimal outcomes.

Another problem lies in supposing that, by using aliasing or some sort of parallel provisioning techniques in the DNS, one will have solved all or even a substantial number of the surprises that users will face. Even if it is possible to make two names somehow be the same in the DNS, that sameness is invisible to other applications on the Internet. Those applications often need to know what names they themselves are known by to others.

For example, if an email is sent to localpart@dname.example.com and that name has a DNAME that resolves to realname.example.com, the mail transport agent is not going to rewrite the server-part of the address. It is going to send the mail to the MX for realname.example.com. If the mail exchange server for realname.example.com does not know it has to handle email for dname.example.com, the mail will be rejected (or, worse, just disappear). HTTP servers are similarly ignorant about all the names pointing at them; indeed, if they weren't, modern virtual hosting would be impossible.

Internet protocols can be divided roughly into two types: those that need to know their own name or that can take advantage of knowing it, and those that do not. In FTP, for instance, it (usually) does not matter what domain name you used when you connected: once you have connected and are authenticated, the name of the server you used makes no difference. Many other protocols (including HTTP and SMTP) are designed to be sensitive to the name with which they were contacted; in other cases (such as IMAP), one mode of operation is sensitive to the server's name, and another is not. Whenever a server is sensitive to the name by which it is known, that sensitivity means that the server needs to know all its own names when they start service. Without support for lookup of the aliases of a host (in the DNS such support does not today exist), this operation has to be supported by direct configuration of the server software.

On top of the above, it is worth remembering that a very significant portion of Internet users may not rely on DNS names to find what they are looking for, but instead type whatever they want to find into a search engine. Search engines do not treat sites as being related to one another on the basis of their IP addresses, or most other things obtainable from the DNS. Instead, sites with different http server names are grouped together depending on various http response codes (generally, http redirects). It is an http redirect that causes two otherwise-unrelated DNS names to be treated as “the same” web server. Adding any sort of support to the DNS to link the http servers together will have little or no effect on their relationship to each other in search engine results, at least until the search engines start using such features.

It is conceivable that the DNS could be altered to support a way to look up variants (of whatever type and however they might be supported) for a name. This would permit server software to do some auto-configuration of itself at start up time. At the moment, however, such support does not exist and it is not clear how long it would take to fully understand the issues, including whether this could be done without causing security or operational problems, and then standardize on a representation in the DNS; much less to add such support to server software and deploy it widely enough across the network to make it useful.

This issue is extremely significant to the aimed-for benefit from adding variants: it might be that the cost of adding this support (whatever such support turns out to be) outweighs the benefit in some cases, because of the far greater chance that applications are not configured correctly and the great difficulty in troubleshooting such conditions given the limited tools that are available. This consideration does not mean that all types of variants are impossible, or that trying to have multiple, related, names is wrong in every case. But it does mean that in most cases the costs and risks involved are significant and would require demonstrating a high level of user benefit to undertake development of such functionality via the DNS.

2 Project Overview

Historically, the DNS root zone has been limited to a subset of the characters in the US-ASCII (American Standard Code for Information Interchange) character set. This is changing with the introduction of **Internationalized Domain Names** (IDNs), including the introduction of new top-level domains (TLDs) in multiple scripts, enabling Internet users to access domain names using writing systems familiar to them.

The opening of the IDN country code Top-Level Domain (ccTLD) Fast Track Process[3] by the ICANN Board in October 2009 enabled countries and territories to submit requests to ICANN for a limited number of IDN ccTLDs representing their respective country or territory names in scripts other than US-ASCII characters.

The new generic Top-Level Domain (gTLD) Program[4], approved in June 2011 and opening for applications in January 2012, will allow for the first time the addition of IDN gTLDs into the root zone.

2.1 The Variant Issues Project

IDNs can serve as powerful tools for broadening the Internet's capacity and accessibility; however, for a good user experience they also raise unique issues. One important issue concerns the use of "variants," which, according to one technical definition, occur when a single conceptual character can be identified with two or more different Unicode Code Points.[5] TLDs containing one or more such characters might be considered “variant TLDs,” and unless carefully implemented, might result in user confusion or a poor user experience. While the concept of “variants” is raised in a number of contexts, and in some cases is regarded as critical for the successful adoption of IDN TLDs to meet user needs, there is no single definition or rule for determining whether TLDs can be considered variants of one another.

In an effort to develop potential solutions for the delegation of IDN variant Top-Level Domains (TLDs), the ICANN Board in 2010 passed a resolution directing the development of a preliminary report on the viability, sustainability and delegation of IDN variants.[6]

[3] http://www.icann.org/en/topics/idn/fast-track/

[4] http://newgtlds.icann.org/

[5] http://www.rfc-editor.org/rfc/rfc3743.txt

[6] The Board resolution provided that “The CEO is directed to develop (in consultation with the board ES-WG) an issues report identifying what needs to be done with the evaluation, possible delegation, allocation and operation of gTLDs containing variant characters IDNs as part of the new gTLD process

The IDN Variant Issues Project plan was published in April 2011[7], along with a call for volunteers, and work commenced shortly thereafter. Six script case study teams (Arabic, Chinese, Cyrillic, Devanagari, Greek, and Latin) worked to identify the set of issues that, if resolved, could enable the delegation of IDN variant TLDs for the benefit of the respective user communities. The case study teams comprised a total of 66 experts from 29 countries and territories, and offered expertise in the areas of DNS, IDNA, linguistics, security & scalability, policy, registry/registrar operations, and community representation. The case study team reports were produced on schedule and published for public comment in October 2011.[8]

ICANN, assisted by a coordination team comprised of representatives from the case study teams, has worked to build on these team reports to develop this integrated issues report, to cover both common issues germane across the cases studied and issues particular to specific cases. An integrated analysis of the issues associated with IDN variant TLDs will be an important milestone toward considering subsequent work in this area.

The coordination team provided valuable language and script support in discussion of the issues, and also served as reviewers to the structure and content of this report. Many of the discussions and comments from the coordination team are reflected in the report; however, this integrated issues report is the product of ICANN and as such may not necessarily reflect the views of individual members of the coordination team.

2.2 Objectives of the Integrated Issues Report

As detailed in the project plan, this issues report is intended to describe each of the general and case-specific issues to be resolved for the cases studied.

In accordance with the scope defined for the project, the following objectives have been established for this issues report:

- Identify the set of issues relevant to all the studied scripts.
- Identify any sets of issues relevant to only some of studied scripts.
- Provide a brief analysis of the issues, including the benefits and risks of possible approaches identified.

in order to facilitate the development of workable approaches to the deployment of gTLDs containing variant characters IDNs. The analysis of needed work should identify the appropriate venues (e.g., ICANN, IETF, language community, etc.) for pursuing the necessary work. The report should be published for public review." See http://www.icann.org/en/minutes/resolutions-25sep10-en.htm#2.5

[7] http://www.icann.org/en/announcements/announcement-20apr11-en.htm

[8] http://www.icann.org/en/announcements/announcement-4-03oct11-en.htm

- Identify areas where further study or work could be pursued.

2.3 *Scope of the Integrated Issues Report*

As noted above, this report is designed to provide a review of issues concerning the delegation of IDN variant TLDs. Other related issues are discussed in the report to the extent they are relevant.

It is important to note that there are a number of scripts not represented by the six case studies. In addition, as noted by several of the teams in their reports, some of the scripts studied are used to represent a number of languages, and not all of those languages were represented on the case study teams. Similarly, none of the teams specifically included experts on typography or on human cognition, even though those topics might be important for this subject matter. Accordingly, all possible cases have not been analyzed, and this work is not being represented as comprehensive of all potential issues. However, it is expected that this project's gathering of DNS, operations, and language expertise for these scripts is relevant to a significant percentage of the world's Internet users, and will be an important milestone in the work in this area.

Devising variant rules and proposing variant management solutions are not within the scope of this report. In some instances, a range of possible solutions are considered and analyzed with a view toward informing potential later phases of the project that would be focused on solutions.

3 Range of Possible Variant Cases Identified

One of the central concerns of this report is to identify possible variant issues that have been raised within the six script case studies. In the light of concrete knowledge of the scripts, the case study teams have been able to provide in-depth exploration of the relevant problems presented by each script. These can be understood most effectively if they are brought under a system of classification which draws attention to their similarities, but also to their specific differences.

3.1 Classification of Variants as Discovered

A classification is here proposed to create a framework for understanding of the issues, as well as to suggest common features that may cause a variant case to occur, and which may inform consideration of how such a case could be incorporated in a variant management mechanism.

This classification is fundamentally semiotic: that is to say, it is based on the formal details of the notations employed by the different scripts, and the rules governing their use. The policy justification for interest in these cases lies elsewhere, in the use made by various communities of the scripts, and particular dangers (e.g., of user confusion) that may attend this use. These communities, however, are extremely diverse, not only in their existing capabilities and experiences, but also in the diverse relationships of national and dialectal groups to use of their own scripts. Making a semiotic classification is the only way to be sure of addressing the common properties of all the scripts.

For the purposes of this section, although “variant” has not been defined, a working definition of “exhibiting some relationship such that one of the variants may, under some circumstances, somehow be conflated with another” is used. Although imprecise, it captures the central sense common to all the ways the term appears to be used with respect to DNS names.

Fundamental to the classification are some basic distinctions. First of all is a distinction between **code point variants**, where a single character is in some way closely related to – or likely to be confused with – an alternative, from **whole-string variants**, where the token at issue is longer than a character, and may be a morpheme, a full word or even a phrase, some meaningful element in a language that uses the script.

In Nenets, the letter ӈ EN WITH HOOK (U+04C8) may vary with the digraph нг EN + GHE (U+043D + U+0433): this is a code point variant. In Greek, Πειραιάς and Πειραιεύς refer to the same city, Piraeus: these are whole-string variants.

3.1.1 Code Point Variants

We treat as code point variants those cases where a single character is at variance with a short sequence, such as a digraph or a trigraph. The essential is that the variance is at character level, i.e. within the writing system, rather than in the relations of particular words.

Within the code point variants, it is important to distinguish the exchangeable from the visually similar. An **exchangeable** variant is a case where two or more characters, or code points are seen by a user as so closely related that they may fill the same role in a domain name label. The user, in some or all cases of their use, is indifferent between them, and so the system must recognize that they are – on occasion at least – non-distinct. A **visually-similar** variant is a case where two or more **glyphs** are so much alike that one may be mistaken for the other. The vagaries of font choice, point size and rendering mean that these cases are not entirely foreseeable, but in general it can be identified where caution is required.

It is entirely possible that a variant character may be both exchangeable and visually similar with another character. Since these two types of cases are likely to be supported differently, for these cases, attention must be given to overlap and precedence of policies.

3.1.2 Whole-String Variants

It would also be possible to identify whole-string variants which are purely syntactic, having to do with strings of characters and their order, perhaps re-ordering them or transforming them in some other way, but without bearing on meaningful linguistic elements (for example, Pig Latin → Igpay Atinlay). These would be classed, in principle, as whole-string variants.

Note the case of code points like tónos in Greek, or zero-width non joiner (ZWNJ U+200C), a control character that is used in the spelling of Arabic or Devanagari. These have effects on the rendering of a string as a whole -- either (as with tónos) because there can only be one of them in a word, or (as with ZWNJ) because they might be invisible in their effects on the word's rendering, except in one crucial locus. Nevertheless, they are not classed here as creating whole-string variants. Instead, it is envisaged that such elements would stand in alternation to the absence of those elements: a registered TLD containing such an element would trigger a variant label which would be the same string but without the tónos or ZWNJ.

Linguistic variants make up the most significant part of whole-string variants: this is where the variant relationship would hold between elements of particular languages (e.g., prefixes, suffixes, words or phrases). In principle these could hold within a given language, or as relations between the words of different languages (as translations, or historical etymologies). It will be seen that the main concrete case which has arisen in the case studies concerns linkage of elements of different dialects (or registers) of a single language (specifically Greek) as variants. Hence the particular note of **dialectal** variants within linguistic variants.

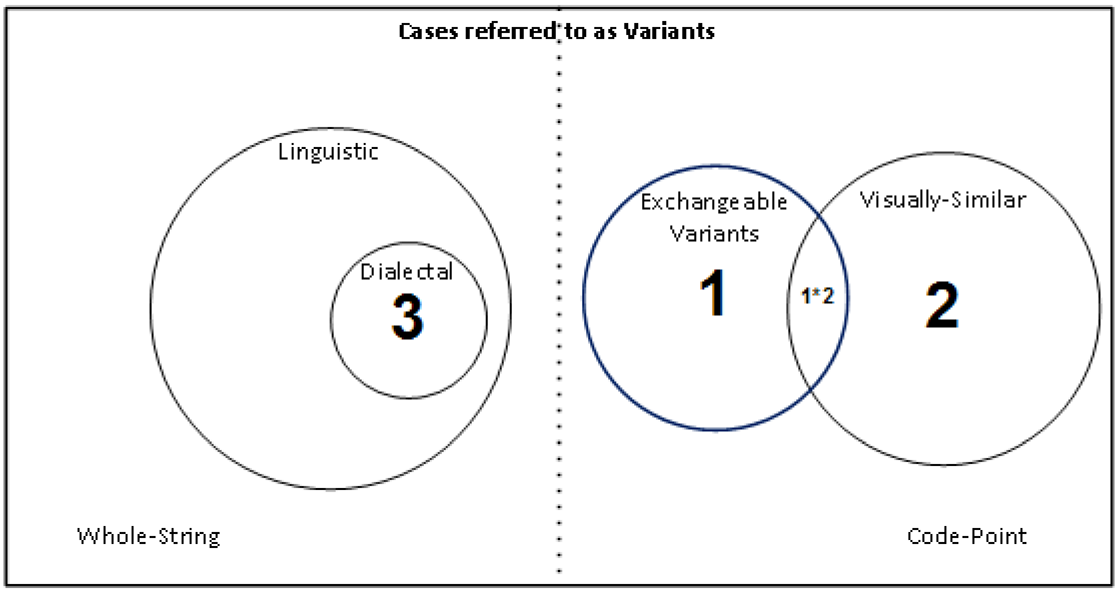

Figure 1: Graphical representation of variant cases identified in the case studies

3.2 *Taxonomy of Identified Variant Cases*

The broad types of cases described above can be further organized into a taxonomy of specified classes.

The key role played by the concept of **abstract characters** in this taxonomy may entail some difficulties of interpretation. The assignment of code points and representative glyphs is definite within Unicode, but the recognition of abstract characters is implicit (Unicode names of characters tend to refer only to superficial graphic features of the glyph). In fact, an abstract character's identity depends on a combination of linguistic analyses of all the languages in question, and gross facts about glyph-class. Essentially, a linguistic test of whether there is one or more abstract character depends on whether there is a discernible contrast within a language; but it also depends on the case of the glyph: a pair of upper- and lower-case glyphs do not represent the same abstract character.

A term which comes closer to the linguistic understanding of "character representing a single phone within a language" is the **conceptual character**. In RFC 3743 (often called "the JET Guidelines"), "variants" are said to be the situation "wherein one conceptual character can be identified with several different Code Points in character sets for computer use."[9] It is not, however, further defined.

The presumption is that, within a language, every abstract character corresponds to a single phoneme or grapheme (or in UniHan a single sememe, i.e., an interpretable

[9] http://www.rfc-editor.org/rfc/rfc3743.txt

meaning, usually with determinate phonic pronunciations). However, although a single character may be consistent within a language, as between languages it may represent very different phonemes, graphemes and sememes. Consider, e.g., “a” in English and Spanish spelling, which are pronounced very differently; standard and swash kaf in Sindhi and Arabic, which marks a linguistic contrast in one language but is stylistic in the other; and as a representative Chinese character, 人 (U+4EBA) has a basic meaning ‘human being’, but in Chinese is pronounced rén and in Japanese jin, nin, hito etc.; each language then uses it with different (but overlapping) combinatory properties.

Nevertheless, the Unicode doctrine is that Abstract Characters are what is encoded. “When an abstract character is mapped or assigned to a particular code point in the codespace, it is then referred to as an encoded character.”[10] And in practice, “abstract character” is a concept that is needed and is reasonably clear within a given language, giving a sense of an underlying unity in cases where code points seem to have been multiplied beyond necessity, in the process of defining Unicode.

The classes described are identified using the cases of 1, 2, and 3 based on Figure 1 above, and references used in the examples are to the corresponding sections in that team’s case study report.

[10] See http://www.unicode.org/versions/Unicode6.0.0/ch02.pdf.

Arabic	Chinese	Cyrillic	Devanagari	Greek	Latin
1. Exchangeable but not Visually Similar					
1(a). Compatibility mappings					
		Rarely used character/sequence of characters, e.g., EN WITH HOOK (U+04C7 / U+04C8); EN + GHE (U+043D U+0433) (3.5); C WITH ACUTE (U+0107) / C + J in Montenegrin (U+0441 U+0301) (9.1)	Homophonous spellings (4.2)		(Discussed for Swedish and German umlaut in 6.3)
1(b). Join-control characters					
Discussion of ZWNJ issues (5.21)			EYELASH RA in Nepali represents a different phoneme from RA, and is currently only rep'd by combination of RA with VIRAMA and ZWJ (U+0930 U+094D U+200D) (4.3.1). Correct spelling of inflected forms in Nepali requires use of ZWNJ (4.3.2)		
1(c). Upper/lower case and underspecified information					

Arabic	Chinese	Cyrillic	Devanagari	Greek	Latin
Optional diacritics (notably vowels) (6.4)	Instances where one SC corresponds to one or the other of many TC, often context dependent (2.1 final para.)	IE/IO in Russian (3.1); SMALL LETTER I WITH GRAVE (3.3);		TONOS may be absent (11)	Case folding (6.3)
1*2. Exchangeable and Visually Similar					
1*2(a). Same abstract character with more than one encoding					
Identical: decomposables, esp. if not normalized (6.1) 6.1 Appendix A.2 (A.2.1 – A.2.2)	Non-identical: Generic variant characters (5)	Identical: decomposables, esp. if not normalized (3.8)			SMALLER TURNED E (U+01DD), SMALL LETTER SCHWA (U+0259) (6.2) (Discussed at 6.5: Precomposed characters) (Discussed at 6.4: decorative and contrastive variants)
1*2(b). Same abstract character differently rendered in some contexts					
KAF Group, HEH Group, and YEH Group (6.1.abc), MARBUTA Group (6.1g) HEH with				SIGMA and FINAL SIGMA (5, 12)	

Arabic	**Chinese**	**Cyrillic**	**Devanagari**	**Greek**	**Latin**
HAMZA Group (6.1h) NOON Group (6.1j); KAF Group, YEH Group, YEH with HAMZA Group; variants of dot orientation (all in 6.2) 6.1 a thru k – Appendix A.1					
1*2(c). Different abstract characters, but exchangeable for users					
6.3 a thru d vowels with/without HAMZA (U+0621), All forms of ALEF (e.g., U+0627) with/without composed form (6.3ab); TEH MARBUTA (U+0629) and HEH (U+0647) (6.3c); Arabic-Indic and extended Arabic-Indic digits (6.3d)	Simplified characters (SC) / Traditional characters (TC) (5)	GHE/GHE WITH UPTURN (U+0433, U+0491) in Ukrainian (3.2);			
1*2(d). Improper characters					
		APOSTROPHE (U+02BC)	MODIFIER LETTER APOSTROPHE (U+02BC)		(Discussed at 6.7: Punctuation: Latin

Arabic	Chinese	Cyrillic	Devanagari	Greek	Latin
		in Ukrainian (3.6)	/ zero in Boro/Dogri/ Maithili languages (3.4)		characters incorrectly substituted for APOSTROPHE)
2. Visually Similar, but not Exchangeable					
2(a). Simple Visual Similarity					
Non-identical: GHAIN/FEH Group, QAF/AIN with 2 DOTS ABOVE Group, AIN/FEH/QAF with 3 DOTS ABOVE Group (6.2) 6.2 Appendix B2 and B3		Old Letters, e.g., YAT/SEMISOFT SIGN (U+048C / U+048D) (3.4); ZE (U+0437) / DIGIT THREE (U+0033) (3.7); various forms of GHE, KA and EN (e.g., U+0433, 0491, 0493, U+043A,049B, 049D, 04A3, 04A5) (3.9); Appx I passim	GHA (U+0918) / DHA (U+0927); BHA (U+092D) / MA (U+092E) etc (3.2.1 + Appx III); composite characters DGA/DNA/DRA etc. (3.2.2 + Appx IV); Potential mistaken characters due to rendering errors, environmental restrictions (3.3.1)		(Discussed at 6.6: combining marks) (Discussed at 6.7: Punctuation: Latin characters confusable with APOSTROPHE)
2(b). Inter-script					
		Especially of concern for WHOLE-SCRIPT CONFUSABLE strings with Greek and Latin (4)	Especially of concern for WHOLE-SCRIPT CONFUSABLE strings with other Brahmi scripts, e.g. Gujarati (3.5, 4.1)	Especially of concern for WHOLE-SCRIPT CONFUSABLE strings with Cyrillic and Latin	(Discussed at 6.1, 7 and 8.) Especially of concern for WHOLE-SCRIPT CONFUSABLE strings with Greek and Cyrillic

Arabic	Chinese	Cyrillic	Devanagari	Greek	Latin
3. Linguistic variants					
				Equivalence between corresponding words in Dimotiki and Katharevousa dialects (13)	

Table 1: Range of Variant Cases Identified in Case Study Team Reports

3.3 *Discussion of Variant Classes*

Unicode typically assigns a unique code point to what it recognizes as an "abstract character" (i.e., a unit of information used for the organization, control, or representation of textual data). (In practice, the glyph is whatever is rendered, being sensitive to fonts, resolution, or other issues.) This classification considers cases where this one-to-one correspondence fails for some reason.

Class 1. Exchangeable but not Visually Similar

<u>Class 1(a). Compatibility mappings</u>

By this is meant cases where, within a particular language's spelling system, a digraph or trigraph (sequence of two or three glyphs) is taken as a conventional equivalent for another (usually rather rare and distinctive) glyph. The compatibility referred to here is backwards-compatibility, as when a new (more exact) representation is explicitly kept equivalent to an old make-do representation.

An example would be the homorganic syllable-final nasal (anusvara) in Hindi representing a nasal consonant before a following syllable-initial consonant (e.g., writing ambā, hindī gaŋgā as "ãbā, hĩdī, gãgā," as is normal in Devanagari). These are the "homophonous spellings."

If any of these equivalences are directly represented in the root, the effect will be to impose this equivalence on all languages which use the given script, because a single rule for processing them all will be necessary. See section 4 for a discussion of this issue.

<u>Class 1(b). Join-control characters</u>

There are certain code points (notably ZERO WIDTH NON-JOINER, U+200C, "ZWNJ"; and ZERO WIDTH JOINER, U+200D, "ZWJ") which, under the rules of the IDNA Protocol, may only be inserted in positions in a label that should trigger a particular effect on rendering.

For Arabic, this is instantiated at length in 5.21 of the Case Study.

In Nepali, the "eyelash ra," a separate character from "ra," but not represented by a separate code point is rendered through the insertion of VIRAMA (U+094D) and ZWJ (U+200D). Showing the presence of a morpheme-boundary in a noun may be rendered through the insertion of ZWNJ (U+200C).

Class 1(c). Upper/lower case and underspecified information

While DNS labels are comprised of octets and can hold any octet desired, they are defined in RFCs 1034[11] and 1035[12] such that they treat ASCII in a special way. While the DNS preserves ASCII case differences, the difference is not considered in the matching rules. For instance, the label "example" in the DNS is supposed to be preserved just as entered in the authoritative zone file during transmission and caching, and also the label "ExAmple" is also supposed to be so preserved. These two labels, however, could not both be in the same zone for the same resource record, because for the purposes of matching they are the same.

The IDNA Protocol uses a different rule: a code point must be stable under both Normalization Form KC ("NFKC")[13] and case folding operations to be a **valid code point** in the protocol (see RFC 5892[14]). Therefore, upper-case characters in scripts that use them (Latin, Greek, and Cyrillic) are not allowed in **U-labels** – even if the character would have the difference preserved in a traditional LDH label (to use an artificial example from above, the string "ExAmplé" is not a valid U-label, because of the two upper-case characters E and A. This means that a distinction is present in usual Latin/Greek/Cyrillic text is routinely neutralized in U-labels and hence IDN TLDs.

Class 1*2. Exchangeable and Visually Similar

Class 1*2(a). Same abstract character with more than one encoding

A clear example that many composite characters (e.g., vowels and consonant signs that carry diacritic marks, such as <é>, <ā>, < ç>) have more than one representation as code points, namely, as pre-composed wholes, and as the simple letter with a separate combining diacritic. This occurs in Arabic, Cyrillic and Latin. Many of these cases (including some that were raised in the case study reports) do not bear on IDNs, because they do not qualify as valid U-labels according to the IDNA standard. Where

[11] http://www.rfc-editor.org/rfc/rfc1034.txt

[12] http://www.rfc-editor.org/rfc/rfc1035.txt

[13] Normalization is a mechanism for making different strings – in this case Unicode strings – match one another under the right circumstances. A complete outline of Unicode normalization is beyond the scope of this document, but it is a critical part of the way IDNA operates. A large number of potential problems in IDNA are solved because of the normalization rules. To understand Unicode normalization, see "UNICODE NORMALIZATION FORMS," UAX15, http://www.unicode.org/reports/tr15/. For details of how Unicode normalization interacts with IDNA and U-labels, see RFC 5892, especially section 2.2 (http://www.rfc-editor.org/rfc/rfc5892.txt) and RFC 5890 (http://www.rfc-editor.org/rfc/rfc5890.txt), especially the definition of U-label.

[14] http://tools.rfc-editor.org/html/rfc5892

there are different standard ways of encoding the same abstract character as precomposed and decomposed forms, IDNA will allow only one form, and thus user interfaces and preprocessors are free to map all other forms into the preferred form. For example, the letter "o" with a circumflex accent can be written in more than one way: by using LATIN SMALL LETTER O, U+600F and COMBINING CIRCUMFLEX ACCENT, U+0302; or by using LATIN SMALL LETTER O WITH CIRCUMFLEX, U+00F4. After normalization, the string is the same code points (in this case, the latter, precomposed form). There are nevertheless some cases where normalization does not completely cover the range of potential inputs. Extensive examples are in the Arabic case study report, Appendix A.2.1 wherever it says "not defined;" the failure of normalization also contributes to the example of Eyelash Ra outlined in the Devanagari report, section 4.3.1. See the discussion of ZWJ as well.

A rarer case is where the glyphs assigned to distinct code points turn out to be indiscernible. It is unclear in these cases whether there is more than one abstract character involved: in any case, the variants do not contrast with one another, and it is a matter of indifference, in ordinary use, which one is intended. There is, however, a security danger comes from code points which are distinct, but invisibly so, because two strings that differ only in the substitution of the invisibly-different code point are completely indistinguishable to a human (competent with the script in question) but are nevertheless different labels and so lead to different DNS names.

This is the case of Chinese generic variant characters (often resulting from the unification of sources to create UniHan), but also of the two ways of expressing 'turned e' or 'schwa' in Latin script (U+01DD and U+0259).

The decorated characters in Latin provide many examples of distinct (sets of) code points which represent the same characters, although the Latin report (with one exception) recommends banning them all from use in TLD labels.

Class 1*2(b). Same abstract character differently rendered in some contexts

These are like the cases in 1*2(a), but differ in that the contrasting code points only have differing glyphs in some contexts. "Context" here can mean either a linguistic context (the glyph's position next to others in a string), or else regional context, where the way that users treat the writing system differs between languages. For use of such code points in a TLD label, however, a common set of rules on such variant characters must be maintained.

This case frequently occurs in the languages written in Arabic script, where the different glyphs may represent what are historically different ways of writing a single given character (but have been given varying code points in Unicode). For example, the letter

YEH is represented with U+064A in Arabic, but U+06CC in Iranian languages such as Persian, causing a different incidence of the letter’s distinctive two dots.

Conversely, different forms (glyphs) of an Arabic character are displayed depending on where they appear in the word (initial, medial, final and isolated), but all are generated by renderings of the same character (i.e., using a single code point). Given that the separate codings for the positional variants (U+FB50-FBFF), which were once available, are disallowed by the IDNA 2008 protocol, it will be impossible to make them variant characters, since they have no identity as separate characters.

In Greek, the positional form of lower-case s (sigma) occurring finally in a string has been assigned a separate code point [<ς> (U+03C2) not <σ> (U+03C3)]. Nonetheless, it clearly represents the same conceptual character, and in upper case <Σ> shows no variation.)

Class 1*2(c). Different abstract characters, but exchangeable for users

The classic case is the Chinese equivalence between Simplified and Traditional characters. These are defined as quite separate code points and hence (prima facie) separate characters. Yet in terms of their contribution to writing particular words, there is no difference between the members of a corresponding Simplified-Traditional pair of characters. The only way to make them contrast is to talk about the actual form or identity of the characters themselves, rather than what they mean or how they sound.

In Arabic script, many combinations of alif and hamza with vowels are assigned their own separate code points: they are distinct from those vowels without alif and hamza, but the conventions of the Arabic language (and some other languages’) spellings allow the forms with and without alif/hamza to be equivalent. In other languages that use Arabic script, this equivalence is not present.

Class 1*2(d). Improper characters

There is a requirement expressed across languages for characters which are in a block of code points distinct from those reserved for the relevant scripts: e.g., the apostrophe character used in Cyrillic for Ukrainian and Belarusian (U+02BC); the apostrophe character used in Devanagari for Boro, Dogri and Maithili (likewise U+02BC).

The presence or absence of such characters in the spelling of specific names and words is not straightforward. Because of the characters' status outside the regular code block for the script, the prospects for their routine use are unclear. If both presence and absence of these characters are ruled as acceptable spellings in TLD labels, names and words where these characters are used will become thereby open to use as variant labels.

A further problem is that many of these characters (e.g., the various “apostrophes” – U+0027 APOSTROPHE, U+02BC MODIFIER LETTER APOSTROPHE, many others in the codeblock 02A0-036F, as well as U+A78B LATIN CAPITAL LETTER SALTILLO and U+A78C LATIN SMALL LETTER SALTILLO etc.) all have very similar small glyphs, and are visually confusable. Some of these characters, such as U+0027 APOSTROPHE, are not permitted at all in U-labels.

Class 2. Visually Similar, but not Exchangeable

Class 2(a). Simple Visual Similarity

These cases are self-explanatory. Glyphs are not sufficiently distinct to represent reliably their associated code points (nor abstract characters).

For discussion of their treatment within a variant framework, see §3.4.

Class 2(b). Inter-script

If labels are to be restricted to a single script, the only case of this is the whole-script confusable case. It is expected that there will be mechanisms to detect these, and block them where necessary. There is a bigger issue to consider if labels are to be constructed out of sets of code points not restricted to those sharing a single script property; see the discussion in section 4.1 for some alternatives.

For discussion of their treatment within a variant framework, see §3.5.

Class 3. Linguistic variants

This raises a different set of issues from the Character Substitutability cases just considered. Potential variants are whole strings (usually words) rather than single characters within a string.

For discussion of their treatment within a variant framework, see §3.6.

3.4 *Visual Similarity Cases*

The cases described in classes 2 and possibly 1*2 above concern visual similarity of characters. Issues concerning these cases are considered in this section.

3.4.1 Treatment of Visual Similarity Cases

As discussed in section 3.2, there are certain cases where the *only* factor creating a possible variant relationship is that a character is visually confusable[15] with another.

No unified user expectation on how to handle such cases emerges from the case study reports. Some highlight the need to delegate a visually confusable variant label along with the applied-for label; others advocate that such variant labels should only be blocked.

Since visual similarity is not only limited to variants, in the new gTLD Applicant Guide Book, the string similarity review process is established with the objective to prevent user confusion and loss of confidence in the DNS resulting from delegation of many similar strings. Note, according to AGB, “similar” refers to strings (labels) so similar that they create a probability of user confusion if more than one of the strings is delegated into the root zone.

To consider how to treat cases of pure visual similarity, it is useful to recall the assumption stated in section 1.3:

> *Given that experience in this area is limited, and actions taken will create precedents and outcomes that cannot be undone, variant TLD labels should be narrowly defined, and restrictive rules for active use of variant labels in the DNS should be adopted. Wherever possible, instead of adding a new type of variant TLD label, an alternative approach should be used – for example, using an existing ICANN evaluation or objection process that delivers an appropriate way of blocking undesired TLD strings. If there is a process already in existence that delivers a similar result to what is desired, that process should be used rather than establishing a new type of variant label.*

Building on this assumption, it follows that variant cases where the only determining factor is visual confusability with another label should be subject to blocking, at least in the initial stages. The existing visual similarity processes should be able to address these cases. It is noted that a desired outcome, as expressed by different communities, is predictability in the handling of visual similarity assessments. It seems that for some script communities, it is possible to produce a table that would be able to generate these visually similar variants deterministically. Thus, existing visual similarity processes could potentially be improved by incorporating these tables from the script communities. It may also be possible to modify or extend the zone generation rules

[15] In this context **visually confusable**, refers to two different strings of Unicode characters whose appearance in common fonts in small sizes at typical screen resolutions is sufficiently close that people easily mistake one for the other.

suggested by this report to support determinations of visual similarity; see section 4.1 for further discussion.

3.4.2 Cross-Script Visual Similarity

The cases discussed in Class 2(b) above concern the similarity of characters across scripts. Cross-script visual similarity is possible, especially among Cyrillic, Greek and Latin scripts (e.g., *peace* (Cyrillic: U+440, U+435, U+430, U+441, U+0435) and *peace* (Latin: U+0070, U+0065, U+0061, U+0063, U+0065) but also as between Devanagari and closely related Brahmi scripts (e.g. Bangla, Gurmukhi). It is possible for two candidate U-labels to appear to be the same in different scripts. Even excluding Unicode properties like Common and Inherited, it is possible to create two IDNA-valid strings in two different scripts where competent users of each script are likely to regard the two strings as the same one. This issue is highlighted in some team reports as "whole-script confusables," which is also what a similar phenomenon is called by Unicode.

As discussed in section 3.4.2, this can be treated as a straightforward matter of string similarity. Note that, because this is an issue of all the component code points being confusable with one another, it is distinguished from the whole string issues discussed in section 3.5 below.

3.4.3 Terminology concerning Visual Similarity

This section focuses on issues related to the acceptability and use of Unicode characters for TLDs; hence we do not seek to adopt terms that are more generally relevant to the relation between symbols and meanings.

One such term is **homograph**, which means an "entity sharing a written form with another entity." This term is ambiguous as to the level of the entity, referring to distinct characters which happen to have a similar written form just as much as it refers to distinct words written in the same way (e.g., *lead*, *rose* in English).

If so used, homograph makes a false analogy between the relation of glyph to the character it represents and the relation of a character-string to the label, name or word it may represent. It is to be avoided for that reason.

Homograph is, in fact, usually applied at the word level. The term has a well-established definition as being a relation of words within a language. In this use, it does not extend across language/script boundaries.

By contrast with *homograph*, a **homoglyph** refers only to the situation where multiple abstract or conceptual characters (and usually their multiple associated code points) are represented with the same glyph. This term is independent of language and/or script. It can be applied to similar glyphs within the same language or script. However, it can also be applied to two elements that cause confusion between languages or scripts. Hence, the glyphs with the shapes "a, e, c, p, x" (and in some italic fonts "*g, u, m, n*" too) are homoglyphs, as between Latin and Cyrillic.

The value of using the term *homoglyph* within the variant discussion is dubious. It focuses on the possible multiplicity of interpretations of a single glyph (or set of similar glyphs) without considering the actual danger of confusion this causes, or the degree to which users are indifferent to the difference between the interpretations, leading them to exchange them freely.

For this reason, the report uses the terminology "visual similarity," which is not specifically restricted as to length of string of glyphs (one or many), or degree of similarity between glyphs in view (absolute or approximate).

3.5 Whole-String Issues

The cases described in Class 3 above concern a variant relationship between two or more whole strings of characters. As opposed to code-point level variants (in which the locus of difference lies in individual code points within a string), whole-string level variants would set whole strings (including potential DNS labels) in contrast. This section considers cases of this kind and the likely impacts of their implementation in the root of the DNS.

From the types of whole-string variants, the linguistic variants subtype would seem interesting to some users given that it appeals to some part of the rule-set of a language, whether its orthography, its phonology, its grammar or its lexicon.

However, linguistic variants present a difficulty, in that the rules that define them cannot be represented as well-defined automata, since generalizations within and between languages are subject to arbitrary exceptions, as well as being hard to define formally, at any level of abstractness.

Examples of what might be considered linguistic variants include:

1. Transcriptions (e.g. Ευαγγελία (Greek) and *Evangelia*; 网络 (Simplified Chinese) and *wang luo* (Pinyin transcription), which means *network* in English), which involve constructive use of another language's phonology and orthography to reconstruct the effect of an expression's pronunciation in a given language.

2. Homophonic strings within a language's spelling system (e.g., in English *too* and *two*); this will include different spellings of the same word (lexeme) in the same

language and script (e.g., in English: *color* and *colour*); in the extreme case, this may extend to strings representing the same word in the scripts of a multi-script language (e.g., in Japanese: ほんだ (Hiragana), ホンダ (Katakana), 本田 (Kanji), *Honda* (Latin), in Devanagari: हिन्दी हिंदी (homorganic nasal) or गरदन गर्दन (variant generated because of loan word which has been assimilated into the language).

3. Corresponding terms in different dialects of the same language, e.g., Ευάγγελος and Βαγγέλης, which are accepted forms of what was (historically) the same name (in Katharevousa and Dimotiki respectively); *diaper* (US English) and *nappy* (UK/AU English).
4. Synonyms within a language (e.g., in English: *buy* and *purchase*).
5. Different identifying expressions within a language (often dependent for their interpretation on conventions, or other shared information): these may be personal (e.g. *Elizabeth II, The Queen, the present queen, the Queen of England, the Head of the Commonwealth* etc.), geographic (e.g., in English: *New York City*, *NYC*, *New York New York*, *the Big Apple* etc.), or in any other field.
6. Translated strings and inter-linguistic correspondences: representations of the same concept/notion in two languages (e.g., *green* (English) and *vert* (French)).

In this breakdown, it becomes clear that linguistic variant rules are never exhaustively and formally predictable, being dependent on equivalences made at various levels (from orthography through grammar and meaning to irony) by users of a given language.

Given the analysis of linguistic variants above, it is clear that some people might want (some of) these cases to be considered variant labels. Writing system and spelling reforms have also had implications for languages that have at times been written in Cyrillic and sometimes also either Arabic or Latin script. Even English has undergone significant spelling reforms, such that some words can be spelled more than one way (e.g., *color* and *colour*) as described above.

One of these types of cases, "Corresponding terms in different dialects of the same language," (3), was recommended for consideration by the Greek case study report, such that Katharevousa and Dimotiki strings should be treated as variants to one another: the registration of one such string would cause the corresponding string in the other dialect to be blocked.

The Greek case study team report noted:

> *Alternative, same[-]meaning words, if the domain consists of words[,] are to be disallowed. This rule is especially put in place to deal with words in*

> *“Katharevousa” and “Dimotiki.” Dimotiki is the contemporary Greek language, where Katharevousa was in use before 1976. However, many word types from Katharevousa are still in use in Dimotiki and the user should be protected of confusability issues between these same meaning words. For example “Πειραιάς” and “Πειραιεύς” are two names of the same city in Greece, respectively in Dimotiki and Katharevousa – The applicant will only choose to register one of these and the other one should be excluded of registration. The team recognizes the difficulties this procedure presents for automated systems, however, since TLD registrations are not expected to be of great volume this rule could be implemented with relatively low cost for ICANN. (p.14)*

The first sentence of this text suggests that the Greek Case Study group have not carefully distinguished the requested inter-dialectal equivalence of words from a more general synonymy (Cf. 3 vs. 4 in the list above), which would be highly subjective in detailed application. Since the recommendation is only for the inter-dialectal equivalence, this may not matter. However, even if check-lists of all the relevant words and names are available, the relationship may be less neat than is assumed.

Although Katharevousa and Dimotiki are largely related formally, this is overall a complex and sometimes unruly linguistic relationship.

For example there is a general principle (among many other equivalence rules) that ancient (Katharevousa) names ending in -εύς have a modern (Dimotiki) form ending in –εάς or –ιάς: e.g. Ατρεύς Ατρέας: βασιλεύς βασιλιάς: Ζεύς Δίας: Ιδομενεύς Ιδομενέας: Πειραιεύς Πειραιάς etc. The list is long, and not clearly finite. But at the same time the equivalence, in the most salient cases at least, is not exact, but highly context- and referent-dependent: so Ατρεύς not Ατρέας is the asteroid Atreus, whereas either can refer to the mythical hero of that name. On the other hand, Ζεύς and Δίας can both refer to the god Zeus, but only Δίας applies to the planet Jupiter. If the Dimotiki and Katharevousa forms were to be judged equivalent in TLDs, implementers would find it necessary to refer to a specific fixed list of names (perhaps even excluding Ατρεύς and Ατρέας, Ζεύς and Δίας). This might have the effect that if new names in -εύς or –εάς or –ιάς were invented, they would not automatically get a variant. On the other hand, native-speakers of Greek would largely be able to predict a possible Katharevousa form, when presented with a new Dimotiki example ending in –εάς or –ιάς.

A further problem stems from the need to agree a specific fixed list, not only of names, but also of every word in the Greek language (onomasticon), and specifically one, which specified the Dimotiki and Katharevousa equivalent for each headword (lexicon). Besides the need to maintain both of these, there is the further complication that Greek is a highly inflected language, and policies would need to be defined on which precise parts of each word’s alternations would be recognized.

The aforementioned paragraph from the Greek team report recommends an outcome which would "exclude from registration" the second variant string, i.e., blocking it. The gTLD Applicant Guidebook defines two processes that could address this issue in a similar way. The String Similarity review described in Module 2[16] compares any two applied-for gTLD strings for visual similarity. If the strings were found to be confusingly similar, they would be placed in contention sets for later evaluation and resolution. Additionally, the review compares each applied-for gTLD string against existing TLDs. If an applied-for gTLD string failed the String Similarity review due to similarity to an existing TLD it would not pass the Initial Evaluation.

Secondly, Module 3 describes the public objection process in which string confusion is one possible ground for a formal objection to be filed to a gTLD application. String confusion in this context is not limited to visual similarity of characters; but may be based on any type of similarity (including visual, aural, or similarity of meaning). The use of any of these two processes could yield the outcome requested by the Greek case study report for the Dimotiki/Katharevousa issue.

As noted above, linguistic variants are not amenable to algorithmic treatment, because the linguistic principles that cause them do not usually exhibit complete regularity. In order to produce a general solution to any general case of linguistic variants (even in a single language), it would be necessary to generate (in advance) a dictionary or set of dictionaries that would govern the handling of submitted strings; this would require the engagement of relevant expertise for each language with linguistic variants. Presumably, the dictionaries would need to be reconciled with one another, to ensure that any possible conflicts would be dealt with. Since there is no requirement that DNS labels actually be words, submitted candidate labels would need to be checked to see whether they matched anything in the relevant dictionary or dictionaries, probably including checking for substrings in the dictionaries and substrings in the submitted candidate label. In the event a match was detected, then zone policy would apply as to what to do with the resulting matching label (e.g. to Block it, Reserve it, or Activate it).

It would be necessary to develop a policy for maintaining the dictionaries. It is not clear how ICANN could make effective exclusion rules permitting one type of linguistic relationship to be treated as a variant case, but another not to be. It is also not clear that the system would be subject to practical automation, given the very large number of potential combinations and the likely requirement to perform substring matching. It would be necessary to formulate a policy that could accommodate future changes to dictionaries, and to formulate a policy that accommodated such changes as to create a conflict between then-existing root zone entries. Additionally, it seems it would be

[16] See http://www.icann.org/en/topics/new-gtlds/rfp-clean-19sep11-en.pdf

necessary to have a dispute resolution procedure to handle cases where different applicants' desires went unmet.

Placing any such variants in the root would create indeterminacy in the system, and uncertainty for users. If a user knows that linguistic variants (with all their intrinsic unpredictability) are part of the system, it spreads the sense that what is typed (or seen on the screen) is not necessarily what others get. This will cause difficulties for users. And it may create even greater difficulty than the convenience that others would derive in having a facility of linguistic variants. Such a facility will inevitably be limited, and we do not yet know how its limits could be set.

Ultimately, a decision would have to be made as to whether these cases are better implemented as variants in the root, or whether other existing policies, such as the String Similarity review or the Public Objection process (described in Modules 2 and 3, respectively, of the new gTLD Applicant Guidebook) would be a better way to accommodate this need.

3.6 Synopsis of Issues

The case studies have enabled a common classification of variant types, subject to updates from findings concerning other writing systems not covered by the case studies. This adds specificity and clarity to considering the variety of potential variant cases and how significant they are to particular writing systems. To create a variant management mechanism, the first issue to be solved is how to define and delineate the variant cases that may exist. That is, to create a set of rules for addressing various cases, it must be possible to constrain the space such that the problems to be solved are known.

Many of the issues highlighted here are complex, and it is clear that the cases in the script studies vary widely. The classification exercise undertaken here demonstrates that there is not a single "variant problem" to which a solution may be developed, but an array of unique characteristics and considerations to be kept in mind in further discussions.

4 Discussion of Issues: Establishing Variant Labels

Every DNS zone has, either implicitly or explicitly, a set of rules that governs what labels are permitted in the zone. This often goes by the general rubric “registry policy.” Some of these policies are related to various conditions that must be met before a registration is permitted. Some of the policies are rules that govern labels: whether they are permitted, and what effects they might have on other candidate labels for the zone. We call this set of rules the **label generation rules**.

The label generation rules for the operation of an IDNA-enabled[17] zone must, at the very least, include the rules constraining which code points are permitted in a U-label in that zone. (Even “everything permitted by IDNA2008” is such a rule; it is not possible to permit U-labels without implicitly adopting some rule.) We call this list the **code point repertoire for the zone** and, more informally, the **zone repertoire**. (It is important not to use “repertoire” on its own in order to avoid confusion with the meaning of that bare word in the Unicode standards.) In addition, whenever there are variant relationships among code points, the label generation rules must include the specification of which code points may be used as alternatives for other code points (the **alternative code points**), the status of the label that results from the alternation of the code point in question (the **code point variant status**), and any degrees of freedom with respect to the status values.[18] We call the set of these last three items the **code point variant rules**. The alternative code points may need a way to specify “substitute by removal”: code points such as ZERO WIDTH NON-JOINER (U+200C) and MODIFIER LETTER APOSTROPHE (U+02BC) seem at least sometimes to need a variant label that does not contain those code points, in order to permit practical entry under some current user input methods.

The root is a shared resource, and that means that, all else being equal, ICANN should not privilege one group of language users over another in the administration of the root zone. Accommodating every desire that everyone might have is a practical impossibility, and even ensuring equitable treatment is complicated. However, it is necessary to have an agreed and accepted mechanism for defining zone repertoires and code point variant rules for the root that could take into account the needs of every language community that can be supported by IDNA.

[17] In principle, this is true of non-IDNA zones, too, but as those cases are not directly relevant to the present concern, we will ignore them for the purposes of this discussion.

[18] There may be more than one way to represent the label generation rules. For instance, RFC 3743 defines the repertoire as the list of Valid Code Points. For a given code point, the alternatives are all listed in the Preferred Variant and Character Variant columns of the Language Variant Table. Those code points in the Preferred Variant column are the ones that normally result in a resulting label being activated (which means there is less freedom about the status). Those code points in the Character Variant column are ones that normally result in a resulting label *not* being activated. RFC 3743 does not provide a convenient mechanism to Block a label.

ICANN must have a way to validate potential IDN variant TLD labels when submitted, and to validate all IDN TLDs requested for variant labels and variant conflicts. By the same token, because the root is a single, shared zone, it is necessary to adopt a single, internally consistent set of label generation rules that governs the operation of this single zone. It is worth observing that, while IDNA2008 determines whether a code point is permitted merely by examining Unicode properties (and not by considering code points one at a time), the protocol was designed with the explicit assumption[19] that registries would add additional restrictions to the protocol for the purposes of registration. The most natural restriction is to refuse to register code points except those needed to write a supported language. But unlike many other zones, the root might need to support any language; that need may entail careful linguistic study of a large portion of the PVALID code points in IDNA2008.

Identification of an appropriate authority for the code point repertoire for the root zone is a difficult undertaking. To the maximum extent possible, the relevant language communities need to agree on a shared set of code points for the zone repertoire. It is not clear on what authority ICANN would rest its claim to being able to decide among different candidate zone repertoires in the event of a dispute. If, on the other hand, no dispute exists, then it does not seem that there would be different candidates for the zone repertoire.

4.1 Establishing the Label Generation Rules

A detailed proposal for the label generation rules is beyond the scope of the present document. Such a proposal would properly belong in a proposal for implementation. This section outlines the parameters by which the rules might be determined.

There are three independent factors that need to be considered to build a process to generate label generation rules. These can be thought of as three parameters to be set in the process.

The first parameter is that of relative comprehensiveness (henceforth, "comprehensiveness"). This parameter establishes the degree to which the whole of Unicode (or some relevant Unicode subset) needs to be considered. The maximal setting of this parameter would require that the entire Unicode code space be considered in preparing the label generation rules; the minimal setting of this parameter would require that a rule be developed for any code point actually requested in any U-label in the root zone, but that other code points might be ignored. The maximal setting has the conspicuous advantage that it likely makes for the most stable rules: because the only additions to the zone repertoire could ever come from Unicode additions. The disadvantage is that Unicode encodes some characters that are extremely obscure, and so it might be difficult if not impossible to find relevant experts. Also, considerable work would be undertaken that might never prove useful (it seems very unlikely that anyone

[19] RFC 5891, section 4.3

will want a TLD spelled using Linear B, for example). A slightly more relaxed approach would be to allow some **script tables** to be passed over. The minimal setting has the advantage that effort is not wasted on code points nobody wants. A middle-range approach might generate rules for those code points that seem likely candidates for requests.

The second parameter is that of expert authority involvement (henceforth, "expert"). The maximal setting would require expert development of the rules for the set of code points in question, and by a fairly large body of experts with relevant knowledge of IDNA and DNS, the scripts and languages in question, writing systems, or all of these. The minimal setting would permit anyone to establish any set of rules they liked (with the only constraint being that conflicts would not be permitted, and would be resolved in favor of the most cautious action). The maximal setting has the advantage that the rules are more likely to be stable, because the expert study is likely to expose important issues and set rules to deal with them. However, it is not clear how ICANN might determine expertise, and a situation where someone was dissatisfied with a given result could result in disputes about the legitimacy of one or more experts. In addition, many of the Variant Issues Project case study reports acknowledged that they were missing relevant expertise on some languages using the scripts, and it is not clear how subsequent efforts along these lines could attract more expertise. The advantage of the minimal setting is that it gets ICANN completely out of the role of evaluating people's claims about how their language works, but it would appear to invite attacks where someone sets rules designed to prevent someone else's TLD label from being permitted. It also seems likely to cause overall instability of the rules. A middle-range setting in this case might empanel experts to review proposals for rules that were previously solicited, but not require them to do all the development themselves.

The third parameter is that of code point property qualification (henceforth, "qualification"). This determines what would be required for a code point to be considered for inclusion in the zone repertoire in the first place. The maximal setting of this parameter requires that a zone repertoire consist only of code points all having the same Unicode script property. The minimal setting of this parameter permits completely arbitrary combinations of code points to make up a zone repertoire. The maximal setting has the advantage that it relies entirely on a property already defined by Unicode, but one should keep in mind that even LDH labels do not all come from the same script (HYPHEN-MINUS U+002D, and all the digits have the script property COMMON). The minimal setting avoids any problems having to do with multi-script writing systems by permitting arbitrary sets of code points; it also allows a script that needs but one or two code points from Common or Inherited to use them without requiring looking at the entire Common and Inherited ranges. But this flexibility might entail confusing or conflicting rules when more than one language community wants to use the same code point in different ways. Medium-range settings for this parameter might include treating Common and Inherited script properties as somehow being part of any script under consideration, or else starting with the Unicode script tables and then allowing arbitrary build-up of the zone repertoire by restricting the other ranges permitted to be considered.

4.2 Sample Options for Establishing Label Generation Rules

With these parameters in mind, it might be useful to consider some ways the parameters could be set in order to illustrate the range of options that they make available. These examples are not exhaustive; they're also only sketches, offered to provide an intuition about how the label generation rules might get established, rather than complete analyses of the different approaches.

Note that in what follows, "zone repertoire" and "label generation rules" are sometimes used informally to refer to the *subset* of the eventual zone repertoire and label generation rules. By definition, it is impossible to have more than one zone repertoire in a zone at one time, because the zone repertoire is just a list of the code points permitted in the zone. For practical reasons (as noted above) it is not possible to permit inconsistent rules about the handling of code points into the root zone.

1. Complete generation for every script table to be used, by an expert panel

   This scenario sets the comprehensiveness parameter nearly to its maximal value, the expert parameter to the maximal value, and the qualification parameter to its maximal value. In this scenario, ICANN would select some number of scripts (from those already defined by Unicode) it wants to support. It would convene one expert panel per script, and instruct them to build a completely comprehensive set of label generation rules for all the code points in the script. The zone repertoire for the root would thereby be established as the set of all and only the code points in all those script tables to be studied (except those DISALLOWED by IDNA2008). Each panel would develop the list of alternative code points and associated code point variant rules for the relevant script. Label generation rules would only be needed for scripts to be supported in the root zone, though of course it would be necessary to establish additional rules on the introduction of a new script. In the end, a single set of label generation rules would be merged from the collective output of the different script expert panels.

   Using this method, code points that are needed by some language, but that are not included in the script table used by that language, would not be permitted. Dealing with languages written in more than one script (e.g., Japanese) could be extremely tricky. Since the deliberation is at the level of a script, it would seem necessary to require that every code point in a U-label have the same script property as every other code point in the U-label. U-labels in the root zone would not need to have any identifying tag, because the script of the label could be derived from the script property of any code point in it (and any labels of mixed scripts would not be permitted).

It is not clear what one might do if a script expert panel were unable to come to consensus on the necessary rules for every code point in that script. If consensus were required, it would permit a single individual to block registrations for everyone. While we do not have examples of such fractiousness in our work so far, how a language is written is often an intensely (politically) fraught question.

In the event Unicode added or removed a code point from a script already included in the label generation rules, it would be necessary immediately to convene a new script expert panel to decide what to do about the new code point. It would be critical to prevent any registration with that code point until the panel had ruled, and probably necessary to have grandfathering rules for handling already-registered labels using code points implicated in new variant rules. If a code point changed scripts -- unlikely but not impossible under the Unicode rules -- then a similar immediate convention of the expert script panel would be needed. (It is worth noting that the most common code points available under IDNA2008 have been stable for a very long time, and are extremely unlikely to change in this way; all the examples are likely to be very rare characters. To all but eliminate this risk, it might be worth adopting a stability criterion so that a code point is ineligible for inclusion in the Label Generation Rules unless it has been completely stable in several Unicode iterations. Under this scenario, that would require restriction of the entire script; but other scenarios described below might permit restriction of just one code point.)

2. Assemble an expert panel for scripts likely to be desired, and include code points on a “best efforts” basis

   This approach lowers the setting of the comprehensiveness parameter, and leaves everything else the same. In this scenario, the scripts are again selected in advance by ICANN, and ICANN assembles the relevant panels to develop the rules for the script. The panel, however, need only rule on code points with which it is familiar. Code points that the panel does not feel qualified to address would be left out of the eventual zone repertoire delivered.

   This scenario has similar properties to the previous scenario, but differs in interesting ways. The zone repertoire for the root cannot be determined in advance, and can be derived only after all the expert panels have reported. Failure to reach consensus on some code point does not completely block progress: the panel could leave such a code point out on the grounds that consensus was not achieved.

This scenario also means that applicants for root zone U-labels would need to ensure that the code points they intended to use were actually in the zone repertoire. If not, such a potential applicant would need to engage whatever process was in place to try to alter the zone repertoire. During application evaluation, ICANN would need to add a step to ensure not only that the script in question was one of the permitted ones and that all code points in the U-label were in the same script, but also that all the code points were in the zone repertoire.

This scenario seems likely to reduce the chances of serious issues arising from changes in Unicode, because code points that are likely to have any changes are also the ones most likely not to be well-understood (and therefore to be excluded from the zone repertoire). The set of label generation rules is slightly more likely to experience some instability under this approach than under the first, because a new participant might ask for changes to the zone repertoire in order to add a code point from a script table that has already been studied.

Because this scenario is likely to require maintenance of the label generation rules that is more complicated than simply dealing with new Unicode versions, it might be necessary to create some sort of standing expert review panel. The details for this are something to be undertaken in the event of implementation. ICANN has other standing panels designed to ensure expert review of various technical and policy matters, and the composition of the label generation rule maintenance panels would presumably be determined along similar lines. The panel, however, would need to include expertise in all the relevant languages and scripts, as well as in the DNS and IDNA.

3. Create policies for script-relative lists of code points

   This scenario keeps the comprehensiveness and expert parameters the same as in the previous scenario, but lowers the qualification parameter so that zone repertoires may be built to extend the Unicode Script Tables. Because of the possible difficulties with “one Script Table” policies, it might be desirable instead to permit language communities to specify the code points they want to use to write their language. In this scenario, ICANN would assemble and direct the various expert panels for scripts as above, but explicitly permit the panels to extend the script table defined by Unicode with code points from other scripts. The panel would select some group of code points, which we call a **representation repertoire**, as the list of code points to be included in the zone repertoire. Similarly, the panel would determine **representation variant rules** that would be intended to be part of the code point variant rules. The combination of the repertoire and variant rules is called here the **representation label rules**, and all of these are to be identified by a **representation identifier**.

This scenario has many of the properties of the previous one, but it does not restrict the experts to predefined subsets of the Unicode code space. Instead, they are free to build the representation repertoire from the whole of Unicode. The expert panels might be organized around broad script categories (as the Variant Issues Project case study teams were), around a specific language or multiple languages, or even across linguistic lines in the event that seemed desirable (e.g., to address cross-script Latin/Greek/Cyrillic issues). Panels would work on code points with which they were familiar, and would usually need expertise from all the relevant language communities as well as experts in DNS and IDNA.

Different representation repertoires might include the same code point without harm, but it would run against the spirit of this scenario for broad swaths of different representation repertoires to include the same code points or be intended to be used with the same languages. The basic goal in this scenario is to produce something like the Unicode script tables while still allowing a small number of code points to be allowed across scripts; a subsidiary goal is to avoid having to deal with the entire Common and Inherited ranges in Unicode. When the different representation repertoires are combined to make a unified zone repertoire, it is necessary to include, in that zone repertoire, the representation identifier. This way, when a candidate U-label is submitted for inclusion in the root zone, it can be identified which representation repertoire the label is supposed to conform to, and it can be checked to ensure that all the code points in the label are indeed covered in the representation repertoire. (A label that did not conform to some such representation repertoire would be rejected.)

It is not clear what one might do if two panels were to set conflicting representation variant rules for the same code point. Some dispute mechanism would undoubtedly be necessary; the likeliest approach would be to block any potentially-conflicting variant characters across the representation label rules.

If a suitable existing repertoire were not available for a label someone desired, then those wishing to adjust an existing repertoire, or to create a new one, would need to request a new panel to create a suitable repertoire. Label generation maintenance panels, as outlined in the previous scenario, would likely be needed. As suggested above, it would be important to create a process that tended to discourage the creation of new repertoires or many modifications to existing repertoires. The reason for this is the same as the reasoning offered for variant labels in the first place: the “principle of least surprise.” If the goal is to ensure that users are not surprised, then rules that are different depending on when a label was added to the root zone would violate that principle of least surprise. Once a label generation policy permits a code point or establishes a variant rule, it should be difficult (although not impossible) to change that.

A notable advantage of this scenario is that it avoids the somewhat artificial link between the Unicode script properties and any given language. In addition, because it does not require any treatment of code points for which there is no demand until that demand appears, it avoids having to address code points for which the expertise is not available. A significant disadvantage to this approach is that it relies less on existing attributes (such as Unicode script properties) and relies more on expert groups to make findings about the code points. It also has the potential for "deadlock" and competing repertoires where different groups of users cannot come to agreement about how to deal with a subset of code points.

4. Evaluate community proposals for label generation rules

   This scenario would keep the comprehensiveness and qualification parameters the same as the previous one, but lower the "experts" parameter so that the expert panel would merely review submitted proposals from the community instead of developing the representation label rules itself. Because this scenario relies on the interest of the community to develop the representation repertoires and associated variant rules, this approach would minimize the amount of work expended on code points that nobody wants. On the other hand, it seems possible that experts would recognize interactions and issues that might not be included in volunteer submissions. This could lead to considerably greater instability in the label generation rules, as problems were identified (presumably sometimes after the fact of **activation** in the root) and then addressed. Much stronger conflict-resolution mechanisms would be needed for this scenario, but it appears that the experts would act as a filter that would perform most such conflict resolution.

5. Build up zone repertoires *ad hoc*

   This scenario sets the comprehensiveness, qualification, and expert parameters all to the minimum. In this case, interested parties of any sort could submit representation label rules they desire. All of them would be accepted in an additive way to govern the root. Instead of ICANN selecting an expert group, representation repertoires and associated variant rules could be created by interested parties, organized according to whatever principle they liked. The combination of these representation label rules into a single label generation policy would still need a meta-rule in order to deal with conflicts; in general, such a rule should always pick the most reversible action (i.e., blocking), to avoid controversial or possibly harmful entries in the root zone. (So in many cases, conflicting rules would cause potential variants to be covered by the rules all to be blocked.)

The principal advantage of this approach is that it would take ICANN completely out of the role of deciding whose expertise counts with respect to a language or script. A significant disadvantage of this approach would be that it is subject to denial of service: someone who is opposed to variants in principle (or in some specific case) could always create a conflicting rule for any representation label rule, and the conflict-resolution mechanism (in order to do the safest thing) would therefore tend to prevent any variant labels from being activated. Moreover, because this approach tends to encourage more innovative repertoires and associated rules (as there is little cost to adding one), it is more likely to lead to instability in the label generation policy.

5. Discussion of Issues: Treatment of Variant Labels

Once variant labels are identified, a range of possible states and corresponding actions may be taken on those labels. Thus, a variant management mechanism could encompass both active use of labels in the DNS, and prevention of labels from use in the DNS. The possible states that apply to a name are as follows:

Blocked: A status of some label with respect to a zone, according to which the label is unavailable for allocation to anyone. The term "to block" denotes the registry (the zone operator) taking this action.

Withheld: A status of some label with respect to a zone, whereby the label is set aside for possible allocation to some entity. In this strict sense, a withheld name is not actually allocated. The term "to withhold" denotes the registry (the zone operator) performing the setting aside.

Allocated: A status of some label with respect to a zone, whereby the label is associated administratively to some entity that has requested the label. This term, and its cognates "allocation" and "to allocate", represents the first step on the way to delegation in the DNS. When the registry (zone operator) allocates the label, it is effectively making a label a candidate for activation. Allocation does not, however, affect the DNS at all.

Activated/Active: A status of some label with respect to a zone, indicating that there are DNS resource records at that node name; or else that there are subordinate names to that name, even though there are no resource records at that node name. In the case where there are resource records at the node name, any resource record will do. In the case where there are subordinate names but no resource records (except those to support DNSSEC), the label names an empty non-terminal. A registry (zone operator) setting the active status activates the name, or performs activation.

Delegated: A status of some label with respect to a zone, indicating that in that zone there are NS resource records at the label. The NS resource records create a zone cut, and correspond to an SOA record in the subordinate domain. The act of entering the NS records in the zone is delegation, and to do that is to delegate. This definition is largely based on RFC 1034[20]; the reader should consult RFC 1034 for detailed discussion of how the DNS is broken into zones.

Mirrored: A status of some active label with respect to a zone, indicating the isomorphism of the namespace beginning with that label, and at least one other namespace beginning with another active label in the zone. If two domain names are mirrored, then for a namespace

[20] http://www.rfc-editor.org/rfc/rfc1034.txt

starting with one, the namespace starting with the other is isomorphic to the first, subject to the usual DNS loose consistency strictures. The act of setting two or more labels to be mirrored *and maintaining the namespace correctly* is mirroring. Currently, there are two different techniques for this. The first is aliasing: CNAME, DNAME, and other such techniques that redirect a name or a tree, effectively substituting one label for another during DNS lookup. The second is by using provisioning constraints, such that an underlying provisioning system always effects a change in all of the names whenever that change is effected in one of the names. The set of domain names (not labels) that are supposed to be the beginning of isomorphism are mirrors. Mirrors whose namespaces have not been maintained to preserve isomorphism are broken mirrors.

The state values blocked, withheld, and allocated are mutually exclusive. The active (which includes but is not limited to delegated and mirrored) state usually implies that allocation has occurred.

These states may result in a different user experience, as well as having an impact on the operations of ICANN, the TLD registry operator, and other stakeholders that are part of the Internet ecosystem. This section discusses the issues that arise in this area as a result of IDN variant TLD labels.

5.1 Possible States for Variant Labels

The states associated with the actions described above are of the predictable forms: blocked, withheld, allocated, activated, delegated, and mirrored.

Since blocked means that nobody may have the allocation, it follows that nobody could request such a state (for that would imply some sort of proto-allocation over the domain); therefore, blocked is a pure matter of registry policy (in the context of this report, an action taken by the registry for the root zone, i.e., ICANN). Blocking could be the consequence of the combination of a registry policy and some state of the registry. For example, the rule for a code-point could be such that, if a label containing that code point is allocated, then a label containing some other code point must be blocked.

Because blocking is a matter of registry policy, the change of a label from blocked to any other state is either a consequence of a change in registry policy, or else a change in state in the registry such that the blocking condition (e.g., the allocation of a label with a block-generating code point) is removed.

A withheld status results from the combination of policy and the requests of applicants. For example, suppose a registry policy permits (but does not require) a TLD label and its variant TLD label to be delegated. In the event the applicant chooses not to delegate the variant TLD label, registry policy may require that the variant TLD label be withheld from all others but the applicant throughout the registration lifetime of the **fundamental label**. In accordance with an applicant request, the variant TLD label could move from being withheld to being allocated (and then anything that can be done with an allocated label would apply).

In principle, a change in registry policy (but not a change in registry state) could cause a label to move from withheld to blocked. If a change in registry state were to cause this, it would be an indication that the registry policy was inadequate in the first place, because it had not addressed potential conflicts well enough. However, since both states imply the label is not active in the DNS, the implications for users seem negligible, if any.

An allocated status would result from any case where an application containing a variant TLD label is approved, and it is not blocked or withheld. An allocated status could also result when a variant TLD label associated with a fundamental label is either also requested to be activated, or required by registry policy to be activated (see below). By request, the status of an allocated label could be changed to withheld. Allocated labels may normally be activated, subject to usual registry policies, at the initiative of the registrant of that label (i.e., in the context of the root zone, usually the TLD operator). A change of state from allocated to either withheld or blocked should have negligible implications for users other than the registry to which the label was allocated.

An activated status results from placing some sort of DNS record into the parent zone such that it is possible to perform a DNS lookup on the name and receive an answer. It is possible for the registry to require the simultaneous activation of a group of names, denying all of them if any of them cannot be activated. Delegation and mirroring are just species of activation. A domain can be de-activated by removing the relevant records from the DNS (for technical reasons, when there is a delegation the name often has to be removed from both the parent side and child side name servers in order that it stop resolving, but the details of how to ensure a name has stopped operating completely in the DNS are beyond the scope of the present report).

De-activating a TLD in the root could have grave consequences on users if such TLD has itself active names under it. Removing a TLD from the root zone is by no means a simple operation, and the few cases that have happened in the past took in the order of years to complete.

5.2 User Experience with Variant Labels

The case study reports discussed issues to be considered from the user's perspective, such as input methods like keyboards, and software issues. While many of these issues are directly relevant to user experience of IDNs, they are not a result of variant labels as such (although input methods may drive some calls for variants). The use or expectation of IDN variant TLDs, however, affects user experience and this section explores those issues. On top of these issues, it is important to keep in mind the limitations in the overall Internet operating environment described in Section 1.

5.2.1 IDNA2003 to IDNA2008 migration issues

Despite the definition of domain name labels as octets (which means, in principle, that there are no character set-specific rules about the handling of those labels), RFC 1034 defines a Preferred Name Syntax[21]. This is the source of two features of the DNS that cause trouble for internationalization. The first is the “LDH” convention: conforming labels are made up of letters, digits, and the hyphen. The second is the issue of case.

The DNS is often said to be “case insensitive”, but that does not quite capture how it behaves. For LDH names, it is case-insensitive, but case-preserving.[22] This means that, while the domain name labels “example” and “ExAmple” match one another for the purposes of lookup, they may be displayed differently to users. This facility is at least sometimes used to make names easier to remember (or pronounce correctly).

This feature of LDH naming may have established a user expectation that extends to other scripts that use case, such as Cyrillic, Greek, and Latin. Unfortunately, IDNA does not work similarly with respect to case. IDNA2003 folds the case of upper-case code points to lower-case[23] on the Unicode form of the label it receives. This step often causes diacritics to be lost, which means that the case-folded name may not match the name stored in the DNS. By contrast, under IDNA2008 all upper case code points are disallowed. Application software may or may not do something with upper case code points (though there are recommendations that applications perform a case fold operation prior to putting the string into NFC form to be used as a U-label; see RFC 5895[24]); but even after the application has performed some mapping, the results may be surprising to the user. To give an example: to many French speakers, the upper case of LATIN SMALL LETTER E WITH ACUTE, U+00E9 (é) is LATIN CAPITAL LETTER E, U+0045

[21] RFC 1034, section 3.5

[22] Case insensitivity is specified in RFC 1034, sections 3.1 and 3.5. Section 3.1, however, says that case is to be preserved on receipt.

[23] This is an oversimplification. More accurately, the Nameprep stage of IDNA2003 processing uses a Stringprep appendix that specifies case-fold mappings in many instances. See RFC 3454, Appendix B.2 for Stringprep, RFC 3491 section 3 for the specification of which mapping to use, and RFC 3490 for the specification of when to use Nameprep.

[24] See http://www.rfc-editor.org/rfc/rfc5895.txt

(E) rather than LATIN CAPITAL LETTER E WITH ACUTE, U+00C9 (É). But the case-folded form of E is e, so someone who tries to use the label "ECOLE" will not manage to reach the label "école".

This issue is not directly related to variants, but it is rather a general issue of IDNA support. It might, however, be one important type of variant-inspiring behavior, because most of the deployed code today supports IDNA2003 and not IDNA2008. As a result, one class of desirable variants might be those that cause the loss of information due to case folding not to be apparent to users. The Latin case study report rejected variants of this (and every other) sort because they are not generalizable in the Latin script.[25] The issue is nevertheless much (if not all) of the basis for the Greek case study report's recommendations with respect to tonos and final form sigma.[26]

5.2.2 Types of users

The users of the DNS are varied in respect of what they expect, what they want, and what they need. The deployment of any variant strategy will affect the different types of users differently. There are two basic categories of affected user. One category includes those who are directly involved in some way in the registration or operation of the name in question. The other category includes only those who are in no way involved in the operation, but interact with the name somehow. Some people can fall into both categories, but only these two groups are relevant. They can be described as follows:

- System administrator of the affected systems: One who has to configure the variant domain names to work on the target systems;
- Other network operators not involved in operating the target name: People who need to deal with network traffic from multiple names that are somehow supposed to be the same;
- Registrants: The primary contact for registration of a name;
- Registrars: The service providers who perform actions on the names during registration (registration privacy vendors and resellers would be special cases of these);
- Registry operators: The service providers who provide the authoritative repository where a name (and its variants) might be registered;
- Software developers: The developers of software such as hosting management systems that need to be able to treat multiple names as somehow being variants of one another, as well as end-user software like web browsers that will need to be able to cope with cases where variants are exposed to them;

[25] Latin case study, p.1

[26] Cf. Greek case study, sections 11 and 12, pp. 9 ff

- Law enforcement and other security investigators: Those who may need to interact with a domain and any of its variants due to legal or network security reasons;
- End users: the average Internet user exposed to the domain name and any variants. This is the user "reading the label on the side of a bus," but it extends to anyone who is exposed to variant labels on the Internet.

The effects on registry operators and registrars are covered in section 5.4. The rest of the affected parties are considered below.

It is important to recognize that some parties who might be affected are not affected in every case. For instance, while a registrant is affected no matter what the status of variants labels, other network operators experience traffic (and therefore possible cost) only when a name and its variants are actually in the DNS and originating traffic.

5.2.3 User capabilities

Any user of any of the types listed will have some capabilities with respect to any given internationalized label. There are three broad categories of capability.

The first is competence in the script or language relevant to the label. A user with this capability can understand the label (or, anyway, find it intuitive enough to use – there is no reason to assume the label will be a word), has the ability to type and otherwise use the label, and so on. A native user of simplified Chinese encountering a simplified Chinese label while using his or her own computer might be an example of this category.

The second category is of limited competence with the script or language relevant to the label. A user with this capability might be able (for instance) to read the script well enough to recognize roughly what the label is, but not read the intended language of the label well enough to use it easily (consider an English speaker confronted with two labels, one intended for a Swedish audience and one for a Norwegian audience). On the other hand, the user may simply be faced with technical limitations that cause difficulty in the use of the label (imagine someone at a public-terminal keyboard in Iran attempting to write a label that includes ARABIC LETTER YEH, U+064A, ي).

The third category is of no familiarity with the relevant script or language. Users in this category can at best cut and paste unfamiliar symbols, and may not even be in a position to do that due to display limitations or other issues. They cannot read the labels, and the **A-label** form of the label is exactly as useful (or useless) to such a user as the U-label form.

5.2.4 What users expect and what they get

For each category of users, it is useful to think of the user experience in terms of what the user might expect. Imagine a base string X and another string Y. X is a U-label that makes a top-level domain. There are four possible experiences a user might have:

E1: The user expects Y to work as a substitute for X, and it does.
E2: The user expects Y to work as a substitute for X, but it does only partially or unreliably.
E3: The user expects Y to work as a substitute for X, and it does not.
E4: The user does not expect Y to work as a substitute for X, and it does.
E5: The user does not expect Y to work as a substitute for X, but it sometimes does.
E6: The user does not expect Y to work as a substitute for X, and it does not.

For the present purposes, E6 is not that interesting: without any variant effort, this expectation and experience would be met by the traditional operation of the DNS. Therefore, we will concentrate on cases E1-E5. For the following discussion, imagine two candidate fully qualified domain names: example.X and example.Y.

Before proceeding, it is necessary to expand on this very simplistic notion of "work." We have here distinguished between "not working at all" and "working partly." Given the loose consistency requirements in the DNS, most services on the Internet do not work or fail to work all at once: with the exception of cases where a service is simply not configured or activated (not working at all), most cases of "not working" are intermittent. Failures related to DNS resolution or inconsistent resolution results take time to propagate, and an observation made by one person from one position on the network may not be repeatable elsewhere. Experience with deployed systems suggests that configuration is likely to have frequent minor inconsistencies, with some system or some part of some system incorrectly configured instead of being completely broken. These considerations, and the general principle of robustness for Internet servers, mean that inconsistent behavior or intermittent failures are more likely than complete failure. In the following discussion, "does not work" merely means "the support is not available or it is somehow disabled"; every other type of failure is a partial failure. But it should be remembered that in many ways this sort of partial failure (or inconsistency) is worse, because users cannot discern a reliable pattern.

Moreover, the example that we are using is itself artificially constrained: in the case where there are variant TLDs, there is every reason to believe that there would be variants at the second level as well; so the two names example.X and example.Y would in practice likely be a list of several names instead. The number of fully-qualified names would be the number of TLD variants times the number of second-level label variants. In general, the number of fully-qualified names in the entire set of names to be managed is determined by multiplying the number of variants in each label of any one of the fully qualified names in the set by the number of labels in that member of the set. So, If there were three levels (e.g. 'www.example.X' and variants of all those levels of label) then the number of variant names is the number of variant TLD labels times the number of second level variant labels times the number of third level variant labels.

System administrators

The system administrator has a practical problem to address whenever performing any administrative functions in an environment where variants of any kind are active: as of this writing, there are very few tools that already support administration of IDNA labels using the U-label form. This means that administrators need mostly to work with names in A-label form: "xn--sample-punycode-output-here." Moreover, tools mostly do not have direct support for linking various names together, so as far as the tools are concerned these variants are all just separate names from one another. For instance, apart from adding parallel configuration for different names to a mail server, many mail servers have no way of telling that the same local-part at example.X and example.Y are really the same mailboxes.

In the case of E1 (user expected a variant and such is implemented), the system administrator for the affected domain has to configure all the services at example.X also to respond to example.Y (so that, for example, the web server responds to web page requests on http://example.X and also http://example.Y; mail to user@example.X and mail to user@example.Y both get directed to the same mailbox; and any other service that is aware of the name with which it was contacted works in parallel as well). If the system administrator does this, then each such service will work the same way for example.X and example.Y. There is, of course, a maintenance cost, in that services for example.Y need to be reconfigured whenever services for example.X are. Similarly, any site renumbering and so on will require twice the effort (and additional effort for each additional working variant). If the system administrator fails to maintain the services in a coordinated fashion, then there will be inconsistent results and possibly service outages.

It is difficult to know how case E2 (user expected a variant and it works unreliably) could happen here, unless the TLD operator of X and Y were having problems with Y. For the system administrator, such a case would mean service outages and surprises that would be difficult to diagnose.

In the case of E3 (user expected a variant which is not implemented), the system administrator configures all the services for example.X also to respond to example.Y, just as in the case of E1. The effects of this, however, are either negligible or mildly dangerous, depending on how the parent domain is administered. If Y is not active because it is allocated-only, blocked or withheld as a possible variant, then there is little harm: the system administrator’s systems respond to a name that will never be looked up on the Internet, so no damage occurs. Matters are different, however, if a system administrator configures the systems to respond to example.Y, but Y is in fact a completely separate domain delegated to a different registry operator than X. In this case, the system administrator has configured the systems to operate using a name that the system administrator does not actually control. In principle queries for example.Y should not arrive to the servers for example.X (since the parent name does not consider them variants), but they could if the servers are also recursive. In effect, the system administrator will have accidentally hijacked example.Y, from the point of view of those querying the servers configured by said system administrator.

In the case of E4 (user does not expect a variant but such exists), the system administrator does not configure anything to respond to example.Y, but it turns out that Y is in fact an activated variant of X. In this case, the service will receive connection attempts on example.Y, even though the services believe example.Y is not one of their names (for services that are aware of the name with which they were contacted). The registrant of that name might in this case lose web hits, emails, messages, etc. This case might be something that would cause others to have E2- or E3-type experiences.

The case of E5 (user does not expect a variant, but it works sometimes) is, for the system administrator, just like E4 only with the addition of intermittency. The intermittency could be only in time: because the failure would have to result from DNS failures in Y (or its children), it is unlikely that the failures would be restricted only to some services.

Other network operators

Network operators who are not directly related to the operation of example.X may nevertheless encounter example.X (and example.Y), because the network operators are intermediaries between the operator of example.X and other end users. The effects on these operators are mostly subtle and not that serious; on the other hand, if they happen they will contribute to a general failure of reliability of variant names. Moreover, for very large network operators, even relatively small numbers of failures can result in significant increases in logging and troubleshooting costs. Additionally, given the state of the DNS today, there will be no ways inside the DNS for a network operator to tell that example.X and example.Y are related to one another; and because the network operator has no existing relationship to these names, he or she would have no reason to suppose they are related to one another. The operator would have to resort to tools like WHOIS, which are poorly adapted to use in automated systems.

In the case of E1 (user expected a variant and such is implemented), the network operator sees no effect, but logs will be less compressible than they otherwise would be. Even at large sites, the additional compression overhead is unlikely to be a really significant cost, but because this is an effect it is included here for completeness.

In the case of E2 (user expected a variant which works sometimes), the network operator occasionally experiences the same problems as in E3. This "sometimes" might be over time, or it might be consistent but related to a single service. For instance, if the system administrator discussed above failed to keep the configuration for the mail server up to date, mail to example.X might work while mail to example.Y failed; if the mail was coming from the network operator's site, then the network operator may experience increased support and troubleshooting costs.

In the case of E3 (user expected a variant which is not implemented), the network operator may experience increased customer support costs, as customers of said network operator find that example.Y does not work. In addition, for mediated services (like email, where the network operator likely runs the outbound mail server) there will be additional load due to failed transmissions.

In the case of E4 (user does not expect a variant but such exists), the network operator may experience traffic from an unexpected source. This could cause the network operator to react to the traffic as abuse, or just to reject it, particularly if the network operator is expecting the reverse DNS tree and the forward DNS tree to match exactly. If the system administrator for example.X and example.Y is doing a good job, however, this case is not very likely.

A similar effect, rather more likely, would happen in case of E5 (user does not expect a variant, and one works sometimes). Suppose that the system administrator of example.X and example.Y has done an indifferent job, and so example.Y is incorrectly configured in the DNS (such that it has no mail exchange). Users of example.X, however, sometimes use the example.Y form when sending their email. Inbound email to the network operator's mail server might be checked for proper configuration, in order to ensure that mail from user@example.Y is legitimate. Because example.Y could not possibly get mail back, the network operator might reject the mail (probably discarding it), and might begin to treat the source of the mail as a source of spam. This treatment might even affect mail from example.X, because many spam handling systems work by identifying the IP address (of the source mail server) rather than domain name.

Registrants

A registrant is in some sense in the best position to refine his or her expectations of what will happen with a name, but is also constrained by registry policies and the available registry and (where they are involved) registrar systems in what the registrant receives. In particular, in our example we are supposing a registrant of example.X, so the variant relationship is at the parent. If the parent operates by doing automatic mirroring (by, for instance, using a DNAME record to point Y to X), then the user might not be aware that he or she is in fact registrant of two different DNS names. (Compare also to the system administrator discussion, above.)

Before consideration of the cases, it is important to note that registrants need to have an understanding of the various possible states of variant labels whenever they have options for managing the variants. For instance, in a case where a registry automatically withholds variants but does not allocate them without some action on the part of a registrant, the registrant needs to know to look for this action. Similarly, if there are restrictions on how a variant name may be operated (like a restriction to Mirroring, for instance), the registrant needs to understand these rules. This may be more information about the domain name life cycle than a registrant is usually prepared for.

Case E1 (user expected a variant and such is implemented) is of course the ideal for the registrant, as long as the registrant actually knows that the support is there.

Case E2 (user expected a variant but it only sometimes works) might happen if the chain of systems between the registrant and the registry had missing or buggy features: the management of a variant label, for instance, might not work correctly. This is likely to cause apparent service failures on example.Y.

Case E3 (user expected a variant which is not implemented) is the one that is most difficult for the registrant, because the registrant expects certain things to work that do not. This leads to unexpected failures (e.g., a web site that does not serve all the visitors that were expected, or mail that cannot be received from some people). This failure case is also likely to be a surprising one – that is, the registrant likely does not know to look for a failure in the first place, and therefore is not aware that the problem exists. (If the registrant knows about the problem, of course, he or she could simply register the additional name, assuming the Y TLD also existed. But the primary motivator for variants is for the case where humans do not detect a difference even though machines do.)

Case E4 (user does not expect a variant but such exists) is possibly a pleasant surprise for the registrant of example.X. There is also the case of a potential registrant (not to be confused with that of example.X) that tries to register example.Y, which may result in some confusion for a non-Active variant. However, this case is no worse than when a user expects to register a name and finds that it is already registered in a scenario with no variant TLDs.

Case E5 (user does not expect a variant but it works sometimes) is not much different from case E4.

<u>Software developers</u>

The software development communities that are affected by variants are quite different from one another. On the one hand are those who are developing systems and support tools for system administrators and others (we can call these "systems programmers"). This group will need to be able to expose as much information as possible about variants, because the administrators of the systems need to be able to manage in small increments. On the other hand are those who develop software aimed at end users (we can call these "application programmers"). This group needs to make the variant experience as smooth as possible for users, hiding the details of differences in the DNS so that the user's experience is as though there were no variants at all. The good news for all of these cases is that, if a developer makes an effort to support the scenario well one time, that effort is completely reusable for any other such cases. In both cases, the programmers have to build for a completely generic situation and therefore cannot take into consideration hints about the likely situation of the user (except in very unusual circumstances such as custom software development).

For case E1 (user expected a variant and such is implemented), the systems programmer needs to be able to link together example.X and example.Y so that the names function as one; at the same time, the systems programmer needs to be able to detect the differences so that configuration can happen correctly for variant-unaware utilities. The support tools will not be able to do everything they need to do via the DNS (at least initially), because there is no way in the DNS to signal reliably the link between two names. (When using aliasing like DNAME, the link is one way; so if Y has a DNAME pointing to X, you have no way of knowing that there is such a DNAME record when you look up example.X.) Because X.509 certificates (which underlie security protocols like TLS, which is the current standard for, e.g., commercial transactions on the Internet) are tightly linked to the domain name with which they are used, application programmers will probably need to be able to recover from TLS failures relating to variant names. Such behavior is today unspecified, and so will need to be specified if it is ever to work.

(Such specification, and the subsequent deployment, should be expected to take several years even if they happen.)

In case E2 (user expected a variant that only works sometimes), the systems programmer's situation is just like E1. Poor tools are likely to cause E2-type experiences for systems administrators, however. E2 cases present a problem for application programmers, who will have a difficult job knowing how to support their users.

For case E3 (user expected a variant which is not implemented), there is no work for the programmers to do: if the feature is not supported in software the programmer connects to, then the programmer cannot do anything.

Case E4 (user does not expect a variant but such exists) is just like case E1, really, except that in some ways it is likely to be harder for the systems programmer. When a user expects a variant to work and it does, the systems programmer has someone to ask about what he or she would have expected to work. This means that the systems programmer can rely to some extent on system administrator input to help with configuration and discovery. If the system administrator does not know about variants, however, the systems programmer's code might want to be able to find out about them automatically. As noted in case E1, such discovery is difficult. Because an application programmer aims to make the user experience as smooth as possible, this case is just like E1 (or perhaps more accurately, case E1 is actually case E4).

Case E5 (user does not expect a variant, but one works sometimes) is similar to case E1, but is likely to be treated by developers as a case of E6. This will probably lead to E3-type behavior for others.

Law enforcement and security

Those performing law enforcement or security investigations may be a special class of user in that their needs lie somewhere between those of the end user and those of the (other) network administrator. They have in principle no special relationship with the operator of the domain name in question, but encounter the names in question as part of some other activity (like an enforcement action or a security breach investigation), and not as part of an effort to obtain some resource or direct some traffic towards the domain.

Case E1 (user expected a variant and such is implemented) is desirable: the user gets exactly what he or she expects, and can discover all the links among (in the example) example.Y and example.X.

Case E2 (user expected a variant but it works only partially) might be troublesome. If the problem is intermittent functioning, then the user might interpret that intermittency as part of the problem and be led astray in the investigation. If the problem is only partial or inconsistent support, then the user might have a more difficult time undertaking the sort of investigation he or she desires. For instance, if WHOIS support is incomplete and fails to reveal all the links one might need, then the user may not realize that the different names are somehow to be treated as related to one another.

Case E3 (user expected a variant but it does not work) is unlikely to cause any troubles, because this sort of user is going to be interested in things that are (or were) actually working.

Case E4 (user did not expect a variant and it does work) is quite bad for this type of user, because the user will not realize that the name he or she is investigating is not the only name under investigation. It might be that the array of support tools available to the user will reveal the links among the different names, in which case there is merely confusion rather than failure to complete an investigation. If, however, the investigation relies primarily on the DNS and aliasing techniques such as DNAME are in use, it may be practically impossible to discover the variant relationship.

Case E5 (user did not expect a variant, but it works sometimes) is similar to case E4, with the added complication of intermittent or inconsistent behavior.

End users

The end user is in a position not dissimilar to the network operator, in that the end user does not in principle have any special relationship with the operation of the domain name in question and so does not have any information that might lead to the belief that a variant ought to work. End users are the primary target of support for variants: the naïve user who types a string expecting it to function as a domain name, or who clicks on a link seeing it as connecting to a familiar web site, is at bottom what motivates any type of variant. Therefore, case E1 (user expected a variant and such is implemented) is the ideal one: the end user got exactly what he or she expected, without having to worry about how a name was actually made out of Unicode code points.

Case E2 (user expected a variant, and it works inconsistently) is a frustrating experience for users. With a name that works under some circumstances and not others – either over time, or in some services and not others – the user cannot easily learn a new pattern of behavior with respect to the DNS. This inconsistency is also one that will be problematic across domains. In the absence of consistent and predictable variant behavior that corresponds to users' practical model of how naming works on the Internet, any variant strategy will quickly fall away as users start to treat domain names as unreliable. Given the inconsistency of conventions for relating various names to one another in the DNS today (and all of the system administration problems such related names would bring), users are today experiencing E2 cases.

Case E3 (user expected a variant which is not implemented) is the worst case for end users, and it has two failure modes. The first is denial of service: the user attempts to visit http://example.Y, reading it as being the same URI as the http://example.X that he or she saw in an advertisement, but the connection does not work because A is either blocked, reserved, or allocated-only; or has no variant at all, and example.Y is not registered. In this case, the user is frustrated but no serious harm has arisen. The second mode is a false connection: the user attempts to visit http://example.Y, reading it as being the same URI as the http://example.X that he or she saw in an advertisement, but arrives at a site controlled by a registrant different to that of example.X. Even if this effect is not the result of malicious work on the part of A's operator or example.Y registrant, it is bad. It is extremely unlikely that phishers and other malefactors will go to the trouble and expense of registering a TLD when there are much cheaper alternatives already available and which are just as effective. But even an accidental connection of this sort to a perfectly legitimate site operating at example.Y presents issues of possible credential leakage, accidental disclosure of information, and user confusion and frustration.

Cases E4 and E5 (user does not expect a variant but such exists or works sometimes) is an uninteresting case since the end user is not affected in any way either for good or bad.

5.2.5 Consistency

As noted above, from an end user's point of view, there are two kinds of inconsistency that could be observed in the face of variant labels. The first is inconsistency with respect to a given name or service: a name and its variant might work for one service but not for another, or the variant name might work some days and not others. But there is another type of inconsistency, at least as bad, and which is already deployed today.

Consistency of behavior of a system – its predictability – is a critical feature for that system to be relied upon. In the absence of consistent and predictable variant behavior that corresponds to users' practical model of how naming works on the Internet, any variant strategy will quickly fall away as users start to treat domain names as unpredictable. Given the inconsistency of conventions for relating various names to one another in the DNS today (and all of the system administration problems such related names would bring), users are today experiencing E2 cases. A model of variants that expands their use at the expense of the predictability of Internet naming systems would be harmful.

5.2.6 Chains of actors and distributed systems

The Internet is mostly made up of distributed systems, and with a few exceptions it is rare that systems dependent on a large number of people all doing the same thing at the same time get deployed. The practical difficulty of getting everyone to change naming systems at the same time on the Internet, for instance, is part of the reason that IDNA avoids actually changing the DNS protocols themselves, even though that decision means that several desirable features cannot reasonably be accommodated. But as we see from the example scenarios above, a suitably invisible end user experience in the presence of variant names is difficult to predict. It seems to depend on several independent actors all acting well at the same time, with failures at one point in the system cascading through until visible to the user. Because of the distributed nature of the Internet, and the protocol layering that is one of the Internet's chief strengths, the technical problems raised by variants are not amenable to a simple technical solution.

5.3 ICANN Operations and Variant TLDs

In applying a variant management mechanism for the root, ICANN would incur a number of operational issues, discussed in this section.

5.3.1 Evaluation

The evaluation processes for gTLD applications are documented in the gTLD Applicant Guidebook.[27] IDN ccTLDs may currently be requested via the IDN ccTLD Fast Track process, in accordance with the procedures in the Final Implementation Plan.[28] The Fast Track was intended to enable the introduction of a limited number of non-contentious IDN ccTLDs, associated with the ISO 3166-1 two-letter codes, to meet near-term demand while the overall policy is being developed. The ccNSO is undertaking a policy development process[29] concerning the introduction of IDN ccTLDs.

These current processes do not allow for the delegation of IDN variant TLDs until such time as a variant management mechanism for the top level is in place. This section describes the various evaluation activities that could be required in a scenario where variant TLDs were able to be requested. These can be considered according to two

[27] http://www.icann.org/en/topics/new-gtlds/rfp-clean-19sep11-en.pdf

[28] http://www.icann.org/en/topics/idn/fast-track/idn-cctld-implementation-plan-16nov09-en.pdf

[29] http://ccnso.icann.org/policy/cctld-idn

types of evaluation: 1) evaluation activities relating to the variant request, and 2) additional variant considerations in existing evaluation activities.

5.3.1.1 Evaluation of Variant TLD Requests

These activities could be covered by the Label Generation Rules, and the issues involved depend significantly on the approach taken to establishing these rules, as discussed in section 4.1 above. If, at the time it is evaluating specific requests, ICANN can refer to an existing authoritative source that will produce a definitive result on whether two strings can be considered variant TLD labels, the impact would largely consist of new sub-processes to take the variant labels into account within existing evaluation steps. In the ad hoc approach discussed above, ICANN would not perform any evaluation except to determine whether any label generation rules were conflicting.

If label generation rules are not pre-existing at the time ICANN is considering requests, then this work could occur in parallel with specific requests, resulting in possibly extended timelines and, depending on the approach adopted, additional review steps for ICANN.

In such a case, existing evaluation processes might need to add a specific "variant review" to the evaluation process, to account for a review and determination on the set of variant TLDs itself. Issues to be addressed would include:

a. What would be the selection process and qualifications needed for individuals performing evaluation tasks? (The Chinese case study report, for example, has identified a need for linguistic experience and knowledge.)
b. What would be acceptable reference sources for evaluators to use in making judgments?
c. How would ICANN ensure that evaluators were unbiased in their decisions?
d. What precedential value would be set by evaluator decisions? Could each case continue to be assessed individually or would this create conflicts over time?

The relevant costs incurred by ICANN in performing evaluation processes would also vary depending on the approach taken to establishing label generation rules. The level of case-by case-review required, and the number of determinations required on requests would have a significant impact on both the resources needed to execute such reviews and the risks to be anticipated in such a process. To the extent that additional costs are incurred by ICANN in connection with the evaluation of requests for variant TLD labels, an additional aspect of the set of evaluation issues concerns whether there should be additional fees instituted to cover any relevant costs.

The issues associated with treatment of variant TLD labels (i.e., possible states which may apply to a label) are discussed in section 5.1 above. If label generation rules are in place to determine how each type of variant label can be treated, the evaluation ICANN must perform is limited to confirmation that a request does not conflict with these requirements. In the absence of established rules, the desired states for each of the labels would also need to be reviewed in evaluation. Issues in such a case, in addition to those mentioned above, would include whether a certain variant TLD label in a requested state would create either a) a security or stability issue, or b) a user confusion or user experience issue. State changes could require a specialized evaluation process in some cases.

These issues would seem to be consistent across scripts. While the sources and standards used by evaluators would vary according to the relevant script, the evaluation steps themselves would not differ.

5.3.1.2 Variant Considerations in Existing Evaluation Processes

In the event that requests for variant TLDs could be submitted, application processes would need to account for different types of construction of variant sets depending on the script or language involved. In some cases, there may be a "base label" with variant labels associated, while in other cases, a set of variant TLD labels would be essentially equivalent in status. Could a common format for characterizing a requested set of TLD labels be developed or would there need to be flexibility for each case to use a specific notation? Another issue would concern whether there was a maximum size to one of these sets, and what conditions might apply depending on the number of labels involved.

Once the variant TLD labels in an application have been accepted through one of the means described above, the evaluation tasks for a TLD application that included variant TLD labels would proceed through the usual stages as relevant. This could include reviews for:

a. String similarity – This review determines whether an applied-for TLD is so similar to an existing TLD or other applied-for TLD that it creates a probability of user confusion. In the review steps developed by ICANN, this analysis is limited to visual similarity. The string similarity review takes place in the interest of avoiding user confusion through the delegation of many similar TLD strings. Issues to be considered in the case where variant TLD labels are part of an application include:

    1. The methodology for considering the set of variant TLD labels in an application in the string similarity review against existing TLDs or other applied-for TLDs.
    2. Whether this review would differ based on the requested status (e.g., delegated, withheld) for any of the variant TLD labels in the application.

b. DNS stability – This review determines whether an applied-for TLD meets the technical string requirements. An issue to be considered in the case where variant TLD labels are part of an application is whether all variant TLD labels included, whether intended for active operation in the DNS or not, would need to pass this review.

c. Geographic names – The IDN ccTLD Fast Track process contains a meaningfulness requirement: string(s) must be a meaningful representation of the name of the corresponding country or territory. The gTLD evaluation process contains a review to determine whether evidence of government approval is provided where required. In the case where variant TLD labels are part of an application, it would need to be determined whether such requirements would necessarily apply to every label contained in the application, including the variant TLD labels, or whether the requirements could be differentiated within the set.

d. Technical/Operational/Financial capability – These reviews take place in the gTLD evaluation process and test whether the applicant has the requisite technical, operational, and financial capability to operate a TLD registry. An issue to be considered here is whether there are additional reviews that ICANN should undertake concerning the variant management mechanism itself, i.e., to ensure that any guidelines or requirements concerning the applicant's operation and management of the variant TLDs are anticipated and understood by the applicant, and taken into account in its plans.

e. Registry services – This review takes place in the gTLD evaluation process and determines whether the registry services offered by the applicant might adversely affect DNS security or stability.

As an additional consideration, the gTLD evaluation process contains a mechanism for formal objection to be filed to an application on certain limited grounds. A formal objection by a party with standing will trigger a dispute resolution proceeding with an expert panel rendering a determination. Note that such an objection is to an application, not necessarily to a string. In the case of an application containing variant TLD labels, an issue to be addressed is whether the set of labels in the application would always be maintained as a set for purpose of the objection and dispute resolution proceedings, or whether they could be split, so that only one of the labels within the set was at issue. For instance, a withheld variant label that is added to the variant set by a mechanical rule might not be a basis for a valid objection to the original application, while an activatable label would be. The outcome of a dispute resolution proceeding on an application containing variant TLDs could be either to a) uphold the objection (i.e., the objector prevails), b) deny the objection (i.e., the applicant prevails) or c) a split result where the objection was upheld only for certain TLD labels within the set, meaning that the label generation rules could be subject to an exception based on this determination. If label generation rules are subject to exception, however, they no longer provide an algorithmic way to determine whether a variant arises from a given candidate label.

5.3.2 Management of Established Variant TLD Labels

As discussed above, one possible approach incorporates a common set of rules developed for the root zone, to determine the circumstances under which variant TLD labels may be categorized with a given status (cf. section 4 above). In the absence of such rules, it will fall to ICANN to make these decisions on a case by case basis. This would for the most part take place in an evaluation phase, except in cases where requests were being considered to change the status for a particular variant label. In these cases, resources and standards for considering the requests would need to be developed, raising the issues of:

a) Who is/are the appropriate parties for making determinations on such requests, and

b) What standards should be used for considering such requests.

When variant TLD labels are assigned to particular states, it will additionally be ICANN's responsibility to maintain records concerning the set of names in these various states. The database of delegated top-level domains (i.e., the root zone database) is maintained and made publicly available via the IANA function. To the extent that variant TLD labels are "blocked," "withheld," "allocated," etc. lists of such labels would need to be maintained and updated, along with clearly documented procedures for the circumstances, if any, where states for these names may be changed.

5.3.3 Delegation of Variant TLDs

To the extent that requests for delegation of variant TLDs are approved, this would take place following the existing IANA delegation procedures. This would include maintaining and publishing registration data for new TLDs, and distribution of updated zone file data according to current procedures. Delegation requirements might need to be adjusted to, for example, synchronizes changes to TLD records such that variant TLDs are updated at the same time, or ensure that re-delegation occurs at the same time. There may need to be an updated record format or new fields in the database to account for the association of one or more TLDs as variant TLDs.

5.3.4 Contractual Provisioning

Where a request including variant TLD labels is approved, ICANN may enter into an agreement with the relevant registry operator. Certain frameworks are available for ccTLDs, while gTLDs are expected to enter into a standard registry agreement with ICANN. These mechanisms can continue to be used for cases where there are variant TLD labels. A set of issues to be resolved, however, concerns the requirements to be followed by operators of variant TLDs. These could include:

a. Whether there should be specific reporting requirements concerning the variant TLDs, and if so, what data should be reported.

b. Whether specialized technical requirements for the management of the variant TLDs are necessary to support the security and stability of the DNS, and if so, how these requirements are specified

c. Whether specialized policy requirements for the variant TLDs are necessary to support a good user experience, and if so, how these requirements are specified.

d. Whether specific service-level requirements for performance of the variant TLDs should be instituted, and if so, how these requirements are specified.

e. Whether the registry fee structure should be adjusted to take into account the existence of the variant TLDs, and if so, how this should occur.

The ability of the registry to require and enforce parallel registration operation further down the tree could be of concern.

5.3.5 Security and Stability of the DNS

Security considerations are relevant in making the determination of which code points are valid and should be used in top-level labels, as well as the criteria for considering requests for IDN variant TLD labels. Similarly, such considerations are relevant in the determination of rules for assigning variant labels to particular states, and for the management of IDN variant TLD labels. These have been discussed in the relevant sections above.

Additionally, there has been significant study, consultation, and analysis in connection with expansion of the root zone to include new top-level domains. Modeling, monitoring, and reporting will continue during, and after, the first application round of the New gTLD Program, so that root-scaling discussions can continue and the delegation rates can be managed. To the extent that variant TLDs are delegated, these would be incorporated in this modeling, monitoring and reporting.

Delegation of any new top-level domains is conditional on the continued absence of significant negative impact on the security and stability of the DNS and the root zone system (including the process for delegating TLDs in the root zone).

5.4 *Registry / Registrar Operations*

Should a variant management mechanism for the top level be adopted by ICANN, this would entail a number of operational issues for registry operators as well as for registrars operating for the relevant TLDs. Although not necessarily tied to ICANN's management of the root zone, these are issues that would be expected to be resolved the ICANN community. These issues are discussed in this section.

5.4.1 DNS Resolution

Depending on the variant management mechanism implemented, delegation of variant TLDs may mean the TLD operator is required to invest more resources in zone file generation and management of registrations in the variant TLDs. This could also propagate to secondary name services and therefore increase the cost of running the DNS services for the registry. The maximum investment is possible to describe by imagining that the variant management mechanism is implemented via mirroring without any sort of aliasing. In this case, the TLD operator must in effect operate exactly as many zones as variants delegated, with a combinatorial number of labels in each such zone.

To the extent that an aliasing behavior is desired or implemented in TLD registries, this will have an effect on TLD registry operations. However, policies regarding DNS behavior could be difficult to enforce beyond the level in the DNS hierarchy at which the policy is defined. Specifically, a registry may choose to establish a policy wherein all possible variant labels will behave the same (return the same response in the DNS) at the TLD level of the DNS hierarchy. Although this can work in many cases at the TLD level, the DNS cannot enforce this policy on the delegated second-level domain names in the TLD. This can have a dramatic effect on the user experience.

In the event some sort of active variants were to be supported in the root zone by using DNAME, the root name servers would need to be able to support the additional load represented by CNAME synthesis due to requests from DNAME-incapable DNS clients. Given the current provisioning and traffic mix in the DNS, and laboratory tests, this additional load does not appear to be a concern compared to some of the other effects of supporting variants in the root. This belief has not been (as far as can be determined) verified empirically on the Internet. Adding a DNAME to the root would undoubtedly be an innovation, akin to the decision to sign the root using DNSSEC (though with perhaps fewer risks).

5.4.2 Registration Process

The shared registration system (SRS) is a critical registry function enabling multiple registrars to provide domain name registration services in the TLD. This in many cases includes the EPP (Extensible Provisioning Protocol) interface between the registry and the registrars. Extensions to EPP may be required to enable registration of second-level domain names under the applied-for TLD and the variant TLDs. Without a standard implementation of such extensions, registrars may face complexities in interfacing with these registries implementing different extensions.

Although not a variant issue per se but a general IDN issue, indications of the relevant script(s) and language(s) for registered domain names may need to be incorporated by the TLD registry. As noted in the Devanagari case study report, it may be sufficient in some cases to tag a domain name with either its script or its language; however, a script may support a number of languages and in some cases, a language uses more than one script. Moreover, as section 4.1 describes, it is possible to construct a system in which neither language nor script is the identifier needed. The technical issue is that there is no uniform way to do this in the standard EPP protocol used.

This issue also affects registrars in two ways. To the extent there is no standard, a registrar will have to implement all EPP extensions that various registries may choose to specify to resolve this issue. For those ccTLDs that do not use EPP, registrars will have to implement whatever is required in order to support that ccTLD.

In addition, when registrars are present they are the interface to the registrant. Registrars that choose to support multiple scripts and languages will need to develop user interfaces that facilitate and simplify the identification of the script and language in use by a registrant, and permit the registrant to understand its choices with respect to the names it is actually contracting to operate when registering a name subject to variants.

Accordingly, it appears that a successful registration process in IDN variant TLDs will require significant coordination, perhaps including an additional OT&E process, with registrars.

It will be the task of the registry operator to formulate policies on how domain names are managed in the variant TLDs. For example, policies could cover whether the same domain name under the variant TLDs must be associated with the same registrant. Consideration would also need to be given to registry policies on expiration, deletion, and transfer of registered names. It will also be for the registry operator to determine the appropriate pricing models for such registration offerings. The relevant policies concerning registrations in the variant TLDs, as well as the relevant IDN tables or other reference documents used for domain name registration at the second or lower levels, should be made available to the public. A failure of consistency in these policies across registries could have disastrous effects on user expectations; see section 5.3.2 for more discussion.

Careful attention to registration policies for IDN variant TLDs is essential to minimize user confusion and opportunities for abusive registrations. It is expected that the IDN Implementation Guidelines will be followed in this regard. The Guidelines are a list of general standards for IDN registration policies and practices that are designed to minimize the risk of cybersquatting and consumer confusion, and respect the interests of local languages and character sets. Registries seeking to deploy IDNs under their agreements with ICANN have been authorized to do so on the basis of the Guidelines. The Guidelines are, as of this writing, silent on the practice of the management of variants.

5.4.3 Whois (Domain Name Registration Data Directory Service)

As discussed in several case study reports, there are two sets of issues related to IDN variant TLDs and Whois: the first set of Whois issues are caused by the introduction of IDNs in general, the second set of issues are caused specifically by variants. Both sets of issues need to be addressed to ensure a good and consistent user experience for querying domain name registration data.

5.4.3.1 Issues that IDN introduces to Whois services in general

As noted in several reports (Arabic, Chinese, Devanagari), the critical Whois issue facing the deployment of IDNs is the fact that the standard WHOIS protocol (as defined by RFC 3912) has not been internationalized, which means there is no standard way to indicate the character encoding in use.

The WHOIS protocol is a simple request and response transaction: a domain name is submitted to a server and output is returned. A consequence of the lack of a standardized approach to internationalization is an increasing number of local, regional, and proprietary solutions that attempt to address the issue. This variability has a dramatically adverse effect on the user experience.

As the adoption of IDNs becomes more prevalent, Internet users will expect to be able to register domain names as well as registrant names and addresses in their native languages, using familiar scripts (character sets). This adoption is already well underway, increasing the priority to address this issue.

5.3.4.2 Issues that IDN variants introduce to Whois services specifically

Also noted in several reports (Arabic, Chinese, Devanagari), the introduction of variant TLDs may cause a paradigm shift for some Whois users. Where currently there is typically a one-to-one lookup for a Whois record against a domain name, this might no longer be true in the case of variant TLDs. To illustrate this point further, consider the following domain label3.label2.label1., where each label also has p, q, and r variants respectively, thus the total number of variants for this domain is p*q*r, and each possible variant domain could have different statuses (e.g. reserved, allocated, delegated, blocked, etc.).

The key issue here is to determine the correct response to a WHOIS query for a domain name lLabel3i.label2j.label1k. (request to information for label1k in the case of the root).

These issues require careful examination to determine to what extent data elements should be separated in the Whois database, and to what extent certain elements must always be subject to a joined relationship. Specific issues to be worked out include:

a. For a Whois query for a domain name with variant labels, should the variant labels be included in the response? What if the language or script of the variants cannot be understood or displayed by the user making the request? How could this be determined, since the WHOIS protocol does not have a mechanism to signal encoding?

b. If a U-label is reserved in the registry database but is not present in the DNS, should a Whois request for the domain name return a referral indicating the name is a variant of another name or return the response for the other name? or Should the response indicate that the name does not exist?

c. Is there a need for an additional query/service, which returns the Label Variant Set against a requested domain name? Should such a service also return the status of each label in the set?

d. Would the response against a blocked variant label be different from responses to labels with other status (reserved, allocated, etc.)?

e. Will the creation and expiration dates of the Variant Label Set be inherited from the fundamental label, as suggested? If yes, then if a variant label is either added or changes its state, how will this information be part of the Domain Name Registration Data? Would history be maintained and communicated for such changes?

5.4.4 Data Escrow

As noted by the Chinese case study team, a specific data format has been specified in the gTLD Applicant Guidebook for registries to submit the registration data to the data escrow service provider. The data escrow format currently supports variants; however modifications may be needed in light of the issues detailed in this report to support variant TLDs.

5.4.5 Rights Protection Mechanisms

The use of variant TLD labels (and variant domain names in these TLDs) may have an impact on existing rights protection mechanisms such as the UDRP. The UDRP is a policy for resolving disputes arising from alleged abusive registrations of domain names (for example, cybersquatting), allowing expedited administrative proceedings that a trademark rights holder initiates by filing a complaint with an approved dispute resolution service provider. Typically, a UDRP complaint will concern only the name registered at the second level. In the case where domain names exist in variant TLDs, an issue to be resolved concerns whether corresponding names in all TLDs in a variant set are considered together in the event of a UDRP proceeding, or whether and under what circumstances there could be separate consideration and determinations on some names in the set. Where certain names have different statuses (e.g., blocked, allocated), this could also have an impact on such a case. The applicable fees in UDRP proceedings might also be adjusted to take into account the existence of registrations in variant TLDs.

A policy whereby a set of variant names is always maintained may force registrants into dispute resolution proceedings as a result of registry variant practice (such as automated generation of a variant label set). However, a practice of splitting and making separate determinations within the set runs counter to the interest in maintaining a straightforward and predictable process, particularly since the existence of variant labels suggests that the contents of the set are equivalent in some way.

Note also that a UDRP complainant will present trademark or rights data that does not necessarily follow the same label generation rules that are used for the root zone or by the registry. A complainant must prove that the domain name in question is identical or confusingly similar to a trademark or service mark in which the complainant has rights. Although a panel might make use of existing variant policy references in considering a case, it is more likely that a panel would refer to standards of trademark law for determining what is "identical or confusingly similar," and such standards would be an important consideration relating to issues in this area.

A determination in a UDRP case may result in cancellation or transfer of a domain name, or no action if the complainant does not prevail. Issues to be resolved in this context concern whether a determination would also need to apply to the entire variant set, or whether could there be a split decision concerning the treatment of names (e.g., a decision that of the set example.X, example.Y, and example.Z, only labels example.X and example.Y infringe the complainant's rights while example.Z does not). There are wide implications here given that a legal determination could create an exception to the established label generation rules

In addition, the existence of variant TLDs may have an impact on new rights protection mechanisms being instituted as part of the New gTLD Program. New gTLD registries are required to introduce certain rights protection mechanisms during their startup phases. These services concern second-level registrations, and include a sunrise period and a trademark claims service to provide notice to a potential registrant where a domain name matches a trademark that has been recorded in the Trademark Clearinghouse. These services use a specified definition of identical match. To the extent that registrations take place in variant TLDs according to the registry policy, these rights protection processes should take into account the existence of variant TLD labels. For example, if a trademark holder is eligible to register example.X in the sunrise period, they could also be eligible to register example.Y (or ineligible, if registry policy blocks such registrations). Consideration would need to be given to the definition of identical match and the possible incorporation of variants into the criteria for triggering a Sunrise or Trademark Claims notice.

The Uniform Rapid Suspension (URS) system is a complement to the UDRP, to be used when suspension of a domain name is the desired outcome. Accordingly, the issues discussed above also apply here.

The (Trademark) Post-Delegate Dispute Resolution Policy (PDDRP) addresses infringing uses of the TLD post-delegation. However, outcomes under this policy would take the form of remedial measures to ensure against future infringing registrations, suspension of new domain name registrations, or, in extraordinary circumstances, termination of the Registry Agreement. Since domain name registrants are not a party to the action, a recommended remedy would typically not take the form of deleting, transferring, or suspending domain name registrations. In the case of variant TLDs, the issues to be considered here would concern whether the penalties could apply to just one of the variant TLDs, or would necessarily apply to all.

5.4.6 Security/Stability Considerations for TLD Registries/Registrars

Security and stability considerations should also be relevant to registry operators, registrars and data escrow agents or other service providers. As noted in the Devanagari case study report, a suggestion for evaluating variant policies and their implementation is to log, review, and analyze DNS query traffic. Specifically, the behavior of applications and services, and sometimes the users that use them, can be inferred from traffic patterns found in sequences of DNS queries and responses. For example, registries could review DNS traffic of the TLD for queries of non-existent domains (i.e., in DNS terms reviewing the NXDOMAIN responses). An analysis of these transactions may indicate that language tables are incomplete or that variant usage is not as expected. Conversely, an analysis of the queries that indicated that certain type of variant is not being queried (while the fundamental label or other type of variant is) could indicate a superfluous variant category.

Providing a consistent, uniform, and non-surprising (i.e., user expected) experience to the user is an essential component of stability. DNS transaction logs could provide some insight into user expectations and thus some ability to confirm that the needs of a user community are being met.

Some TLD registries may wish to consider partnerships with second-level domain holders to continue the analysis at lower levels in the DNS hierarchy.

As is clear from this section, there are a number of complex issues in the operational area that registries, registrars, and other providers should be aware of to facilitate successful operation of variant TLDs. As noted by the Latin case study team and others, the impact of variant TLDs on registries and registrars may be highly dependent upon differing implementation methods, and any proposed implementation will require broad stakeholder participation to ensure that registries and registrars provide stable, secure, consistent, and unambiguous DNS operations. This includes the greatest possible clarity in communication and understanding of variant TLDs, to limit IDN end user, registrant, registrar, and registry confusion.

Areas of application behavior, resolution and registration services, WHOIS service, and business logic all need to be examined in order to determine if these objectives are achievable.

6 Other Related Issues: Code points currently not permitted in the root zone

The gTLD Applicant Guidebook currently restricts U-labels for inclusion in the root zone to those in Unicode General Categories Ll, Lo, Lm, or Mn; this is anticipated to be expanded to include Mc, in order to support a number of letters that are necessary to permit Devanagari strings. In any case, the general principle for inclusion in the root zone is that only letters be permitted; this principle is an attempt to stick to the spirit of RFC 1123, which says that labels in the root zone “will be alphabetic”. At the time of writing of RFC 1123, “alphabetic” really just meant the letters a-z and A-Z, because nothing outside the US-ASCII character set was contemplated.

The current restrictions prohibit a number of code points that are (in at least some circumstances) permitted under IDNA2008, and might be desirable for writing strings in some languages. Prominent among these are U+200C, ZERO WIDTH NON-JOINER, and U+200D, ZERO WIDTH JOINER (both of which are in General Category Cf). Also prohibited are all digits, including digits used with scripts other than Latin (at the very least, those code points in General Category Nd). It is beyond the scope of the present report to evaluate whether the root zone ought to permit U-labels with these code points, but each of these categories raises issues with respect to variants. This section provides a very brief discussion of the issues that might be raised by changing these rules. The discussion here is incomplete because the prohibitions remain in place; but if those prohibitions were lifted, it would entail a great deal of work to explore these issues completely and to understand the potential side effects.

The zero-width joiners are two special controls that are used to indicate connections or disconnections within a string when the string is rendered; they are normally used only in scripts that require complex text layout (such as Arabic and Indic scripts). They are called “zero-width” because they take no space when displayed, and cannot be seen under normal conditions. Their effects, however, can sometimes be observed in the shape of characters that follow them. (In some contexts, however, like in a Latin string, the effect of the zero-width joiners is entirely invisible.)

ZERO WIDTH NON-JOINER, U+200C (“ZWNJ”) is used to prevent rendering of a connection between two characters (usually letters) that otherwise would be joined. It is often used in Arabic-script writing in order to prevent two letters from joining to one another (and thus using the medial form) when they should not be so joined (for instance, because there is a word break at that location). ZWNJ is already constrained in its use in U-labels by a CONTEXTJ rule in IDNA2008; the Arabic case study report, however notes, “The ZWNJ in a few cases is still not visible to all users (e.g., U+0637, U+0638, U+069F, U+06BE, and U+06FF). A comprehensive analysis of Unicode Arabic Script Code Charts is needed to find any additional cases. This process should be repeated as the Unicode gets updated." (item 21.b.iii, p7) One can glean from this that Label Generation Rules permitting ZWNJ in TLD labels would probably need a fairly complicated contextual rule constraining ZWNJ to very specific cases, and likely generating variant labels that also excluded the ZWNJ, in order to ensure that a collision could not happen with such a ZWNJ-free label. Note that the Devanagari case study (4.3.2, p 23) makes it clear that generating such rules could be extremely difficult, because the limits do not appear to be bounded.

ZERO WIDTH JOINER, U+200D ("ZWJ") is used in the opposite case as ZWNJ: it makes two characters join together when normally they would *not*. The Devanagari case study report provides an example (section 4.3.1, p 22) of how ZWJ might be used. It is worth noting that Unicode NFC does not normalize these two different ways of entering the same abstract character, so U-labels containing these two different sets of code points would not be equivalent to one another. It is entirely possible that every use of ZWJ in a U-label would produce variants of this sort, and it would require a great deal of language-specific study for those languages using ZWJ to specify what variant relationships should be included in any Label Generation Rules. Just as in the case of ZWNJ, it is likely that variants excluding the ZWJ would probably also be necessary, in order to prevent collisions with a label that could be typed only by omitting an invisible character.

The digit cases are also somewhat problematic. To begin with, there are two sets of Arabic-Indic Digits (known as Arabic-Indic Digits and Extended Arabic-Indic Digits). Because of the ways these appear, they are prohibited (by an IDNA2008 CONTEXTO rule) from appearing in the same label; apart from that, any type of digit could be combined with any other type in a label. In this case, the issue is that applications will sometimes, in a locale-sensitive way, convert localized forms of digits to their standard "ASCII range" counterparts (i.e.U+0030..0039) for storage, and back to localized versions for the purposes of display. This ensures that arithmetic operations always work consistently (among other advantages); the principle sometimes goes by the slogan "a digit is a digit." If the prohibition against code points in General Category Nd were to be lifted, then a large number of variants could result across all the scripts permitted in the root zone.

7 Discussion of Potential Additional Work

Additional work concerning IDN variant TLDs could concern taking the issues identified in this issues report, and identifying potential solutions to these problems. Any potential solutions identified would need to be analyzed to establish whether they satisfactorily address the sets of issues identified here, and measured on their viability from policy, technical and security perspectives.

It should be emphasized that there should be no assumption that acceptable solutions are available or can be developed to address the wide range of issues identified in this report, or that can satisfy the wide range of community expectations regarding which variant IDN TLDs may be delegated.

In analyzing the issues presented above, the team identified several areas where additional work could occur toward developing solutions. The follow-on phase to this report could be a short phase involving review of these items and development of a plan (including a proposed timeline and budget) for each element. Some possible actions in the various areas are identified here for consideration.

7.1 Developing a Label Generation Ruleset specification

Based on the analysis, a general requirement for all approaches considered is the need to use a tool to machine-generate sets of variants in accordance with a formal label generation ruleset.

ICANN currently manages a voluntary repository of "IDN Tables," of which some contain instructions on computing variants. While some language communities have formalized the formatting of their tables, there is no single established format that can accommodate the various rulesets in existence today.

Recognizing that deployable solutions will require such tables, it is clear that the effort would benefit from the standardization of a table format that would allow software implementers to easily and predictably generate variants. Such a table format should be developed with input from potential implementers and other interested parties, possibly through a technical standards body such as the IETF. In conjunction with this work, ICANN could facilitate a reference implementation of software that demonstrates how the table format could be utilized. Such work could be used internally within ICANN for its processes when handling variants for the root zone.

7.2 Developing a process for label generation

Considering the potential options presented in Section 4.2, an analysis of how labels could be generated for the root zone needs to be conducted to arrive at a suitable approach. This needs to be developed in a way that accords with ICANN’s established processes.

7.3 Examining the feasibility of whole-string variants

While the scope of the work of this group was character-level variants, for some languages, there may be deterministic approaches available to calculating and deriving sets of whole string variants. The report has identified significant challenges to this approach, but there is more work to be done to analyze potential approaches in this area if whole-string variants are considered a relevant area of study.

7.4 Enhancing visual similarity processes

For some script communities, it appears possible to deterministically identify variants that are considered visually similar. This would potentially act as valuable input into existing visual similarity processes, which would benefit from predictable and repeatable processes that are transparent to a user of the process.

7.5 Examining the feasibility of mirroring

Section 5 identified a number of potential treatments of particular variant labels. One potential treatment is the use of "mirroring," whereby two or more labels use a technology (currently a choice between CNAME and DNAME) to ensure they provide the same result in the Domain Name System. Due to the distributed nature of the DNS, using these technologies is a complex challenge, particularly as ensuring appropriate software support for products which rely on the DNS but do not have proper understanding of the many-to-one domain name relationship that mirroring creates.

Another alternative to using specialized DNS records is to use “parallel provisioning,” whereby regular delegations are made using NS records, and the manager of the zone is obligated via contractual or other means to ensure the contents of those zones are synchronized.

The team recommends studying the feasibility of mirroring, in cooperation with technical bodies as necessary.

7.6 Examining the feasibility of packaging labels for unified management

Another potential treatment is to consider two or more labels, with their management conducted by the same party (i.e. a single registry); but with no requirement that the contents of the zones and the delegations therein be strictly mirrored. This treatment recognizes potential legitimate reasons for the contents of the various zones to diverge, while still ensuring that the risk of user confusion is minimized by having one party manage eligibility and delegation policy between these multiple zones. This allows contextual rules to be established that address local requirements concerning confusability, without the exactness of mirroring.

These Labels with a Unified Management Process ("LUMP"), could be studied to identify the feasibility of this approach as a method of managing variant delegations.

7.7 Assessing impacts on existing operations

Recognizing the output of this process is primarily to provide for variants to be delegated as part of the new gTLD process, the impact of the variant solutions would need to be considered with respect to the new gTLD process. Specifically, the label generation process discussed above would need to be considered with respect to updates to the new gTLD process.

7.8 Assessing impacts on the ICANN and IANA process

As manager of the contents of the DNS Root Zone, ICANN would need appropriate processes to implement the requirements of any variant solutions to be adopted. This would include adaptation of the relevant IANA processes to incorporate the concept of variants, and ensuring the equivalence is reflected in its operations. This could involve ensuring the root zone changes are executed in tandem when variants are involved, and ensuring that any labels which are either mirrored or LUMPed are always assigned to the same manager. The IANA WHOIS and website would also likely need to be updated to demonstrate any linked relationship between multiple labels, and the status of various labels that have been generated through the application of the Label Generation Rules.

Appendix 1: The Case Study Team Reports

Arabic

Case Study Team Issues Report:
http://www.icann.org/en/topics/new-gtlds/arabic-vip-issues-report-07oct11-en.pdf

Public Comments:
http://forum.icann.org/lists/idn-vip-arabic/

Summary and Analysis of Comments:
http://forum.icann.org/lists/idn-vip-arabic/

Chinese

Case Study Team Issues Report:
http://www.icann.org/en/topics/new-gtlds/chinese-vip-issues-report-03oct11-en.pdf

Public Comments:
http://forum.icann.org/lists/idn-vip-chinese/

Summary and Analysis of Comments:
http://www.icann.org/en/public-comment/report-comments-idn-vip-chinese-29nov11-en.pdf

Cyrillic

Case Study Team Issues Report:
http://www.icann.org/en/public-comment/idn-vip-cyrillic-06oct11-en.htm

Public Comments:
http://forum.icann.org/lists/idn-vip-cyrillic/

Summary and Analysis of Comments:
http://www.icann.org/en/public-comment/report-comments-idn-vip-cyrillic-29nov11-en.pdf

Devanagari

Case Study Team Issues Report:
http://www.icann.org/en/topics/new-gtlds/devanagari-vip-issues-report-03oct11-en.pdf

Public Comments:
http://forum.icann.org/lists/idn-vip-devanagari/

Summary and Analysis of Comments:
http://www.icann.org/en/public-comment/report-comments-idn-vip-devanagari-29nov11-en.pdf

Greek

Case Study Team Issues Report:
http://www.icann.org/en/topics/new-gtlds/greek-vip-issues-report-07oct11-en.pdf

Public Comments:
http://forum.icann.org/lists/idn-vip-greek/

Summary and Analysis of Comments:
http://www.icann.org/en/public-comment/report-comments-idn-vip-greek-29nov11-en.pdf

Latin

Case Study Team Issues Report:
http://www.icann.org/en/topics/new-gtlds/latin-vip-issues-report-07oct11-en.pdf

Public Comments:
http://forum.icann.org/lists/idn-vip-latin/

Summary and Analysis of Comments:
http://www.icann.org/en/public-comment/report-comments-idn-vip-latin-29nov11-en.pdf

Appendix 2: Salient Characteristics of 6 scripts

Arabic

The Arabic script is a distinctively cursive form of the Phoenician script, transmitted to the Arabs through Aramaic-using intermediaries, and subsequently spread massively, first round the Arab empire, and latterly to many other parts of the world where Islam was accepted, over the millennium from the 7th to the 17th centuries AD.

The style of cursive script also varies from region to region, altering the shape, orientation and relative size of some glyphs. But these differences are not represented in Unicode. There is, however, a character (tatwīl - U+0640) which simply represents a stylistic lengthening of the horizontal distance between glyphs.

Like many scripts for Semitic languages (notably, Phoenician, Aramaic and Hebrew) it is an *abjad*, a script where short vowels are usually omitted from representation, as well as doubling of consonants. (There is a system of diacritic signs to represent them, applied optionally. More diacritic sign are available to note consonant doubling, consonant clusters, and some morphological processes.) Arabic, like these other Semitic languages, is written right-to-left.

There is no distinction of upper from lower case. However, every character has up to four glyphs whose use is dependent on the character's position in a word (initial, medial, final or isolated). (Recommended use of Unicode is to employ a single code point to represent each character, and leave the selection of glyphs to the rendering engine; however, Unicode does retain deprecated code points (U+FB50-FBFF) which represent many such glyph directly.) This positional dependence of glyph selection is also seen in other Semitic scripts (e.g. Hebrew square characters, Syriac estrangelo): but in these, Unicode recognizes the different glyphs as distinct code points.

One characteristic of Arabic script is that many of the glyphs representing one character are identical, or extremely similar, to glyphs representing another. They are therefore distinguished by a system of dots (one, two or three) and, in extensions used for languages beyond Arabic, a few other diacritic marks. Unlike the vowel diacritics, use of these dots etc. is not optional. There is regional and linguistic variation in the use of these obligatory diacritics, and these have provided the Arabic script's principal means of extending its repertoire to cover sounds foreign to Arabic. All these obligatory dotted or diacritic-bearing forms are available as pre-composed Unicode code points, and they are not composable alternatively through rendering engines.

There are officially 28 characters in the Arabic abjad, though this does not include pre-composed forms of vowel-carriers with hamza (the glottal stop) nor the vowel diacritics: with these, the total increases to well over 40; other language's character-sets will often exceed this number. Unicode sometimes assigns different code points to characters which – regionally – have somewhat different combinatory forms. In some instances (e.g. the so-called swash kaf ڪ U+06AA) what is a stylistic variant in Arabic (though assigned a separate code point) is used as a distinct character in other languages (e.g Sindhi).

There is one conjunct glyph [لا] representing lām [ل] + alif [ا]. It is not available pre-composed in Unicode's Arabic code block, but deprecated forms of it, alone and with various diacritics, survive at U+FE5 to U+FEFC. Rendering of groups of Arabic letters may result in extremely compact glyphs, e.g. stacked vertically.

Chinese

The Chinese script is a system of several thousand characters, each representing a determinate phonological syllable, and – since such is the nature of Chinese – equally representing the meaning of that syllable. It is possible for a Chinese syllable to have different meanings: in such cases, each character will represent its meaning unambiguously, and will not be accepted as a variant for other characters with the same sound. Dialectally, also, the phonetic value of a character may change, but not its meaning: e.g. when the characters are used to represent a Chinese dialect not mutually intelligible with the standard Mandarin.

Although the Japanese language uses fewer characters (by a factor of 3 to 4), its use of the system is in some ways more complicated than that made by Chinese: although each character still has a single distinctive meaning (shared with the Chinese), it typically has two readings, a short (monosyllabic or di-syllabic) one, which is historically derived from the Chinese pronunciation of the character, and used in loan vocabulary, and a polysyllabic one which represents a Japanese word-stem, totally unrelated phonetically, but which happens to have the same meaning. The actual text of Japanese is also more complicated in that it also contains phonetic symbols (from the *kana* syllabaries), which serve to fill out the specific morphology of Japanese language, as well as specifying the pronunciation of any words not represented by characters.

The Koreans and Vietnamese have historically made heavy use of Chinese characters, each finding distinctive means to adapt them for the representation of their own languages, but now no longer do so.

Chinese and Japanese were traditionally written in vertical columns, the columns being read in sequence from right to left across the page. Nowadays both languages are almost exclusively written horizontally left to right, like most European and Indian languages. The orientation of characters on the page is not affected by these gross differences of character lay-out. However, both languages write text without any breaks between words. They have many distinctive punctuation marks: notably, [、] U+3001 which corresponds to comma [,] and [。] U+3002 which corresponds to period [.]. But all western punctuation marks may also be found used in text.

This script does not allow positional variants of glyphs (whether due to code point difference or rendering), nor does it use diacritic symbols.

Cyrillic

The Cyrillic alphabet, named for St Cyril, is historically derived by Orthodox missionaries from Greek ca. 10th century AD, with some additional glyphs drawn from Hebrew. Its use spread from the Balkans with the Church Slavonic language to the Ukraine and Russia in the 12th century. It was subject to radical spelling reforms in the 20th century under the Soviet Union, which rationalized its use in the spelling of Russian and other modern Slavonic languages, as well as extending its repertory of glyphs to represent many unrelated languages spoken within the Union and Mongolia. Some Turkic languages, spoken in the newly independent states of the former Soviet Union switched their orthography to Latin in the early 21st century.

Cyrillic has more glyphs than Latin, for most languages over 30: Russian uses 32, Ukrainian 34. Some use many more (notably Abkhaz with 62). Digraphs and trigraphs are not used, and although there is some use of diacritics (й, ё) they are less frequent in Cyrillic-using languages than in those that use Latin. More often, the form of a glyph is supplemented somewhat to represent distinct phones. Such extensions of the Cyrillic repertoire are coded by Unicode in a single block with the rest of Unicode. There are in general no conjunct consonants.

In every other important respect (e.g. left-to-right direction, upper and lower case, lack of positional variants, word separation) Cyrillic is like Latin. However there is no such simple arithmetic relation between upper- and lower-case glyphs as holds for Latin code points.

Devanagari

Dēva-nāgarī (Sanskrit for “divine urbane”) is the most widely used exponent of the Brahmi family of scripts, first known to moderns from the emperor Ashoka’s inscriptions (dispersed about India) of the 3rd century BC, though regionally it is largely restricted to northern India and Nepal. It is conjectured to be a development of the Aramaic script which diffused through Iran during the time of the Persian empire (7th-4th centuries BC).

It is formally classed as an **abugida** (a term borrowed from the Amharic language of Ethiopia, whose script is also of this class). Such a script is written with symbols each one of which (barring exceptions) represents a syllable, but which can be modified by diacritics and accompanying glyphs to represent a syllable with a different vowel. Such a syllabic unit is called in Sanskrit **akshara** (‘imperishable’).

Devanagari has symbols for 12 vowels [a, ā, i, ī, u, ū, ṛ, ḷ, e, ai, o, au] 25 occlusive stop consonants [k, kh, g, gh, ŋ; c, ch, j, jh, ñ; ṭ, ṭh, ḍ, ḍh, ṇ; t, th, d, dh, n; p, ph, b, bh, m], four glides [y,r,l,v], three sibilants [s, ś, sh] , and two glottal fricatives [h, ḥ]. Each of these, without further mark, is pronounced with a short [a], [ʌ] or [ə] vowel immediately following the consonant. There is also a nasalization mark. Among the occlusive stops, each point of articulation is represented with a voiceless plain stop, a voiceless aspirate, a voiced plain stop, and voiced aspirate and a nasal. E.g., velar articulation is represented by ka, kha, ga, gha and ŋa. Although it did not originally contain symbols for non-dental fricatives, the unvoiced aspirates (kha, pha) can be marked with nukta (a subscript dot) to represent (fa, xa). Voiced za is indicated by adding a nukta to ja and uvular qa by adding a nukta to k; and in general the nukta has been used to create new symbols to represent sounds unforeseen in the basic scheme of Devanagari.

From the viewpoint of glyph-incidence, the vowels may be considered to have positional variants: when they occur syllable-initially they are represented as independent aksharas; and when they occur after a consonant, they appear as diacritic signs (called matras) attached to that consonant's akshara. However, from Unicode's viewpoint, the syllable-initial and syllable-final vowels are unrelated, being assigned distinct code points. Rendering rules should ensure the proper attachment of the syllable-final vowel glyphs to their leading consonants.

The general form of an akshara is a sequence of 0-5 consonants, followed by a vowel, which is oral or nasal. If there is more than 1 consonant, the final consonant's akshara is used as the basic form, and to it are added various truncated forms of the aksharas representing the preceding consonants: e.g. for the akshara [rtsnyā], truncated forms of ra, ta, sa, and na are added left-to-right to the yā akshara, which is itself ya with added vowel diacritic for ā. If the vowel is nasal, the nasalization mark is added to the whole.

This process of conjoining consonants (by combining the truncated forms) was an immense problem for typographers, which carries over into Unicode rendering. Rendering engines usually start from a sequence of code-points with the virāma sign (aka halant) [U+094D] interspersed: if there is a precomposed conjoined glyph available it is generated, otherwise the different consonant aksharas appear in full, those lacking vowel marked with the diacritic [◌्]. This latter output, with explicit [◌्], can also be compelled by appending ZWNJ [U+200C] to a consonant+virama sequence of code-points. In some cases, the truncated form of the aksharas can be generated independently by appending ZWJ [U+200D] to a consonant+virama sequence of code-points.

The diacritic [◌्] is also used to represent word-final consonants.

Devanagari is written from left to right. As used for modern languages, it leaves blank space between words, although the traditional usage was to write the strings of words in a sentence undivided, breaking only at sentence end. In this context, it is relevant that he complicated sandhi (liaison) rules that apply in some languages written with Devanagari (notably Sanskrit) often apply across word-boundaries and may merge words. Sentence breaks are marked with a danda (।) or double danda (॥), the latter usually only in verse. In modern languages, the query [?] is also used as a punctuation mark.

Greek

The Greek alphabet was derived from the Phoenician script of the eastern Mediterranean in the 8th century BC. It diverged from it principally in systematically distinguishing vowels from consonants (Phoenician having symbolized consonants and in some cases long vowels – though re-using consonant symbols for this). Greek was the first alphabet to note all vowels explicitly. Early Greek had different standards, notably the Euboean, the Corinthian and the Attic, affecting the interpretation of some glyphs. Greek colonists spread the alphabet in different standards all around the Mediterranean, where they were adapted and extended to various languages (including Coptic Egyptian) and as far as Central Asia (for Bactrian), but after 403 BC only the Attic standard was used for the Greek language itself. Direction of writing (after much early fluctuation) stabilized to left-right.

Greek in the Attic standard has 24 letters, each now with an upper- and lower-case glyph. It uses digraphs (μπ, ντ, γκ, γγ, γξ, γχ, αι, ει, οι, υι, αυ, ευ, ου) [(m)b, (n)d, (n)g, ŋg, ŋks, ŋx, e, i, i, i, af, ef, u] but no trigraphs. It has two clear conjunct consonants ξ for [ks] and ψ for [ps]; ζ too is sometimes said to be analysable as [ds]. In the early first millennium AD, an auxiliary system of accents (´ ˆ `) (one per word) and breathings (rough or smooth ‘ ’) (obligatory on any word-initial vowel) – e.g. ἁρμονίη ἀφανὴς φανερῆς κρείσσων [harmoníē aphanès phanerês kreíssōn] ‘A hidden harmony is stronger than an obvious one.’ - Heraclitus) was introduced, which – together with the system of upper- and lower-case - became obligatory. As an artefact of handwritten style, lower-case sigma at the end of a word came to be represented with a different glyph – e.g. επίσης [epísis].

As a result of pronunciation changes down the millennia, many Greek diphthongs have turned into digraphs (i.e. their pronunciation is no longer a combination of the component vowels) e.g. αι [e] - ει, οι, υι [i] - αυ [af] ευ [ef]. A diaeresis (**dialytiká**) mark also came in use to indicate that a high vowel (ι υ) after a vowel is in fact to have its usual pronunciation e.g. αϊ [ai] οϊ [oi] αϋ [ai].

This system of accents and breathings was however radically simplified in 1982, moving to **monotonikó** orthography. In this, breathings are abolished, and the only accent is the *tónos*, glyphically similar to the [´]. All accented variants of vowels bear this single symbol. Unicode has registered precomposed versions of all polytonikó and monotonikó vowel signs. Dialytiká vowels may also bear the tónos.

In every other important respect (e.g. left-to-right direction, upper and lower case, word separation) Greek is like Latin. However there is no such simple arithmetic relation between upper- and lower-case glyphs as holds for Latin code points.

Latin

The Latin alphabet was historically derived from the Greek script (in its Euboean standard, with different letter-values from the established Attic standard). This occurred in northern Italy ca. fifth century BC, very likely via Etruscan.

In its modern form, it has two sets of 26 glyphs in one-one correspondence, lower-case and upper-case. As an alphabetic script, it has glyphs for consonants and vowels. It is written from left to right; words are separated by blank spaces. Glyphs may occur as doublets (digraphs) and triplets (trigraphs), with specific phonetic values assigned by the language. There is one consonant widely used to express a conjunct, namely x (for [ks]).

Glyphs may also occur with diacritic marks, usually indicating different phonetic values for the glyph. In some instances, however, in some languages, diacritic marks are used to indicate the distinctive position of phonetic stress or accent within a word.

Sentences are marked off by succeeding punctuation marks, notably periods. Other marks used at the end of sentences show the degree of discourse connection to the following sentence (colons and semi-colons); or the discourse force of the preceding sentence (queries and exclamations). There are also marks (commas) which serve to mark major breaks in the structure of sentences. Hyphens may be used to mark compound-words, and apostrophes to mark the position of omissions within abbreviated expressions. Periods are also used to mark abbreviations, when only the initial (or the initial and final) parts of expressions are retained. Directed left-right pairs of marks (parentheses, brackets, single quotes, double quotes) mark the boundaries of sections of text for various purposes.

In printed form (as mostly used in printed and computing environments), the script is not cursive: furthermore, the forms of glyphs do not vary with their position in the word or their surrounding glyphs. By contrast, upper-case (aka capital) forms are used to replace the first letter of certain words, to signalize the whole word. Such words are said to be capitalized. Languages differ in their conventions for the use of capitals. Almost universally the first word of a sentence is capitalized. In English, capitalization marks the important status of proper names or titles, as well as for adjectives and nouns derived from such proper names. As examples of how other languages may differ from English, note that in German all substantives are capitalized, while in most European languages, such derived adjectives and nouns are not capitalized.

Unicode provides several pages of “extended Latin” code points, many of them showing regular glyphs deformed in some way. Besides the pre-composed glyphs of letters with diacritics, very few of these are current in the spelling of languages using Latin script. The majority use for such characters is by linguists to represent particular phones more exactly (e.g. in the International Phonetic Alphabet or the Americanist tradition).

Appendix 3: Overview of the Script Case Studies

The six case study reports identify relevant variant issues in six different scripts: Arabic, Chinese, Cyrillic, Devanagari, Greek and Latin.

Broadly the reports share a common structure, beginning with an introduction of the report's authors, and presentation of the history, distribution, and structure of the relevant script. The reports propose specific terminology to give details on the kinds of issues affecting their script which may suggest treatment by establishment of a variant management mechanism. Most of the reports include an exhaustive listing of the Unicode code points (with character names and glyphs) which make up their script (though not Chinese with over 5000 characters to organize) and which they recommend for use in TLD labels. There is then a review of the kinds of relations between characters (and, in a very few cases, words) which might be modelled with the mechanism of variants. The language-related and contrastive complexities of individual characters are later supplemented with issues concerning the ease of character recognition, and other user issues deriving from constraints on the software and hardware environment in which the script is currently used. There are, in some cases, examinations of technical issues not centrally focused on variants (e.g., the degree of acceptability of invisible code points, apostrophes and other characters which are not members of the code block associated with the script).

The focus then moves to the procedures for applying to register TLDs in the script, and the administrative apparatus which remains installed to support them. This last includes operational security concerns and procedures for dispute resolution.

Most of the reports (4 out of 6) also have a summary of conclusions, to emphasize certain of their theses.

Statements of General Principles

The focus of the work is the identification of issues concerning the potential use of variant characters within IDN scripts, to define variant top-level domains (TLDs), whether generic or country-code. The IDNA Protocol determines whether a code point is Protocol-Valid (PVALID) by derivation from certain Unicode properties. It was noted with caution that the teams of experts might not have expertise on every language using the scripts considered. (Arabic 3, Devanagari 1, Cyrillic 1, Latin 11).

Distinctive Terminologies

To define the field of **character variants**, the teams initiated their work around a basic set of definitions for terms as provided by ICANN.[30] These were supplemented by terms

[30] https://community.icann.org/download/attachments/16842778/Draft+Definitions.pdf?version=1&modificationDate=1310669168000

from the Unicode website, RFC 6365, and RFCs 5890, 5892 and 5893. (Arabic §4, Latin §2)

The reports also defined additional terms to address properties of their own scripts. All except the Devanagari and Latin teams found it necessary to define such terms. (Arabic Appendix E, Chinese 3, Cyrillic 2, Greek 2). The coordination team has worked to arrive at a set of generally agreed terminology for the IDN Variant Project.

Code Blocks in Extenso; Label Generation Policy

The IDNA Protocol has a very broad-based filter to determine what code points are permitted under the protocol; the rules are defined in RFC 3892.[31] Many teams considered or proposed further restrictions (e.g. to exclude free-standing diacritics), and listed the resulting subsets of code points which would be available for use in TLDs. (Arabic 5; Chinese 2.1; Greek Appendix A; Latin Appendix B2).

The Arabic report (7) notes that in addition to these lists of label-valid code points, a policy on defining character variants, and a set of other rules and meta-information must also be added, in order fully to identify the set of possible labels. Taken together, these constitute a Label Generation Policy. The Chinese report (7) covers similar ground on the full integration of character repertoire and variant linkages, but basing its discussion on the concept of a **Language Variant Table**, in keeping with RFC 3743.[32]Defining the Scope of Variants within a Script

This was the central concern of each case study team. Inevitably, at the outset of the project, it could not be precisely foreseen what may turn out to be the ultimately acceptable boundary conditions.

So the Arabic report (6) and the Devanagari report (3.2) distinguish cases of identical and similar glyphs (which might ultimately be seen as cases not of Variance but Visual Similarity). The Arabic report also notes the existence of interchangeable characters (where the basis for equivalence is linguistic functions) and optional cases (where the writing system allows some degree of choice in the exactness of a written form). The Devanagari report does not consider these latter as potential variants (4.2). The Cyrillic report (3) looks at a number of concrete issues in various Cyrillic-using languages, where additions and refinements to Cyrillic have created inconsistent usage. Specific examples of these types are listed in Appendices Arabic A and Cyrillic A.

[31] http://tools.rfc-editor.org/html/rfc5892

[32] http://www.rfc-editor.org/rfc/rfc3743.txt

The Greek report (6-9) also focuses on inconsistent orthographical practices (e.g. in use of upper/lower case and the *tonos* accent), but also (5) notes the need for a policy on the orthography of the letter *sigma*, which (by historic convention) requires positional alternation between different code points. It also (13.1) favors recognition of some dialectal word equivalences as variants, specifically between words in the archaizing *katharevousa* standard and the modern *dimotiki* standard language. The Latin report (6) – although it ultimately requests no variants – considers upper/lower case, display forms, glyph identity, decorative forms, issues with diacritics, and punctuation marks.

The Chinese report (5) has a different set of preoccupations since its script is ideographic rather than alphabetic, especially so since the report largely excludes Japanese and Korean with their additional phonetic systems hiragana, katakana and hangul. It excludes half-width characters (as not used for Chinese) and also homographs: here a single character has the same code, glyph and pronunciation, but appears to have multiple meanings. It recognizes variant characters, however, in two issues: the nexus between Simplified and Traditional characters (as a species of "regional variation"); and the Generic variants, where (owing to Unicode's historic policies in defining Unified Han script) subtly different forms of glyphs (derived from differing authoritative sources) have been assigned different code points, although they are functionally non-distinct in Chinese.

Further comparison of the instances and generalizations made by the various groups can be found in section 4 of this report.

The Role of Visual Similarity as a threat to Glyph Recognition

Many reports refer to the role of visual similarity, which concerns the relationship between the form of a character as presented for recognition by users, and the specific character identities assigned by the Unicode code points. (Arabic 6; Chinese 5; Cyrillic 4, 9.1; Devanagari 3.2; Greek 7; Latin 6.2).

The issue includes potential confusions between glyphs in different scripts: especially among Latin, Greek and Cyrillic, but also between Devanagari and other closely related Indian scripts. In fact, it has to do with problems of recognition by users rather than (directly) with clashes of rights inherent in registration. Potential for cross-script confusion is addressed in Cyrillic 4, 9.1; Devanagari 3.4, 4.1; Greek 7; Latin 6.8, 7.

Non-Variant Script Issues

A number of technical issues have arisen in considering the need for variants which are not, strictly speaking, directed at variants. These include:

- available fonts and their impact on glyph shape and recognition (Arabic Appendix C);

- the admissibility of zero-width characters ZWJ and ZWNJ, for the control of glyph rendering (Arabic 5.21 and Appendix D, Devanagari 4.3);
- the admissibility of characters outside the code-block assigned to a given script, notably the (extremely similar) apostrophe, saltillo or turned comma (Cyrillic 3.6, 9.2, Appendix A; Devanagari 3.4, Latin 6.7).

Other User Experience Issues

Some of the reports dwell on specifics of the user situations for Internet use and connections in their specific script areas.

Arabic 13 considers the inadequacies of computer systems (especially their keyboards and operating systems) to input and process the full range of Arabic characters in different regions (cf. Devanagari 5.3); the confusion of font differences resulting from the historic variety of writing styles (and cf. Cyrillic 3.2, 3.3, 9.1); the intrinsic problems of reversing text direction in using a right-to-left script with many left-to-right elements; and the current lack of penetration of IDNs into a variety of computer applications used by Arabic-script users.

Chinese 6.1, 6.2 emphasizes the steep learning curve for users of information technology in China, resulting from historically low levels of technical education, and a dramatic increase in take-up over recent years. The result is inferred to be a wide-scale requirement of Chinese users to have computer systems for use in which they can make the same assumptions as in the rest of their literate practice (and cf. the characterization of the Indian situation in Devanagari 5.5, and of the Greeks in Greek 7).

Latin 4 highlights the implications of this for policy on upper/lower case sensitivity and web-browser behavior. Latin 5 points out that the constraints on usability of available characters which apply to an IDN environment have no precedent in people’s prior experience with ASCII-coded information technology.

Evaluation of Applications, Registration and Operations

Towards the end of all the reports, consideration is given to administrative concerns: how are applications for registration to be evaluated and charged, how are registry records to be kept, how is security to be managed and any disputes to be resolved?

Arabic 8-11, Chinese 8-9, Devanagari 4.4-5, 5 and Latin 9 consider this conglomerate of issues for their respective script areas, but in virtue of their administrative nature, the points at issue are much the same.

The reports all emphasize the multiplication of entities which comes about as variant-generating rules are authorized. As well as there being more potential labels in existence, the root is working with scripts that are each typically used by many languages, whilst many of the languages in turn are used in multiple national administrations. Hence there will be requirements for mutual updating on a massive scale, and contact even before labels are reserved (Latin §8).

In this new, much larger universe of discourse, it will be more challenging to keep track of what is available for users to apply for and registries to reserve, allocate, delegate or block, with legal as well as technical questions to be answered. Maintenance of labels once they are activated will be more demanding. An appropriate fee structure, as well as security procedures, will need to be defined.

As to procedures for dispute resolution, this seems to be a point of decided interest, since it is addressed at length in Arabic 12, Chinese 8.2 and Latin 10.

Conclusions

Each of the reports took the opportunity to stress particular theses identified for the relevant scripts.

Chinese 10 dwells on the need for Chinese **IDLs** and their variants to be delegated to the same entity both in Simplified and Traditional character versions.

Cyrillic 10 stresses the need for a conservative (perhaps even precautionary) approach to the admission of variants. Furthermore, it urges that selective blocking (rather than joint delegation, or some form of aliasing) is the best mode to take account of any variants which are admitted.

Greek 14 gives some explanation of the thinking behind the report's recommendations, but finishes with two "red lines": the requirement that *tonos* accentuation and distinctive final *sigma* be recognized in IDL, and the requirement that a string and all its variants be reserved for the same registrant.

Latin 11 notes that the reception of labels in Latin script is language-dependent, but the script itself is language-independent; it is therefore impractical to signalize any particular relation between two code points in the Latin repertoire as variants of one another. There must furthermore be specific linguistic justification for any Latin character whose inclusion is sought in a label for registration.

Appendix 4: Terminology

Purpose

This section includes newly-introduced terms from this report, and terms that may be unfamiliar to the casual reader.

To the extent possible, we have tried to rely on documents that have been developed outside ICANN. The primary documents used are:

- Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework (RFC 5890);
- Terminology Used in Internationalization in the IETF (RFC 6365);
- The Unicode standard including the standard annexes (Unicode 6.0.0; a new release of Unicode, 6.1.0, is slated for release in February 2012. Definitions here are based on 6.0, but do not appear to be different from the beta 6.1 release as of this writing);
- Each of the six Variant Issues Project case study reports (available from http://www.icann.org/en/announcements/announcement-4-03oct11-en.htm).

If there are script-specific terms, they are copied directly from their respective reports, and contain cross-references and discussion that makes sense only in the context of those reports. They are included here for convenience, and copied verbatim (with the exception of the introduction of the report name) in order to avoid the accidental introduction of any errors in meaning.

Methodology

In this section, we describe our proposed methodology for synthesizing these definitions.

1. If a term is defined in RFC 5890, RFC 6365, the Unicode Standard, we copy the relevant definitions from those documents and also refer to those documents. The reader is strongly advised to read the text in the original, because the sense of the term may be obscured by being taken out of its context.

2. For case study specific terms, we copy directly from the definition in the relevant case study report.

3. Where terms are defined differently by different teams, the terms have been harmonized and, if need be, expanded upon to weld them into a single term useful for the unified report.

4. If a term is defined in one case study report but is generally useful, it has been taken over by this document. We have made no effort to remain consistent with the use

of that term in the original report on the grounds that this approach is really just a variation of (3), above.

It is important to note that these terms are sometimes inspired by, but do not apply to, the specific script team reports; if those reports needed special terminology, they used it, as well as the common set of definitions originally proffered for their use. Some terms will not be useful for every script.

Format of the Definitions in this Document

In the body of this document, the source for the definition is indicated with a footnote, at the end of the definition itself. Many definitions do not have such a footnote, which means that the definitions were crafted originally for this document or are merged from more than one source as noted above. For case study specific teams, they are noted with a short form for the team report (e.g. “Arabic-VIP”).

There may be commentary and examples after the definitions, delimited by the presence of the footnote. In those cases, the part before the footnote is the definition that comes from the original source, and the part after is commentary that is not a definition (such as an example or further exposition). Terms created for this document have such discussion demarcated by the label “Discussion:”

When terms appear for the first time in the body of the report, they are set in bold type. Thereafter, they are used as normal.

Abstract Character:	A unit of information used for the organization, control, or representation of textual data.[33]
Activation:	An action taken on a given label with respect to a zone, indicating that there are DNS resource records at that node name; or else that there are subordinate names to that name, even though there are no resource records at that node name.
A-label:	An ASCII-Compatible Encoding form of an IDNA-valid string.[34] The full definition for A-label is in RFC 5890, together with the definition of U-label and some ancillary definitions. The reader is urged strongly to see that document; most of the following discussion and more is in RFC 5890, and trying to talk about IDNA without understanding that document is likely to fail to use certain terms correctly. A-labels must be complete labels: IDNA is defined for labels, not for parts of them and

[33] Unicode Standard, section 3.4, D7
[34] RFC 5890

	not for complete domain names. By definition, every A-label begins with the IDNA ACE prefix, "xn--", followed by a string that is a valid output of the Punycode algorithm (RFC 3492) and hence a maximum of 59 ASCII characters in length. The prefix and string together must conform to all requirements for a label that can be stored in the DNS including conformance to the rules for LDH labels. Apart from all the other requirements, a string can only be an A-label if it can be decoded into a U-label using the Punycode algorithm, which U-label can be decoded back into the same original string using the same algorithm.
Alias:	An action performed on a given label with respect to a zone, using techniques that redirect a name or a tree, effectively substituting one label for another during DNS lookup.
Allocation:	An action taken on a label with respect to a zone, whereby the label is associated administratively to some entity that has requested the label.
Alternative code points:	Code points that may be used as alternatives for code points in the zone repertoire. For full discussion, see Section 4 of the report.
Assigned Code Point:	A mapping from an Abstract Character to a particular Code Point in the code space.[35]
Blocking:	An action taken on a given label with respect to a zone, according to which the label is unavailable for allocation to anyone.
Character Variant:	In a Language Variant Table, a "Character Variant" is an entry on the second list of code points corresponding to each Valid Code Point and providing possible substitutions for it. Unlike the **Preferred Variants**, substitutions based on Character Variants are normally reserved but not actually registered (or "activated"). Character Variants appear in column 3 of the Language Variant Table. The term "Code Point Variant" is used interchangeably with this term.[36] This term should be used only when discussing implementations of RFC 3743.

[35] Unicode Standard, section 2.4

[36] RFC 3743, section 2.1.14

Code Point:	A value in the Unicode code space.[37] The meaning here is restricted to meaning D10 in the Unicode Standard, section 3.4.
Code Point Repertoire for the Zone:	Also known informally as a **zone repertoire.** A set of code points permitted in U-labels in a zone.
Code Point Variant Rules:	The rules containing the **alternative code points**, the **code point variant status**, and any flags indicating degrees of freedom in the code point variant rules. See Section 4 of the report for a full discussion.
Code Point Variant Status:	The status of a label that results from the alternation of the alternative code point for a given code point in the zone repertoire. See Section 4 of the report for a full discussion.
Delegation:	An action taken on a given label with respect to a zone, indicating that in that zone there are NS resource records at the label.
Domain:	A domain is identified by a domain name, and consists of that part of the domain name space that is at or below the domain name which specifies the domain.[38] The Domain Name System (DNS) name space is a tree structure, with each node and leaf on the tree corresponding to a resource set. These nodes are identified by their names, and the portion that is so named (including everything underneath) is called a "domain". For example, the domain "example.com." is inside the "com." domain, which is inside the zero-length root domain ("."). The names "two.one.example.com" and "one.example.com" are both in the "example.com" domain, with "two.one.example.com" also being inside "one.example.com". Not every domain is a **zone**.
Exchangeable:	A relationship between two or more code points such that a user may treat them as though they fill the same role in a U-label. See section 3 for discussion of the classification of variant relationships.
Font:	A collection of glyphs used for the visual depiction of character data. A font is often associated with a set of parameters (for example, size, posture, weight, and serifness), which, when set to particular values,

[37] Unicode Standard, section 3.4
[38] RFC 1034, section 3.1

	generate a collection of imagable glyphs.[39] For discussion if IDNs, it is important to remember that the font is *not* a property of a code point, and is not something that can be evaluated at registration or lookup of an IDN.
Fundamental Label:	The U-label used as the basis for producing the Variant Label Set. Discussion: In many cases, this will be the label received as a request for allocation by the registry for the zone. In cases where the Label Generation Rules are implemented using a Language Variant Table, the Fundamental Label must be constructed entirely from Valid Code Points (and might not be the label received in a request for allocation).
Glyph:	(1) An abstract form that represents one or more glyph images. (2) A synonym for glyph image. In displaying Unicode character data, one or more glyphs may be selected to depict a particular character. These glyphs are selected by a rendering engine during composition and layout processing.[40] Note that RFC 6365 uses, roughly, the second of these: A glyph is an image of a character that can be displayed after being imaged onto a display surface.[41]
Glyph Image:	The actual concrete image of a glyph representation having been rasterized or otherwise imaged onto some display surface.[42]
Homoglyph:	An abstract character or a conceptual character that is represented with the same glyph as another abstract character or conceptual character.
Homograph:	Entity sharing a written form with another entity. This term is sometimes used to refer to characters written the same way, but it has a well-established meaning with respect to words within a language. It is better to use homoglyph instead. See section 3.4.1
Internationalized Domain Label (IDL):	The term "Internationalized Domain Label" or "IDL" will be used instead of the more general term "IDN" or its equivalents.[43] This term is used in RFC 3743 to talk about individual labels that make up IDNs. It identifies the label that is under consideration under the Language Table. For the purposes of the present discussion, it is better to use the term U-label.

[39] Unicode Glossary, http://www.unicode.org/glossary/#F. See also RFC 6365
[40] Unicode Glossary, http://www.unicode.org/glossary/#G
[41] RFC 6365
[42] Unicode Glossary, http://www.unicode.org/glossary/#G
[43] RFC 3743

IDL Package:	A collection of IDLs as determined by [the guidelines in RFC 3743]. All labels in the package are "reserved", meaning they cannot be registered by anyone other than the holder of the Package. These reserved IDLs may be "activated", meaning they are actually entered into a zone file as a "Zone Variant". The IDL Package also contains identification of the language(s) associated with the registration process. The IDL and its variant labels form a single, atomic unit.[44] This term should be restricted to use conformant to RFC 3743. For a term appropriate to this report, use "IDL set" instead.
IDL set:	A label whose code points are all included in the zone repertoire, along with all of the labels arising from the application of the code point variant rules on that first label.
Internationalized Domain Name (IDN):	Domain names containing characters not included in the traditional DNS preferred form ("LDH"). IDNs under discussion are implemented using IDNA; see RFCs 5890, 5891, 5892, 5893, and (for discussion) 5894 and 5895.
Label Generation Rules:	A set of rules that govern what labels are allowed in a zone. This is not a complete definition: the discussion of this term, and its related terms, is found in section 4 of the report.
Language Variant Table:	The key mechanisms of [RFC 3743] utilize a three-column table, called a Language Variant Table, for each language permitted to be registered in the zone. Those columns are known, respectively, as "Valid Code Point," "Preferred Variant," and "Character Variant," and are defined separately ….[45] Language Variant Tables are one type of **label generation rules**. The original RFC 3743 definition has been expanded upon considerably in various deployed systems, and not every deployed table has exactly three columns. In order to reduce confusion, this document will use Language Variant Table only to refer to such tables as conform to the specification in RFC 3743.
Mirrored:	A status of an active label with respect to a zone, indicating the isomorphism of the namespace beginning with that label, and at least

[44] RFC 3743 section 2.1.18
[45] RFC 3743

	one other namespace beginning with another active label in the zone.
Preferred Variant:	In a Language Variant Table, a list of Code Points corresponding to each Valid Code Point and providing possible substitutions for it. These substitutions are "preferred" in the sense that the variant labels generated using them are normally registered in the zone file, or "activated." The Preferred Code Points appear in column 2 of the Language Variant Table. "Preferred Code Point" is used interchangeably with this term.[46] This term should be used only when discussing implementations conforming to RFC 3743.
Representation Identifier:	A tag or other mechanism used to group together a representation repertoire and its representation label rules. See the discussion in Section 4 of the report.
Representation Label Rules:	The subset of **code point variant rules** proper to a **representation repertoire**. See the discussion in Section 4 of the report.
Representation Repertoire:	A set of code points arbitrarily selected for use in the representation of some language or group of languages. See Section 4 of the report for a discussion of how the set is established.
Representation Variant Rules:	A subset of **code point variant rules** that apply to the code points of the **representation repertoire**.
Script Table:	A Script Table is a table of Unicode Code Points all having the same script property value.[47]
U-label:	An IDNA-valid string of Unicode Code Points, in Normalization Form C (NFC) and including at least one non-ASCII character, expressed in a standard Unicode Encoding Form (such as UTF-8).[48] The full definition for U-label is in RFC 5890, together with the definition of A-label and some ancillary definitions. The reader is urged strongly to see that document, because without an understanding of it the present report will not make much sense. A candidate string, to be a U-label, is subject to the constraints about permitted characters that are specified in Section 4.2 of RFC 5891 and the rules in Sections 2 and 3 of RFC 5892, and the Bidi constraints in RFC 5893 if it contains any character from

[46] RFC 3743 section 2.1.13
[47] Unicode Standard Annex #24
[48] RFC 5890

	scripts that are written right to left. Apart from all other requirements, a string can only be a U-label if it can be decoded into an A-label using the Punycode algorithm, which A-label can be decoded back into the same original string using the same algorithm.
Valid Code Point:	In a Language Variant Table, the list of code points that is permitted to be registered for that language. Any other code points, or any string containing them, will be rejected. The Valid Code Point list appears as the first column of the Language Variant Table.[49] The term Valid Code Point is deprecated in this report, because it can be confusing depending on the context. Use “Label Valid Code Point” instead.
Variant:	A term widely and somewhat carelessly used denoting some sort of relationship between two U-labels or candidate U-labels, or two DNS names or candidate names. A full discussion of what the term means and some more specific types of behavior desired is discussed at length in sections 3 and 4. The term without qualification is also almost without meaning, and the reader is urged to consult that section in order to ensure greater precision in use.
Variant Label:	An informal term for labels that arise as the result of application of the alternative code points rules, as outlined in section 4. See that section for discussion. Note that labels that do *not* arise from the application of the code point variant rules are not variant labels for the purposes of this discussion, even if someone asks that they be treated as variant labels.
Withheld:	An action taken on a given label with respect to a zone, whereby the label is set aside for possible allocation to some entity.
Zone:	A[50] division of the data in the DNS, defined by the boundaries of all the cuts in the database relative to its domain. Note that the foregoing text is not taken verbatim from RFC 1034. In particular, RFC 1034 describes the zones only after all the cuts are made in the database; but for the practical purposes of identifying the zones for a given domain name, it is enough to know where all the parent-side and child-side zone cuts are for that name. The cuts are identified by SOA (Start of Authority) Resource Records. For a complete understanding of zones, zone cuts,

[49] RFC 3743, section 2.1.12
[50] RFC 1034, section 4.2

classes, and domains, the reader is directed to RFC 1034.

Zone-permitted label: A label which is valid under the label generation rules for the zone. Discussion: For practical purposes, the test of validity under the policy will likely be performed on the U-label form.

Zone repertoire: see **Code Point Repertoire for the Zone.**

Appendix 5: Survey of IDN Practices

A survey was conducted in October 2011 of all top-level domain managers, in order to better understand the prevalence of variant usage within registrations conducted by existing domain registries. Respondents were asked to describe their existing IDN usage, and then for those that performed some kind of variant handling, to respond with details about how that work was conducted.

Of the 298 country-code or generic top-level domains in existence, responses were received from 109.

Approximately half of respondents from both categories responded that they had some form of support for IDN registration within their zone.

Of those, the majority responded they check registrations for validity against a code point repertoire of what they consider valid code points. Most of those allowable code points represented the set required to represent the language(s) served by the particular zone. Alternatively, script based approaches were also used that constrained code points to the script(s) used to represent the local language(s).

Respondents that allowed for IDN registrations were asked if they permitted mixing of code points from multiple different scripts in the same label. A small number of TLDs (12) responded that they did, although most of these responses were limited to mixing the script uses to represent the local language, with the Latin letters/digits/hyphens afforded by the conventional DNS.

Of the respondents allowing registrations of IDNs, 15 indicated they had provisions for maintaining variants. Essentially all of those respondents either used the variant approach for CJK espoused by the JET Guidelines, or along the lines of the communal approach to Arabic described in RFC 5564.

Finally, respondents that manage variants were asked if they coordinate such variants across two or more zones. Of the three that answered affirmatively, one managed the same contents between an ASCII and an IDN top-level domain; and the other two maintain a variant zone that is not listed in the DNS in anticipation that the variant label will be delegated in the root zone at a later date.

The primary aim in conducting the survey was to ensure there wasn't a methodology used for variants under deployment that the team was not aware of. The responses indicated that no such novel approaches were undertaken beyond those of which the team was already familiar.

Appendix 6: Acknowledgements

1. ICANN would like to acknowledge the work of the six case study teams: Arabic, Chinese, Cyrillic, Devanagari, Greek, and Latin for their careful and detailed study on these scripts, and the local host organizations that supported the work of the teams.

   All case study team reports can be found at http://www.icann.org/en/announcements/announcement-4-03oct11-en.htm.

2. ICANN would also like to acknowledge the contributions of the Issues Report Coordination Team: Harald Alvestrand, Edmon Chung, Raiomond Doctor, Neha Gupta, Sarmad Hussain, Manal Ismail, Akshat Joshi, Cary Karp, Mahesh Kulkarni, Xiaodong Lee (subsequently joined ICANN staff in December 2011), Evangelos Melagrakis, Alexey Mykhaylov, Panagiotis Papaspiliopoulos, Vaggelis Segredakis, James Seng, Vladimir Shadrunov, Alexei Sozonov, and Joseph Yee.

   The coordination team was comprised of experts from the case study teams, who advised ICANN in completing this integrated issues report.

   The group's work in this capacity began in October at a working session during the ICANN public meeting in Dakar, Senegal, and members have participated regularly on conference calls and review and discussion of drafts of this report.

   Mailing list archives for the team are at http://mm.icann.org/pipermail/integrated-vip/. Materials for this project are also available at https://community.icann.org/display/VIP/Home.

2. ICANN would also like to acknowledge the following consultants for their significant contributions to the creation of this report: Dennis Jennings, John Klensin, Nicholas Ostler, and Andrew Sullivan.

3. ICANN also thanks all those who followed and participated in the discussion mailing list for the Variant Issues Project (http://mm.icann.org/pipermail/vip/).

4. The following ICANN staff members contributed to this report: Francisco Arias, Carole Cornell, Kim Davies, Baher Esmat, Julie Hedlund, Patrick Jones, Hayley Laframboise, Karen Lentz, Kurt Pritz, Naela Sarras, Steve Sheng, and Leo Vegoda.