Reference Label Generation Rules (LGR) for the Second Level — Overview and Summary

REVISION – January 12, 2023 — THIS IS A PUBLIC COMMENT DRAFT

1 Overview

This document describes a set of Reference Label Generation Rules (LGRs) for the Second Level. These reference LGRs were developed according to the "Guidelines for Developing Reference LGRs for the Second Level" [Guidelines]. The set divides into *language-based* and *script-based* reference LGRs. There are some differences in the origin, methodology and design goals between these sets.

The development of some of the language-based LGRs, builds on the results of a previous project [IIS] but provides additional review and development, documentation, and translation to XML [RFC7940]. Where indicated, this may result in extending the repertoire compared to [IIS]. All other LGRs, including script LGRs, are derived from the Root Zone LGR for the corresponding script. This process the repertoire to specific languages and makes suitable additions for the second level and incorporates additional input from the same panels that developed the Root Zone LGRs. The reader of this document is assumed to be familiar with the [Guidelines].

Each reference LGR contains a description section that documents issues specific to that LGR. This overview document provides some general background related to the design and design process common to these LGRs, as well as general considerations relevant to anyone wishing to adopt or adapt these LGRs for use in a particular zone. Not all reference LGRs may be developed or updated on the same schedule; this document applies to all that have been published or are actively under development. Some statements may not apply to every LGR.

The LGRs are specific to a given language or script (and in some cases the combination of language and script) but not necessarily specific to a geographically compact user community. Each file has been reviewed by the community and some have seen additional review by one or more linguistic experts, as well as reviewed by a separate expert for DNS stability and security issues.

Some of these LGRs are more suitable to standalone use, while others have features that make it easier to support them in combination with other reference LGRs other than those intended for standalone use.

1.1 Label Generation Rules (LGR)

A set of label generation rules for a zone governs the set of labels that may be allocated and eventually delegated in a given Zone. A zone may support multiple LGRs if it caters to multiple languages or scripts. While, an applied for label is always validated using a single LGR selected at application time, collisions would be determined by an integrated LGR created for the zone as described below.

Logically, any LGR contains four parts: "...the rules that define allowable Unicode code points (the repertoire), any code point variants that can be substituted to form a variant (the variant rules), the disposition of any resulting label (whether it may be allocated, or is automatically blocked), and a set of optional whole-label evaluation rules that determine whether the output of the previous three portions is still an acceptable label..." (from [Procedure]).

The Label Generation rules are expressed using a standard format defined in "Representing Label Generation Rulesets in XML" [RFC7940][1]. The XML format does not separate the LGR cleanly into the four logical parts described above, but it does provide for a mechanical computation of the status of any label as valid or invalid, and if a valid variant, as to whether that variant is allowed to be allocated, or is instead automatically blocked.

If a zone caters to multiple languages and scripts, with each of them using separate rules, a merged, Common LGR is needed to manage interaction of labels across scripts and languages (such as blocked cross-script variants).[2] The process of creating this Common LGR is called "integration". That process is described in some detail in the overview for the Root Zone Label Generation Rules [RZ-LGR-Overview].

1.2 Reference Label Generation Rules Files

A reference LGR is a set of label generation rules that can be used as is or serve as the starting point for fine tuning an LGR for a specific combination of languages and scripts for a given zone. If a zone supports multiple languages that all share the same script, it may be simplest to just support the script LGR.

The normative definition of each reference LGR is provided as an XML file. The LGRs are expressed using a standard format defined in "Representing Label Generation Rulesets in XML" [RFC7940].

Each of the reference LGRs are provided for a specific language or script as indicated in the <language> element of the XML file, with script-specific LGRs using the "und-XXXX" form of language tag. Each LGR contains all the specifications applicable to the labels from that language (or script), and only those. Each file contains a detailed human-readable description, a repertoire with optional variants, and context or WLE Rules, as well as detailed references that link each included code point to a reference providing data for justifying its inclusion.

From each XML file, a non-normative HTML presentation is generated mechanically with additional formatting. These HTML files are provided for the convenience of the reader. The HTML presentation provides formatted Description section, is augmented by summary data as well as data extracted from the Unicode Character Database [UCD]. In addition, it provides interactive features such as links to code points, references and other items defined in the LGR.

The set of [Second-Level Reference] files can be found on this website:
https://www.icann.org/resources/pages/second-level-lgr-2015-06-21-en

[1] The remainder of this document assumes that the reader is at least familiar with some of the general concepts presented in that RFC.
[2] Even where the individual LGRs contain all the cross-script or cross-reference variants, a merged LGR is required.

2 Notes

The development and review of these reference LGRs proceeds according to the [Guidelines]. The following notes provide some additional highlights as well as information not specific to any individual reference LGR.

2.1 Repertoire

The repertoire for each reference LGR is either based on the Root Zone LGR with suitable additions for the second level, or based on a consensus repertoire derived from the sources consulted. In the case of many of the language-based LGRs, the repertoire caters to more than the code points minimally needed to write the native vocabulary of the language by including code points that are in common use for loan words and the like. Where a language has multiple user communities with some variation of usage, a single, combined LGR is produced. The details are described in each of the LGRs.

2.1.1 Root Zone LGR and Second Level Additions

Many of the reference LGRs are directly derived from the relevant Root Zone LGRs. With few exceptions, these LGRs are developed on a per script basis and they deliberately limit their repertoire to code points in everyday common use, eschewing historic, obsolete or special purpose code points in limited use. Generally, though, they tend to support all languages in active, everyday use written in a given script.[3]

In some cases, the a language-specific reference LGR has been derived by limiting the repertoire to code points used in a specific language, utilizing the language-specific source information provided available in the Root Zone LGR and additional input from other sources as documented in the LGR.

In all cases there have been additions to the repertoire specific to the second level. Generally, these include the HYPHEN-MINUS and either the common (ASCII) digits or the script-specific digits for that script or both. In isolated cases, some commonly used characters ineligible for the Root Zone were added as well. For some scripts, CONTEXTO code points (like '·', used in "l·l") or CONTEXTJ code points (like ZWJ and ZWNJ) are used in an integral way in spelling many words and it may be both possible and desirable to support them with strict limits and proper safeguards in a second level LGR (for example by enumerating specific allowable combinations). All such additions generally proceed in consultation with experts from the same panels that developed the Root Zone LGRs.

2.1.2 Sources for Repertoire other than the Root Zone

In determining the repertoire where it is not derived from the Root Zone LGR, a large number of sources was investigated, from spelling dictionaries released by language authorities, RCFs and national or international standards, to other sources such as ordinary dictionaries, the Common Locale Data Repository (a project of the Unicode Consortium) [CLDR] and finally existing IDN practice for ccTLDs aimed at users of a particular language. The sources and their contribution to the development of the repertoire are documented in detail in each of the LGRs.

[3] Not all languages that may be written in a given script use that script for common, everyday use. Also, some languages are primarily spoken languages only, or they are otherwise in very limited use. Such languages are generally ruled out as justification for adding characters or features to an LGR and are only incidentally supported.

In some cases there has been direct community input into the design of the reference LGR.

The repertoire for any language-LGR, though derived independently and from different sources, is typically a subset of the reference LGR for the corresponding script.

2.2 Extended Code Points

Many, though not all, of the languages supported by language-based reference LGRs are written by compact communities that are in contact with other languages in the same region or in the same country. In those cases, native users may have familiarity with or need for access to an extended set of code points, for example for names of people or places. Certain of the reference LGRs languages provide for those code points by listing them as "extended-cp" if they are not already catered for in the core repertoire. As written, the LGRs treat these extended code points as ineligible for a label, but registries could easily remove the restriction to tailor such reference LGR to their needs. (See also Section 2.6.1, "Reference LGR-specific Rules").

This is in contrast to script-based reference LGRs that typically provide for the full repertoire needed for all languages sharing a common script, or existing country-based LGRs, that provide for the needs of users from the same country or territory, irrespective of whether they write a majority or minority language prevalent in the country.

2.3 Excluded Code Points

For most languages or scripts, there are among the consulted sources some that include a number of code points that are very rarely used, or that are historic or limited to special purposes, like poetry and religious works. Such uses are rarely germane to IDNs, and if included, could confuse users not familiar with them.

Consequently, such code points have been excluded from these reference LGRs and documented as such; either explicitly in the LGR, or implicitly in the background documentation for the cited Root Zone LGR development (see, for example, code points excluded from the [MSR]). In contrast to the "extended" code points, which are specifically called out as likely targets for customization, excluded code points can be safely left unsupported.

If it is felt that some excluded code points (or any other code point not included) might be required, the reason could well be that the scope of the LGR no longer fits the original design, but has morphed from, for example, a language-based LGR to a regional or multilingual LGR. That would be an indication that the chosen reference LGR doesn't really apply and a better approach might be to support the whole script, or cut down a script LGR to size for regional use.

2.4 Sequences and Context Constraints on Repertoire Elements

Certain code points tend to occur only in fixed combinations. The repertoire contains such code points only as part of an explicitly specified code point sequence. This prevents unneeded combinations and primarily applies to combining marks such as diacritics (but also to certain clusters in complex scripts). Another important case is the use of invisible joiner or non-joiner characters the use of which should be

tightly controlled to specific instances where their use is required (and expected) in the writing system as used for everyday communication.

Rendering systems may fail to provide a predictable presentation of combining marks if they are present outside of expected contexts, whether applied to unexpected base characters or, for example, repeatedly applied. In the latter case, in particular, there is a danger of "overprinting", which would mask the presence of an extraneous diacritic. Finally, some diacritics are easily confused with others. Allowing unrestricted combinations would allow these diacritics to be applied to base characters that normally take different diacritics, greatly adding to the risk of creating confusable labels.

Join controls that are present outside their expected context might be ignored in rendering, in the worst case, and because they are invisible by default, would result in two labels that are indistinguishable yet contain different code point sequences.

Additional constraints are provided by Whole Label Evaluation and Context Rules (see Section 2.6). Whole label rules apply to the whole, labels, while context rules are evaluated at a given code point offset in the label. Where both sequences and their constituent elements coexist, all valid partitions of the label are evaluated for variants and disposition. Only the context rules for the members of a given partition are considered; a context restriction on a defined sequence does not apply to a partition where the constituent code points are taken individually and vice versa. Sequences can thus combine code points in ways otherwise prohibited by context rules.

2.5 Variants

A *variant label* can be defined in cases where two labels are indistinguishable, or where for reasons of the writing system, users are prepared to accept one for the other. They are implemented in LGRs by means of defining *code point variants*.

Before RFC 7940 provided a generalized method for expressing variants, the use of variants was limited to a few scripts for which specific RFCs provided both the notation and script-specific rationale.

Since then, and especially during the development of the Root Zone LGR, a more generalized understanding of the usefulness of variants has emerged. In this understanding "blocked" variants perform the first line of defense against duplicate registration of labels that may be indistinguishable to users, or otherwise readily accepted as "same". Any collision between labels based on blocked variants can be detected and handled without manual attention.

Blocked variants are particularly useful in addressing security concerns for zones that support multiple scripts, where certain labels in one script can look like an ASCII label or an IDN label in another script.

Blocked variants are mutually exclusive, but that also prevents dual registration by the same entity. Judiciously used, "allocatable" variants provide a safety valve, by allowing a single entity to register two (or more) labels that would otherwise conflict. This may be appropriate whenever two labels aren't visually identical, but represent two spellings that users may not treat as different, where one of the

spellings is a widely accepted fallback, or where each spelling is confined to a specific sub-group of users[4].

Code point variants defined in an LGR must have symmetric and transitive mappings. In a few cases, this can lead to an imposition of in-repertoire variants for an LGR as result of cross-script variants. For example, the Greek LGR, by long-standing practice, treats 'ι' and 'ϊ' as variants, while the Root Zone LGR work identified the need to map these two to Latin letters 'i' and 'ï' respectively (the Latin and Greek version of the letters are not rendered distinct in all fonts). As a result of the required transitivity, the two Latin letters would pick up an in-script variant mapping in any zone where Greek is also supported.

Now, for English, not allowing a registration of both the labels "naive" and "naïve" to some unrelated entities can be seen as a win for security with limited downside. However, in some other contexts, such imposed in-script variants may limit the availability of some labels as soon as such a "variant" by transitivity has been registered.

In this example, the two spellings both exist, and the fallback (without diaeresis) is readily accepted by all users of English. However, in cases where one or the other two spellings is systematically preferred by different communities, it may be advisable to allow one of the labels as an "allocatable" variant, so that a web resource can be offered to both communities with their preferred spelling.

The reference LGR for the Latin script, for example, following the Root Zone LGR from which it is derived, contains all the variant mappings needed for integration with other scripts, particularly Greek and Cyrillic, but also others. On the other hand, the various language LGRs that use the Latin script are not configured to be used with other LGRs or other scripts in the same zone. They are standalone LGRs that do not contain variant definitions for interoperability. If it is desired to support multiple languages using the Latin script, it is preferable to use the reference LGR for the Latin script as a starting point.

Finally, it should be noted that not all kinds and degrees of label confusion can be handled with variants. For some, other methods for detecting and mitigating confusable labels must be used.

2.5.1 Variants Defined

Several of the language-based LGRs do not include the definition of any variants, but most of the script-based LGRs do. Where variants are included, their selection is informed by existing registry practice, as well as by the work performed at ICANN on the script LGRs for the Root Zone. Any cross-repertoire (or cross-script) variants identified in the Root Zone have been retained here for use in zones that support reference LGRs for more than one script or language. (See Section 4, "Use of Multiple Reference LGRs in the Same Zone").

Language reference LGRs not derived from Root Zone LGRs lack cross-script variant definitions. That makes it difficult to use the LGR other than "standalone"; they are not recommended for use in combination with other scripts or languages in the same zone. Instead, some of these LGRs may have in-repertoire variants that reflect language-specific practice, but would be hard to generalize.

[4] This most commonly refers to systematic letter substitutions, like traditional *vs.* simplified Han ideographs, and not so much to cases like "color" *vs.* "colour" that are not systematic and have exceptions, such as "or" *vs.* "our".

Under the assumption that LGRs not limited to standalone use will be used in concert with other scripts, in-repertoire variants that would be imposed by other scripts or languages are identified. In addition, some or all of the direct variant mappings to other scripts may be listed. However, actual collision testing requires full variant sets, containing all symmetric and transitive mappings. Creating these sets would require integrating all supported LGRs and using the resulting merged or "Common" LGR for all collision testing. (See also Root Zone LGR Overview [RZ-LGR-5-Overview] for a description of that process.)

Because non-IDN labels (so called LDH labels) never had variants, no variants will be defined that would impose any variant relation inside the ASCII set. This makes it possible to retrofit LGRs built on these Reference LGRs into any zone that already supports LDH labels.

However, any delegated LDH labels in that zone will now have variants among the IDN labels, and those IDN labels are blocked from being allocated. After the first IDN labels have been allocated, they may have LDH variant labels and thus block allocation of some future LDH labels.

Any label containing a "unique" code point, that is one that does not have a variant, will itself be unique. As measured by typical word lists tested against these LGRs, the percentage of unique labels can be very high, even for LGRs where significant numbers of code points have variants defined.

2.5.2 Digit Variants

All the LGRs support the common (ASCII) digits. Any script-specific (or native) digits are treated as semantic variants[5] of the corresponding common digits. In zones where multiple scripts are present, all digit sets would become semantic variants of each other as required by transitivity.

In a few cases, different sets of native digits across scripts share forms that suggest the need to include variants based on homoglyph relations. There are cases where digits of different numerical value are homoglyphs of each other and defining them as cross-script variants would create potential conflict with variants based on numeric equivalence. (These are not defined as variants in the reference LGRs).

Some scripts share a single shape for use as both letter and digit; this will make them in-script variants of each other. As a consequence of transitivity, any letters affected will become cross-script variants of all corresponding digits in the other scripts (by value). However, the common digit 0 is not treated as a variant of letter 'o' – because that variant relation does not exist in LDH (non-IDN) labels.

If support were added later for the native digits for any script, some in-script variant relations might need to be added. In zones where multiple scripts are present, it might be difficult to accommodate both the required in-script homoglyph variants as well as variants based on numeric equivalence; thus extensions to the sets of supported digits need to proceed with extreme caution and deliberation.

[5] A semantic variant indicates a code point that has the "same meaning". For digits, this is taken as having the same value. For example "123" and "١٢٣" both represent the same value and to users of Arabic may be interpreted the same, ignoring the choice of digit set as irrelevant.

Finally, some native digits are homoglyphs of or are highly confusable with other code points (letters or digits) outside the script.[6] It is generally not possible to satisfy both the semantic variant relation and such cross-script homoglyph variants in a consistent way, while also maintaining transitivity. As a result, these reference LGRs prioritize the semantic variant relationships among digits, with the thought that not all second level LGRs are necessarily in zones that support multiple scripts, but for which allowing two sets of digits immediately creates an in-script issue requiring semantic variants. Therefore, any need for mutual exclusion of labels based on confusable digit shapes (homoglyphs) across repertoires would need to be addressed outside the reference LGR, for example via additional registry policies.

2.5.3 Variant Dispositions

All LGRs with variants support the disposition "blocked", meaning that either a label or its variant may be allocated, but never both. Some LGRs support "allocatable" variants, meaning that both the label or the variant or both may be allocated to the same registrant. A few LGRs contain additional dispositions, such as "activated", which implies that a label *should* be allocated. There are "optional" forms of some variant types defined that allow for simple customization of variant dispositions as described in the particular LGR. (In all LGRs, the original label, if not blocked or invalid, is reported as "valid".)

2.5.4 Multiple Allocatable variants

The Chinese LGR resolves variant *labels* into "allocatable" and "blocked" only, however it utilizes a rather specific form of *code point* variant subtyping in an attempt to keep the number of allocatable variants manageable. The details are described in the LGR and references cited therein. (Some other LGRs use similar, but simpler forms of variant subtyping, for example, Greek and Latin).

Several LGRs use whole label evaluation rules to limit the available variant labels to those that consistently use either the one or the other variant code point throughout the label, disallowing mixed labels. In some cases, this is implemented by "no-mix" rules for the specific variants, in other cases there are rules that require that all variants to be in a single sub-repertoire (for example, a repertoire for a specific language). Occasionally, both of these approaches are used.

Finally, some LGRs support single "fallback" variants as allocatable variants from any of a number of permutations of the label using one or more instances of some special characters. (See below).

Because the DNS does not natively support variant labels there is a cost to having multiple variants delegated, and thus a need to add such mitigation. The mitigation approaches in a particular reference LGR are not always sufficient on their own because it may not be possible to write generalized rules preferring some variants to others. Thus, registries should implement additional policies to limit the number of variant labels actually delegated.

Please note that even where typical labels may not generate an inordinate number of labels, it is possible to create pathological labels for some LGRs for which the theoretical number of allocatable variants could cause enumeration of allocatable variants to fail by exhausting resources.

[6] In contrast, where scripts share shapes between some of their own letters and digits, variants should be defined, even if that means some letters become variants of unrelated digits by transitivity.

2.5.5 Allocatable Fallback Variants

A *fallback variant label* is a single allocatable variant label that uses substitute code points or sequences for code points or sequences not available (or not allowed) in some contexts, but that would be required for a linguistically accurate rendering of some label. A fallback may not look like the intended label, but is generally recognized as an established "poor man's substitute" for that label.

Widely encountered examples of fallback variants involve the use of ASCII equivalents for Latin letters or sequences. For example, the middle dot in Catalan "l·l" cannot be rendered in ASCII, but the sequence "ll" can be used as a fallback. While it loses an important spelling distinction, for a domain name it may be more important that users without access to a Catalan keyboard can type the "ll" sequence.

Other scripts and languages have similar examples. For example, for the Sinhala LGR, all sequences containing a Zero Width Joiner (ZWJ) have a fallback variant without the ZWJ.

Common to all fallback variants is the desire to allow both the registration of a "preferred" or "intended" spelling of a label and a single fallback that avoids any use of special characters.[7] To support this, the variant mapping from the standard letter or sequence of is of type "fallback", while the mapping in the other direction is of type "blocked".

A third variant type "r-original" is used as a reflexive variant to identify code points that have a fallback mapping, but that appear in their non-fallback form in the original label, and thus "map to themselves". A set of default actions is defined for use with all second level reference LGRs. These actions resolve as "allocatable" any label where all variants are of type "fallback", and as "valid" any label where all variants are of type "r-original". Any variant with a mix of variant type resolves as "blocked".

With this system, registrants are able to apply for one label of their chosen spelling, no matter how many instances or permutations of non-fallback and fallback characters it contains, plus a single fallback variant containing no instance of any non-fallback character or sequence. Any other labels, such as those containing some other mix of non-fallback and fallbacks for the same label, would be blocked variants. As a result, fallback variants are limited to a single allocatable variant label, but also block any other variations of the same label beyond the one selected by the applicant.

Note that if the fallback itself is applied for as the intended label, no other label is allocatable and all other variant spellings containing non-fallback characters are blocked.

2.5.6 Multiple Blocked Variants

There is no limit to the number of blocked variants a label may produce under any of these reference LGRs. An exhaustive enumeration may be computationally infeasible. It is also not required for detecting collisions between labels that are variants of each other. Such testing can be performed efficiently by

[7] Note that a "fallback" representation may spell a perfectly acceptable word in its own right, or be an alternative preferred spelling for the same word in a different locale. For example German written in Switzerland always uses the "ss", while German written in Germany alternates between "ss" and "ß" depending on the spelling of a given word. Nevertheless, for users in Germany, "ss" is both a commonly accepted fallback for "ß" in contexts where the latter is not available, as well as the preferred spelling in words where an "ß" should not be used.

computing a single “index variant” followed by a comparison of these index variants. The performance of this operation does not depend on the theoretical number of blocked variants. (For a more detailed description, and for additional notes how to ensure that index variant calculation is well-behaved, see [RZ-LGR-Overview], but also see Section 4 below).

2.5.7 Context Rules for Variants

Some variants require context rules to be well-behaved. See RFC 8228 or the discussion of this issue in Section 6, “Design Notes for the Root Zone LGR” in [RZ-LGR-Overview]. Any such context rules from the corresponding Root Zone LGRs have been retained, and a few additional ones have been added where needed by new repertoire (for an example, see the Devanagari LGR’s treatment of the HYPHEN).

Context rules for variants are also used in these reference LGRs to mark variants as “optionally-enabled”. The context rule “optionally-enabled” matches any label (it always succeeds). Any variant marked with when=“optionally-enabled” is in force, while marking it with not-when=”optionally-enabled” causes the variant definition to be ignored in processing.

This scheme allows for simple adjustments as long as all variant mappings to and from the same code point are either enabled or disabled at the same time. (Due to a limitation of a single context at a time this scheme cannot be applied to make optional any variants that already carry other types of context restrictions).

2.5.8 Implicit Cross-script Variants

The listing of variants in the Reference LGRs may omit some cross script variants. Once defined in at least one LGR, they are incorporated into the merged common LGR where they are used for calculating index variant used to check labels for collision, both in-script and across scripts. For this type of implicit variant the merged LGR contains a full listing of the variant set under transitive closure, but not all of the reference LGRs repeat the mapping. Most reference LGRs only list their in-script variants.

A second form of implicit variant exists when a code point sequence has both an explicitly defined variant, but also a cross-script variant computed from variants of its constituent elements. For example, the Latin sequence “ss” has an in-script variant “ß” (sharp s or eszett) as well as a cross-script variant to the Greek beta (β). However, the code point “s” has several cross-script variants, for example to the Cyrillic letter Es “s”. That means that the Latin sequence “ss” also has an implicit variant to Cyrillic “ss”, but by transitivity, the Cyrillic “ss” implicitly has “ß” and “β” as its variants. Note that Cyrillic “ss” as a sequence is redundant: it is not a target of an in-script variant, nor does it add to the space of available labels, as the use of two single “s” in a row is already allowed.

For this second type of implicit variant, mappings involving sequences from other scripts are elided if they don’t contribute to the index variant calculation. The Cyrillic LGR, having a mapping from “s” to “s”, does not need a redundant sequence definition plus mapping from “ss” to Latin “ss” to make index variant calculation possible. And a mapping from “ss” to “β” would likewise not contribute, because the

Greek LGR provides the mapping from “β” to Latin “ss”, allowing both a Greek and a Cyrillic label to map to the same index variant.[8]

This elision is not possible if variants are allocatable in one or both directions, or if sequences are not redundant (for example if they contain unique code points, override context restrictions between their constituent code points, or where the constituent code points don’t produce the same index variant).

2.6 Whole Label Evaluation (WLE) and Context Rules

WLE and context rules implement a further constraint on valid labels, for example, by limiting certain code points from occurring at the beginning of a label or occurring repeatedly or simultaneously with other code points in the same label. Context rules are a specialized form of a WLE rule that defines a constraint on the surrounding context for a given code point at that position in a given label (see [RFC7940]).

Some code points may be restricted not by context rules of their own, but due to context rules on all the other code points. Such code points are said to have “implicit” context restrictions. Finally, any code point sequence defined will implicitly override any restrictions resulting from the application of context rules for its constituting elements. Both WLE rules and context rules defined for the sequence as a whole remain unaffected, as would any implicit context restrictions evaluated for the neighboring code points.
Protocol-defined Rules

Because the XML format for the LGR supports machine-evaluation of labels for validity, these reference LGRs include all relevant constraints on labels defined in the IDNA protocol itself, other than normalization. In this way, the LGRs can be used to validate all constraints on any normalized label in one pass. For the purpose of these reference LGRs, an additional rule preventing the mixing of any two digit sets in the same LGR has been adopted project-wide.

Common rules:

- Hyphen Restrictions — restricts the allowable placement of U+002D (-) HYPHEN (no leading/ending hyphen and no hyphen in 3-4 position). These constraints are described in section 4.2.3.1 of [RFC5891].
- Leading Combining Marks — restricts the allowable placement for combining marks (no leading combining mark). This constraint is described in section 4.2.3.2 of [RFC5891].
- Digit Mixing — all LGRs support a rule to prevent mixing of multiple sets of digits in the same label.

Rules for Right-to-Left labels:

- Leading Digit — restricts the allowable placement of digits for right-to-left labels (no leading digit in RTL label). This constraint is described in section 2.1 of [RFC5893].

[8] Because of the possible interaction between visual, semantic or fallback variants, an implicit variant, as in this example, can result in a blocked variant between seemingly unrelated labels, without this connection being immediately apparent in the LGR documents due to the way their presentation is streamlined.

- Mixed Digits — prevents the mixing of European and Arabic (Indic) digits. This constraint is described in appendix A.8 and A.9 of [RFC5893]. (In these reference LGRs this is a subset of the general digit mixing restriction.)

Context rules:

- Japanese in Label — restricts the occurrence of KATAKANA MIDDLE DOT to labels containing at least one code point from any of these scripts: Han, Hiragana, or Katakana; This rule is described in Appendix A.7 of [RFC5892].[9]

Whole label rules:

- No ASCII only Label —restricts IDN labels to those having at least one non-ASCII code point. {RFC 5891]. [10]

For these reference LGRs, the protocol-derived rules, other than the common rules, have only been included if they are needed for labels in the given script. The description section of each LGR file lists the rules and their associated references.

If a label to be validated has already been tested against protocol-derived constraints by the time the LGR is applied, these rules would be redundant and could be removed.

2.6.1 Reference LGR-specific Rules

A number of the LGRs contain additional LGR-specific WLE rules, reflecting further constraints on possible labels based on the nature of the language or script. These are documented in detail in the description section of the respective LGRs. These rules are generally derived from the Root Zone LGRs with suitable extensions in case of added repertoire elements (for example, see the Thai language LGR for the addition of a Thai abbreviation mark with Thai-specific rules).

The majority of these rules take the form of context rules on a given code point or sequence. (Sequences may be defined for the purposes of selectively overriding the constraints defined for single code points.)

In the case of many complex scripts, readers and layout engines expect the label to be series of valid syllables according to the rules of the writing system. Even though Unicode may encode the individual components of such syllables, arbitrary sequences of code points are not only unexpected, but can lead to security and predictability issues for identifiers. This is in contrast to alphabetic or ideographic scripts where the letter or ideograph is considered the unit, and arbitrary sequences are being used in identifiers. For that reason, complex script LGRs have WLE or context rules enforcing the deep syllabic structure of the writing system, while generally not attempting to enforce “spelling rules”. While it may thus not be possible to make labels using non-existing syllables, there are generally no restrictions against creating “nonsense” words from well-formed syllables.

[9] The Katakana Middle Dot is not currently supported in any reference LGR.
[10] By community request, this rule is not enforced in these reference LGRs, meaning that they can be used to apply for whatever subset of LDH labels the LGR may permit (most commonly ASCII digits plus Hyphen).

2.6.2 Special Rule for Optional Repertoire Items

Finally, some of the second level reference LGRs use a special context rule to support adapting a reference LGR to a specific zone. (For details, see Section 2.2). By default, this rule may be present in any table, whether actually associated with any code points or not.

Special rule present in some reference LGRs:

- Extended-cp —this context rule always fails. That means, as published, the LGRs do not allow the code points identified as extended by having been given a context of “extended-cp”.

Simply changing that rule so it always matches would enable the entire set of extended code points without the need to edit the list of characters. Alternatively, the context condition could be removed from individual code points, thus enabling them one by one. Likewise, to create a subset, a code point can be disabled by adding the “extended-cp” context condition. Doing so would mark the code point as deliberately not included instead of merely omitted.

Alternatively, a copy of that rule, named “excluded-cp” could be used for the latter purpose.

2.7 Metadata

The XML file format defines a number of elements for metadata. Several elements are not relevant to reference LGRs, but would be relevant to actual, deployed LGRs. These elements include <scope> element to define the zone to which the LGR applies, or the dates <validity-start>, and <validity-end>.

In adopting a reference LGR as the LGR for a specific zone, values for these elements should be supplied. For more details, see [RFC7940].

3 Use of Reference LGRs

The information in these reference LGRs represents the best available knowledge of the code points suitable for IDNs for users of a given language or script. As [RFC6912] makes clear, IDNs are intended to be *reasonable mnemonics*, and not for the faithful representation of any possible text in a given language. However, what is a reasonable mnemonic is informed by the language of the user community. Letters or diacritics that are unfamiliar in appearance do not make good mnemonics even if they are technically part of the same script. In that sense, where these LGRs have been developed for a given language that fact can also be understood as meaning that they were developed with the expectations of users of a particular language in mind.

3.1 Subset Repertoires

Creating a subset of one of these LGRs would generally represent a more conservative choice (see [RFC6912]). However, the final choice will always have to be made in tension between the two goals of usability and conservatism. There are several issues to consider when contemplating the creation of a subset. The first affects usability. For example, consider the case of reducing any Latin-based LGR to the letters "a-z". This is undoubtedly a conservative choice. But, it also eliminates any gain in usability

compared to non-IDN labels. A subset should always be a carefully designed consistent whole. (See also Section 2.2 and Section 2.6)

The next concern applies to LGRs that contain variants. For those LGRs, the effect of subsetting on the variant sets must be reviewed thoroughly. For each code point to be removed, all variant mappings related to that code point must also be removed. Once these are removed, it may not be possible to add the code point back again in a future version of the LGR due to the risk of stability issues. Finally, any rule that depends on the definition of a given code point must be updated if that code point is removed.

3.2 Overlapping Repertoires

Additional policies, variants and rules may be needed if any of these reference LGRs is adopted along with other LGRs that have an overlapping repertoire.[11] This is especially relevant in the case of LGRs defining variants or LGR-specific rules.

3.3 Repertoire Extensions

There are two ways of extending these LGRs. The first is by allowing additional code points that are considered widely used in the context of a given language or script, or that belong to languages in the region and show up in names or unassimilated loan words. Some of the reference LGRs directly provide information on a suitable set of such extended code points (see Section 2.2).

The second is the use of a number of language-based reference LGRs as "building blocks" in assembling local, regional or script-based LGRs. When used in that fashion, care must be taken so that the resulting LGR provides for a consistent treatment of variants and WLE rules.

Any LGRs to be combined should be from a common script. The issue of mixed script labels is addressed in [RFC5890]. In combining LGRs into a single LGR it is recommended to first combine their core repertoires and, after eliminating duplicates, to consider possible additions from the extended sets separately. A combined LGR could have multiple "language" elements to indicate the range of languages covered, or a single language element indicating the script (see [RFC7940]).

For many scripts, the task of creating a script-wide reference LGR has been carried out based on the Root Zone LGR effort, but in some cases, regional LGRs might be of interest and would benefit from the construction process outlined above.

3.4 Variants and Rules

When merging LGRs, or when using LGRs with overlapping repertoires in the same zone, the "rules" element in the XML must be given special scrutiny. Some of the "rule" and "class" elements may be merged safely. Others may have to be renamed to keep them distinct. "Action" elements must be present in the order required for proper precedence in the merged XML.

That said, most of the LGRs presented here have a generic, default "rules" element. Any two LGRs with only the default rules can be merged and a single copy of the default rules appended.

[11] An example of an overlapping repertoire is the shared use of the Han script for Japanese and Chinese.

Merging repertoires for the purpose of creating an extended LGR (for example a regional LGR supporting a number of languages, but not all uses of a given scripts) is a different process from mechanically merging LGRs to create an "integrated" LGR that can be used in detecting label collision. For more information on the process of integrating LGRs, see [RZ-LGR-5-Overview]).

4 Use of Multiple Reference LGRs in the Same Zone

If a zone supports multiple reference LGRs, cross-repertoire (or cross-script) variant labels may exist (see Section 6.3 "Cross-script variants" in [RZ-LGR-Overview] for a discussion). This situation arises when multiple LGRs are used, each defining the valid labels and variants for a given script or language (in contrast to the case of a combined repertoire as discussed in Section 3.3, "Repertoire Extensions"). For efficient resolution of cross-repertoire variants, a special merged or "common" LGR needs to be created that is optimized for the task. For a discussion, see Section 5.2 "Common LGR" in [RZ-LGR-Overview]). As long as the common LGR file was created using all of the multiple LGRs that have cross-repertoire variants with each other, it can be used for that purpose – even if it contains information from additional LGRs.

Note that these reference LGRs are designed with the assumption that any native digits are variants of the corresponding ASCII digits, if both occur in the same LGR. When multiple LGRs are used in the same zone, transitivity would require that all such native digits are cross-script variants of each other. The effect of this transitivity is omitted from the individual LGRs; it would have to be applied to a common LGR as part of the merge process to enable for proper cross-repertoire collision detection.

Common LGR files for some combination of these reference LGRs may be provided. If so, the preamble to the file states which files were included in its preparation.

5 Review

These reference LGRs have been reviewed by the community as part of a public review process. Certain of the reference LGRs were additionally reviewed by members of the corresponding Root Zone Generation Panels or by independent reviewers with expertise in Unicode and linguistics, as well as IDNA and DNS security. Any LGRs were updated to reflect the input from the review. For the independent expert reviewers, review results where available are found at [Second-Level Reference].

5.1 Versioning

Each reference LGR has a version number and a date of publication. Any changes or corrections to a published reference LGR result in a new version number and publication date. The nature of any corrections or changes is documented in the description. Drafts and revisions published for public comment carry the same version number as the version targeted for publication, but not the final publication date.

6 Contributors

The following cumulatively lists contributors to the development of any Second Level reference LGRs.

1. Developers

Asmus Freytag
Michel Suignard

2. Expert Reviewers

Michael Everson
Nicholas Ostler
Lu Qin
Wil Tan

3. ICANN Staff

Sarmad Hussain
Pitinan Kooarmonpatana
Anand Mishra

4. Root Zone Generation Panel Experts

An early version of all script-specific and some language-specific LGRs was reviewed by experts from the Root Zone LGR project. For a list of members for each Generation Panel, see the proposal document for the corresponding Root Zone LGR.

7 References

[CLDR] CLDR - Unicode Common Locale Data Repository: http://cldr.unicode.org

[Guidelines] Internet Corporation for Assigned Names and Numbers, “Guidelines for Developing Reference LGRs for the Second Level”. (Los Angeles, California: ICANN, October 2015) https://www.icann.org/en/system/files/files/lgr-guidelines-second-level-27may20-en.pdf

[IIS] IIS, IDN Reference Tables, https://github.com/dotse/IDN-tables

[MSR] Integration Panel, "Maximal Starting Repertoire - MSR-5: Overview and Rationale", 24-June 2021, https://www.icann.org/en/system/files/files/msr-5-overview-24jun21-en.pdf

Note: MSR-5 is based on Unicode 11.0, the latest version for which IDNA 2008 tables are available at the time of its publication. Due to the fact that most recent additions to Unicode have been for rarely used code points, the probability that future versions will impact these reference LGRs is very low.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, https://tools.ietf.org/html/rfc5890.

[RFC6912] Sullivan, A., Thaler, D., Klensin, J., and O. Kolkman, "Principles for Unicode Code Point Inclusion in Labels in the DNS", RFC 6912, April 2013, https://tools.ietf.org/html/rfc6912.

[RFC7940] Davies, K and Asmus Freytag: “Representing Label Generation Rulesets using XML”, August 2016 https://tools.ietf.org/html/rfc7940.

[RFC8228] Asmus Freytag: "Guidance on Designing Label Generation Rulesets (LGRs) Supporting Variant Labels", RFC 8228, August 2017, https://www.rfc-editor.org/info/rfc8228.

[RZ-LGR-Overview] Integration Panel, "Root Zone Label Generation Rules - LGR-5: Overview and Summary", 26 May 2022 (PDF), https://www.icann.org/sites/default/files/lgr/rz-lgr-5-overview-26may22-en.pdf

[Second-Level Reference] ICANN, Second-Level Reference Label Generation Rules, https://www.icann.org/resources/pages/second-level-lgr-2015-06-21-en